



ଓଡ଼ିଶା ରାଜ୍ୟ ମୁକ୍ତ ବିଶ୍ୱବିଦ୍ୟାଳୟ, ସମ୍ବଲପୁର, ଓଡ଼ିଶା
Odisha State Open University, Sambalpur, Odisha
Established by an Act of Government of Odisha.

**PG DIPLOMA IN COMPUTER APPLICATIONS
(PGDCA)**

CSP-44

Application Development Using PHP

Block

3

Advance PHP

Unit -1

Object Oriented Concept in PHP

Unit -2

File management and Exception Handling

Unit -3

Database Connectivity in PHP



ଓଡ଼ିଶା ରାଜ୍ୟ ମୁକ୍ତ ବିଶ୍ୱବିଦ୍ୟାଳୟ, ସମ୍ବଲପୁର, ଓଡ଼ିଶା
Odisha State Open University, Sambalpur, Odisha
Established by an Act of Government of Odisha.

EXPERT COMMITTEE

Dr P.K.Behera Reader in Computer Science Utkal University Bhubaneswar, Odisha	(Chairman)
Dr. J.R.Mohanty Professor and HOD KIIT University Bhubaneswar, Odisha	(Member)
Sh Pabitrnanda Pattnaik Scientist –E,NIC Bhubaneswar, Odisha	(Member)
Sh Malaya Kumar Das Scientist –E,NIC Bhubaneswar, Odisha	(Member)
Dr. Bhagirathi Nayak Professor and Head(IT & System) Sri Sri University Bhubaneswar, Odisha	(Member)
Dr. Manoranjan Pradhan Professor and Head(IT & System) G.I.T.A Bhubaneswar, Odisha	(Member)
Sri V.S.Sandilya Academic Consultant (I.T), Odisha State Open University, Sambalpur, Odisha	(Convener)

PG DIPLOMA IN COMPUTER APPLICATIONS

Course Writer

Mr. Aseem Kumar Patel

Academic Consultant

Odisha State Open University, Sambalpur, Odisha

Course Editor

Mr. Satya Sobhan Panigrahi

Asst. Prof, SIT, BBSR

UNIT-01

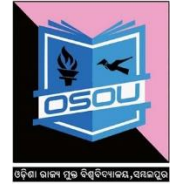
Object Oriented Concept in PHP

UNIT STRUCTURE

1.1	Introduction	01
1.2	Basics of OOP in PHP	01
1.2.1	Pillars of OOPS	01
1.3	Advantage of Object Oriented Programming	03
1.4	Understanding classes and objects	03
1.4.1	Class in PHP	04
1.4.2	Object in PHP	05
1.5	PHP Class Properties and Methods	07
1.5.1	Access Modifiers	07
1.5.2	PHP class Methods	07
1.5.3	PHP Static Class Properties and Methods	08
1.5.4	PHP Class Constants	09
1.6	Constructor and Destructor in PHP	10
1.6.1	Constructor in PHP	10
1.6.2	Destructor in PHP	12
1.7	Magic Methods in PHP	13
1.8	Inheritance in PHP	15
1.8.1	Multilevel and Multiple inheritance in PHP	17
1.9	Interface	19
1.10	Abstract class	21
1.11	Differences between abstract class and interface in PHP	22
1.12	Final class and method in PHP	23
1.13	Polymorphism	24
1.13.1	Method Overriding	24
1.14	Let us sum up	26
1.15	References	26
1.16	Model Questions	26

UNIT-01

Object Oriented Concept in PHP



1.1 Introduction

Object-oriented programming (OOP) was first introduced in php4. Area for OOP in PHP version 4 was not very vast. There were only few features available in php4. The major concept of the object oriented programming in PHP is introduced from version 5 (we commonly known as php5). Also, PHP community has the plan to modify its object model structure in a better manner in php6 (not released yet). But still, in the php5 object model is designed nicely. If you have the good understanding of OOP then you can create a very good architecture of your PHP application. You only need to know some of the basic principles of object-oriented programming and how to implement that concept of OOP in PHP. In whole series I will use abbreviation OOP for Object Oriented Programming.

The object-oriented programming style in PHP is a great way to build modular, reusable code, letting you create large applications that are relatively easy to maintain.

1.2 Basics of OOP in PHP

Object oriented programming is a design concept. OOP is nothing but a technique to design your application i.e. web based or windows based application.

In object oriented programming, everything will be around the objects and class. By using OOP in php we can create modular web application and perform any activity in the object model structure.

1.2.1 Pillars of OOPs

There are six Pillars of Object Oriented Programming:

1. Object
2. Class
3. Abstraction
4. Encapsulation
5. Inheritance
6. Polymorphism

Let's understand each of pillars

1. Object

Anything in the world is an object. Look around and you can find lots of object. Your laptop, pc, car everything is an object. In this world every object has two

thing properties and behaviors. Your car has property (color, brand name) and behavior (activity).

If you have an object for interest calculator then it has property interest rate and capital and behavior simple interest calculation and compound interest calculation.

PHP object an individual instance of the data structure defined by a class. Objects are also known as instance.



2. Class

The class is something which defines your object. For example, your class is Car. And your Honda car is object of car class.

Blueprint of the object is class. The class represents all properties and behaviors of the object. For example, your car class will define that car should have color, the number of doors and your car which is an object will have color green and 2 doors. Your car is an object of a class car. Or in terms of programming, we can say your car object is an instance of the car class. So structural representation (blueprint) of your object is class.

3. Abstraction

Abstraction means displaying only essential information and hiding the details. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation.

Consider a real life example of a man driving a car. The man only knows that pressing the accelerators will increase the speed of car or applying brakes will stop the car but he does not know about how on pressing accelerator the speed is actually increasing, he does not know about the inner mechanism of the car or the implementation of accelerator, brakes etc in the car. This is what abstraction is.

4. Encapsulation

Encapsulation means putting together all the variables (instance variables) and the methods into a single unit called Class. It also means hiding data and methods within an Object. Encapsulation provides the security that keeps data and methods safe from inadvertent changes.

5. Inheritance

Inheritance is the ability to create classes that share the attributes and methods of existing classes, but with more specific features. Inheritance is mainly used for code reusability.

When one class is derived from another class this is called an inheritance. The new class (also called subclass, derived class, child class) can inherit members

from the old class (also called superclass, parent class). The new class can also add additional properties and methods.

6. Polymorphism

Polymorphism definition is that Poly means many and morphos means forms. It describes the feature of languages that allows the same word or symbol to be interpreted correctly in different situations based on the context. Object-oriented programs are written so that the methods having the same name works differently in different context.

1.3 Advantage of Object Oriented Programming

There are various advantage of using OOP over the procedural or parallel programming. Following are some of the basic advantages of using OOPS techniques.

Re-Usability of your code: If you will use OOP technique for creating your application then it will gives you a greater re-usability. For example, if you have created calculator class at one place then you can use the same calculator class in your application.

Easy to Maintain: Application develop using OOPS technique are easier to maintain than normal programming. Again let us take an example of your interest calculator class. Suppose your business need to change the calculation logic. They want to add some charges if your capital is less than 200 USD. Just think about your application is big and developed using normal programming techniques. So first you have to analyze that at how many places we have calculated interest, and then you will change. But just think of OOPS technique. You just need to change in your method of interest calculation at one place.

Good Level of Abstraction: Abstraction means making something hidden. By using OOPS technique you are abstracting your business logic from implementation. It will provide you greater ease. Again let us take an example of interest calculator. If you have created class for interest calculation and your team is going to use that class. Now you are only concern about how interest calculation will be performed because you have created that. Your team member is always have understood that if they will set rate and capital property and apply interest calculation method then it will return interest.

Molecularity: If are creating a separate class for your every problem then you are making it modular. So if someone needs to change in the business logic part then he will always go to your business logic code part.

1.4 Understanding Classes and Objects

Classes, objects, properties, and methods are the basic building blocks that you can use to create object-oriented applications in PHP.

A PHP class is a unit of code that describes the characteristics and behaviors of something, or of a group of things.

Class is like your house blueprint. Before your house is constructed, there is a house blueprint. It is not an actual house, but a plan how this house will look like, how many rooms it will have and so on. Then the house will be constructed by following the blueprint exactly. In this analogy, the house blueprint is a class and your actual house is an object. We can have unlimited objects of a class, just like we can build unlimited exact houses by following the same house blueprint.

1.4.1 Class in PHP

Concept of class introduced from php4. But complete coverage of class like access modifier or interface is introduced from php5. Creating class is very easy in PHP. You can create class with help of using **class** keyword in PHP.

The keyword class is used to define a user defined (abstract) data type. It is then followed by the user defined class name (/identifier) and a pair of curly braces { }

e.g. class className { }

Syntax:

```
class className
{
    //variables of the class
    var $var1;
    var $var2;
    //Function of class
    function addVar ()
    {
        return $this->var1 . $this->var2;
    }
}
```

Points to be Remember:

1. Class className is created by using keyword “class”. Your name of the class will be general string without space.
2. Complete block of the class is enclosed within { }.
3. All variables of this class is defined in the beginning of the class. Variables are starting with “var” keyword which is followed by a conventional \$ variable name.
4. The variable \$this is a special variable and it refers to the same object i.e. itself.

If you want to declare \$var1 to be accessible from anywhere then you can use public \$var1 instead of var \$var1. If you will use var \$var1 in php5, the variable will be treated as public by default.

Example:

```
<? php
class Car{
    public $weight = "100kg";
    public $price = "$1000";

    function show(){
        echo "<br/>". $this->weight;
        echo "<br/>". $this->price;
    }
}
?>
```

1.4.2 Object in PHP

Classes are useless without objects. Object is an instance of your class. If you have class then you can create as many objects as you like of that class type. You can create object of your class by using “**new**” keyword.

Syntax to create object using new operator

An object is created using the keyword new to assign the class to the user defined object name.

e.g. \$myObject = new className;

Now in above code we have created object named “myObject” of class myClass. We can create multiple object of your same class. Every object is different from other.

\$objClass1 = new myClass();

\$objClass2 = new myClass();

Example:

```
<? PHP
class Car{
    public $weight = "100kg";
    public $price = "$1000";
    function show(){
        echo "<br/>". $this->weight;
        echo "<br/>". $this->price;
    }
}
```



```

$object = new Car ();           // create a class object
echo ( "<br/>".$object->price ); // access class public property
echo ( "<br/>".$object-> weight); // access class public property
$object->show (); ?>

```

The “->” symbol access operator is used to get or set an object’s attributes.

Example:-01

<? PHP

//Creating class interestCalculator

```

class interestCalculator
{
    public $rate;
    public $duration;
    public $capital;
    public function calculateInterest()
    {
        return ($this->rate*$this->duration*$this->capital)/100;
    }
}

```

```

$calculator1 = new InterestCalculator();
$calculator2 = new InterestCalculator();
$calculator1->rate = 3;
$calculator1->duration =2;
$calculator1->capital = 300;

```

```

$calculator2->rate = 3.2;
$calculator2->duration =3;
$calculator2->capital = 400;

```

```

$interest1 = $calculator1->calculateInterest();
$interest2 = $calculator2->calculateInterest();

```

```

echo "Your interest for calculator1 object is...$interest1 <br/> ";
echo "Your interest for calculator2 object is...$interest2 <br/> ";
?>

```

OUTPUT:

```

Your interest for object calculator1 is...18
Your interest for object calculator2 is...38.4

```

We have created two object of “interestCalculator” class in variable \$calculator1 and \$calculator2.

Now property value of both objects are different. for example \$calculator1 capital is 300 and \$calculator2 capital is 400. Whenever we will call “calculateInterest” function of the both object then they will calculate interest on their own properties.

1.5 PHP Class Properties and Methods

1.5.1 Access modifiers:

Each method and property has its visibility. We can set the visibility of methods and member fields.

PHP5 has three access modifiers. **public, private, protected.**

- The **public members** can be accessed everywhere, both within the class and externally.
- The **private members** is accessible only from within the class that defines it.
- The **protected members** can only be accessed within the class itself or in descendant classes.

PHP Class Properties

Properties are used to add data to a class, it's a class-specific variables.

Example:

```
<?php
class Car{
    public $color = "red";
    private $price = "$2000"; // class properties
    protected $power = "200hp";
}
$obj = new Car();
echo ( $obj->color ); // you can access only public properties
echo ( $obj->price ); /*this statement generate an error because
private properties can't access outside the class*/
?>
```

1.5.2 PHP Class Methods

Methods are class-specific function. Method declare using the function keyword precedes a method name, followed by an optional list of argument variables in parentheses. The method body is enclosed by braces.

Points to be remember about the class methods:

- Methods must be declared in the body of class
- Methods can be declared public, protected, or private.

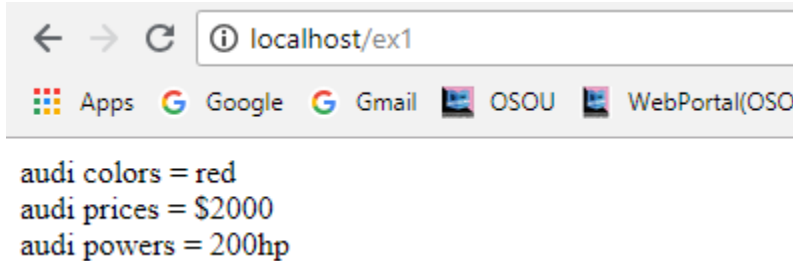
- If you omit the visibility keyword in your method declaration, the method will be declared public implicitly.

Example: ex1.php //filename

```
<?php
class Audi{
    public $colors ="red";
    private $prices="$2000"; // class properties
    protected $powers = "200hp";

public function show() { // access modifier must followed by function keyword
    echo "audi colors = ". $this->colors;
    echo "<br/> audi prices = ". $this->prices;
    echo "<br/> audi powers = ". $this->powers;
    }
}
$obj = new Audi();
$obj->show(); // calling instance show() method
?>
```

OUTPUT:



1.5.3 PHP Static Class Properties and Methods

The static keyword is used to define static methods and properties.

- static methods are callable without an instance of the object.
- \$this pseudo-variable is not available inside the method declared as static.

The static properties and methods accessed using the **scope resolution operator (::)**.

Syntax:

classname :: method_name()

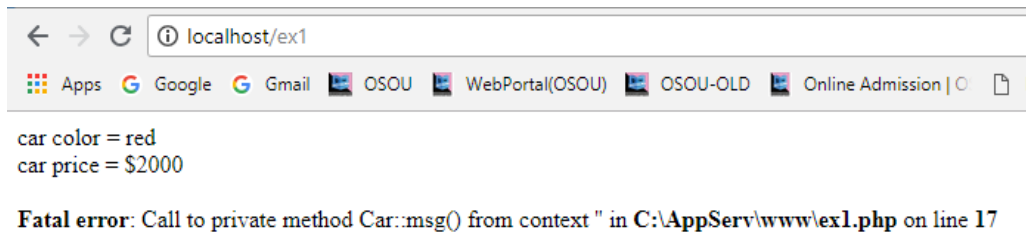
classname :: static_property_name

Example:

```
<?php
class Car{
    static public $color="red"; // static property
    static private $price="$2000"; // static property
    protected $power="200hp"; // instance property

    static public function show(){
        echo "car color = ". Car::$color;
        echo "<br/> car price = ". Car::$price; echo "<br/>";
        // you cannot use the instance property within static method
    }
    static private function msg(){
        echo "this is private static function";
    } // end of class Car
    $obj= new Car();
    Car :: show(); // calling the static show() method
    Car::msg(); // private static msg() method cannot accessed
                 outside the class
?>
```

OUTPUT:



1.5.4 PHP Class Constants

Class constants define with the const keyword. A constant variable cannot start with \$ symbol.

Syntax:

```
const PRICE = "$40000"
```

Class constants variable access like static properties.

classname::constant_variable_name

The value for constant variable must be a constant expression, not a variable, a property, a result of a mathematical operation, or a function call.

Note: class constants cannot be made private or protected. They always publicly visible. For good practice to use all-uppercase letters for class constant names (e.g. PRICE, COUNTRY).

Example:

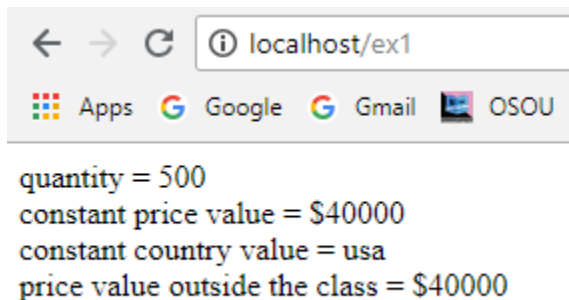
<?php

```
class Car{
    const PRICE = "$40000";
    const COUNTRY = "usa" ;
    public $quantity = 300 ;

    public function show(){
        $this->quantity = 500;
        echo "quantity = ".$this->quantity;
        echo "<br/> constant price value = ".Car::PRICE;
        echo "<br/> constant country value = ".Car::COUNTRY;
    }
}

$obj = new Car();
$obj->show();
echo "<br/>price value outside the class = ".Car::PRICE;
?>
```

OUTPUT:



1.6 Constructor and Destructor in PHP

1.6.1 Constructor in PHP

A PHP constructor is a 'special' member function whose task is to initialize the objects of its class.

Constructor is nothing but a function defined in your php class. Constructor function automatically called when you will create object of the class.

In php4 we can create constructor by creating function with same name of your class. But from php5 you can also create constructor by defining magic function **__construct**.

Syntax:

```
public function __construct()
{
    ... code goes here... }

```

Example:

Constructor in PHP4 (will work in PHP5 too)

```
class interestCalculator
{
    var $rate;
    var $duration;
    var $capital;
    //Constructor of the class
    function interestCalculator()
    {
        $this->rate = 3;
        $this->duration = 4;
    }
}
```

constructor in PHP5

```
class interestCalculator
{
    public $rate;
    public $duration;
    public $capital;
    //Constructor of the class
    public function __construct() // __ is (double Underscore)
    {
        $this->rate = 3;
        $this->duration = 4;
    }
}
```

Example:

<?php

```
class interestCalculator
{
    public $rate;
    public $duration;
    public $capital;
    //Constructor of the class
    public function __construct() // __ is (double Underscore)
    {
        $this->rate = 3;
        $this->duration = 4;
        echo "<br/> interestCalculator class object is created ";
    }
}
```

```
$obj = new interestCalculator ();  
?>
```

OUTPUT:

interestCalculator class object is created

1.6.2 Destructor in PHP

The PHP destructor is used to destroy the objects. Like Java, PHP has automatic garbage collection. A destructor function cleans up any resources allocated to an object once the object is destroyed.

Unless otherwise explicitly carried out, objects are automatically destroyed at the end of the script they were created in.

In some cases, it might sometimes be useful to specifically carry out a task when an object is destroyed, say to close a database connection.

Accordingly, the destructor is a magic method that is automatically called when an object is destroyed, e.g. at the end of the script.

Syntax:

```
public function __destruct()  
{  
    ... code written here...  
}
```

Example:

```
<?php  
class interestCalculator  
{  
    public $rate;  
    public $duration;  
    public $capital;  
    //Constructor of the class  
    public function __construct() // __ is (double Underscore)  
    {  
        $this->rate = 3;  
        $this->duration = 4;  
        echo "<br/> interestCalculator class object is created ";  
    }  
    public function __destruct() { // destroy the safari object  
        echo "<br/> destroy the object";  
    }  
}  
$obj = new interestCalculator ();  
?>
```

OUTPUT:

interestCalculator class object is created
destroy the object

A destructor is called automatically when a script ends. However, to explicitly trigger the destructor, you can destroy the object using the PHP **unset()** function, as follow:

Example:

```
<?php
```

```
class interestCalculator
{
    public $rate;
    public $duration;
    public $capital;
    //Constructor of the class
    public function __construct() // __ is (double Underscore)
    {
        $this->rate = 3;
        $this->duration = 4;
        echo "<br/> interestCalculator class object is created ";
    }
    public function __destruct() { // destroy the safari object
        echo "<br/> destroy the object";
    }
}
$obj = new interestCalculator ();
unset($obj);
?>
```

OUTPUT:

interestCalculator class object is created
destroy the object

To delete an object before the end of a script the keyword **unset()** is used with the object's variable name within its parentheses

Syntax:

```
unset($myObject);
```

1.7 Magic Methods in PHP

Magic methods in php are some predefined function by php compiler which executes on some event.

Magic methods starts with prefix __, for example __call, __get, __set. There are verous magic methods in php.

Here we will discuss some of the most comman magic methods of php which will be used in object oriented programming.

List of Magic methods in PHP

| | |
|--------------|---|
| __construct | This magic methods is called when someone create object of your class. Usually this is used for creating constructor in php5. |
| __destruct | This magic method is called when object of your class is unset. This is just opposite of __construct. |
| __get | This method called when your object attempt to read property or variable of the class which is inaccessible or unavailable. |
| __set | This method called when object of your class attempts to set value of the property which is really inaccessible or unavailable in your class. |
| __isset | This magic methods trigger when isset() function is applied on any property of the class which is inaccessible or unavailable. |
| __unset | __unset is something opposite of is set method. This method triggers when unset() function called on inaccessible or unavailable property of the class. |
| __call | __call magic method trigger when you are attempting to call method or function of the class which is either inaccessible or unavailable. |
| __callstatic | __callstatic execute when inaccessible or unavailable method is in static context. |
| __sleep | __sleep methods trigger when you are going to serialize your class object. |
| __wakeup | __wakeup executes when you are un serializing any class object. |
| __toString | __toString executes when you are using echo on your object. |
| __invoke | __invoke called when you are using object of your class as function |

1.8 Inheritance in PHP

PHP inheritance is one of the fundamental mechanisms for code reuse in OOPs.

By implementing inheritance you can inherit (or get) all properties and methods of one class to another class. The class who inherit feature of another class known as child class. The class which is being inherited is known as parent class. With the help of inheritance we can increase re-usability of code.

Real world example, child inherits characteristics of their parent. Same is here in oop. One class is inheriting characteristics of another class.

When one class is derived from another class this is called an inheritance. The new class (also called subclass, derived class, child class) can inherit members from the old class (also called superclass, parent class). The new class can also add additional properties and methods.

Classes can inherit the methods and properties of another class using the **extends** keyword.

To implementing inheritance in php we need at least 2 classes. One will be parent class and other will be child class. In child class you can inherit all properties and methods (protected and public only) from parent class.

Syntax:

```
class Demo
{
    $model = null ;
}
class Demo1 extends Demo{
    $price = "$4000"; // $model property inherit from Cars class
}
```

Example:

```
<?php
class Super1 {
    private $value;
    function Super1(){
        $this->value=100;
    }
    public function getvalue()
    {
        echo "I am in Super class with value..". $this -> value;
    }
}
```

```

class Child extends Super1 {
    private $value1 = 200;
    public function getChild()
    {
        $this -> getValue();
        echo " <br> Inside Child class with value ". $this -> value1;
    }
}
$obj = new Child();
$obj -> getChild();
?>

```

OUTPUT:

```

i am in Super class with value..100
Inside Child class with value 200

```

Example:

<?php

```

class Person {
    public $name ;
    public function speak() {
        echo "Hi, my name is $this->name, and I'm a Male";
    }
}
class Teacher extends Person {
    public $age ;
    public function speak() {
        echo "Hello, my name is $this->name, and <br>I'm a Teacher!";
        echo "<br>My age is $this->age !!";
    }
}
$obj = new Teacher;
$obj ->name = "aseem";
$obj -> age = "33";
$obj ->speak();
?>

```

OUTPUT:

```

Hello, my name is aseem, and
I'm a Teacher!
My age is 33 !!

```

The new Teacher class has extended the Person class, and in this case has also overridden the speak() method of the parent.

1.8.1 Multilevel and Multiple inheritance in PHP

In PHP multilevel inheritance is possible but multiple inheritance is not possible. In simplified terms in PHP child class cannot inherit more than one parent class. But hierarchical inheritance is possible in PHP.

Hierarchical means Parent inherit property of grandparent class. Grandchild inherit property of parent class. So in multilevel inheritance child can get some property of from grandparent class also.

1.8.1.1 Multilevel inheritance

In multilevel, multiple classes are involved in inheritance, but one class extends only one. The lowermost subclass can make use of all its super classes' members. Multilevel inheritance is an indirect way of implementing multiple inheritance.

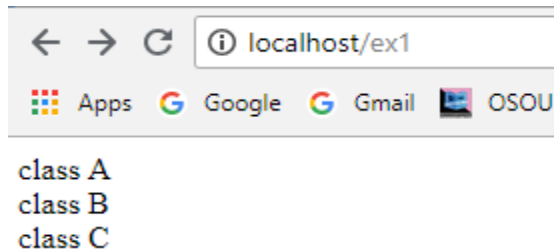
Example:

```
<?php
class a
{
    public function function_a(){
        echo "class A<br>";
    }
}

class b extends a
{
    public function function_b(){
        $this->function_a();
        echo "class B<br>";
    }
}

class c extends b
{
    public function function_c(){
        $this->function_b();
        echo "class C<br>";
    }
}
$c =new c();
$c->function_c(); ?>
```

OUTPUT:



```
class A
class B
class C
```

1.8.1.2 Multiple inheritance

When one child class acquires the property from one than one class is called multiple inheritance. In simplified terms in PHP child class cannot inherit more than one parent class.

```
class A
{
    //Your class body
}
class B
{
    //Your class body
}
class C extends A B
{
    //your class body
}
```

Above program code will not work in php because php does not supports multiple inheritance.

To allow this feature, we can use “**interfaces**” or use "**Traits**" in PHP.

1.8.1.3 Traits in PHP (replacement of multiple Inheritance)

PHP 5.4.0 was coming with one amazing feature called trait. It provide a mechanism which will allow us to implement multiple inheritance in PHP.

The Trait behaves like an abstract class. It cannot be instantiated on its own. Traits can be initialized with the keyword **trait**.

Syntax:

```
trait traitName
{
    public function functionName($args)
    {
        // statement Code
    }
}
```

Example:

```
<?php
trait hello
{
    function sayHello() { echo "Hello"; } // method1
}
trait world
{
    function sayWorld() { echo "World!!!"; } // method2
}
class HelloWorld
{
    // now using more than one trait
    use hello, world;
}
$obj = new HelloWorld();
$obj->sayHello(); // Print : sayHello
$obj->sayWorld(); // Print : sayWorld
?>
```

OUTPUT:

Hello World!!!

1.9 Interface

Interfaces provide a way of implementing multiple inheritance (not directly available in PHP) through the use of the keywords `interface` and `implements`. By implementing interface we are forcing any class to must declaring some specific set of methods in oop.

We can create an interface in PHP using `interface` keyword. Rest of the things are typically identical to classes. Following is a very small example of an interface in PHP.

```
interface abc
{
    public function xyz($b);
}
```

So in above code, we are creating an interface with name `abc`. Interface `abc` has function `xyz`. Whenever we will implement the `abc` interface in our class then we have to create a method with the name `xyz`. If we will not create function `xyz` then it will throw an error.

You can implement your interface in your class using `implements` keyword. Let us implement our interface `abc` in our class

```
class test implements abc
{
    public function xyz($b)
    { //your function body
    } }
```

Example:**<?php**

```
interface person {
    function setName($myName);
    function getName();
    function setAge($myAge);
    function getAge();
}
interface scientist{
    function measure();
    function writePaper();
}
class Geek implements person, scientist {
    public $name;
    public $age;
    public function setName($myName){
        $this->name = $myName;
    }
    public function getName(){
        return $this->name ;
    }
    public function setAge($myAge){
        $this->age = $myAge;
    }
    public function getAge(){
        return $this->age ;
    }
    public function measure(){
        echo "$this->name has just made a measurement!<br>";
    }
    public function writePaper(){
        echo "$this->name has written a paper!<br>";
    }
}
$steve = new Geek();
$steve->setName("Stephen Hawking");
$steve->setAge(71);
echo "The guy who invented big bangs: ". $steve->getName() . " is now
" . $steve->getAge() . " years old!<br>";
$steve->measure();
$steve->writePaper();
```

?>

1.10 Abstract class

Abstract classes are those classes which cannot be directly initialized. Or in other word we can say that you cannot create object of abstract classes. Abstract classes always created for inheritance purpose.

Usually, an abstract class is also known as base class. We call it base class because the abstract class is not the class which is available directly for creating an object. It can only act as the parent class of any normal class. You can use an abstract class in the class hierarchy. Mean one abstract class can inherit another abstract class also.

You can create abstract classes and methods in PHP using abstract keyword.

Syntax:

```
abstract class Cars{
    public abstract function funName( $c );
}
```

Abstract method: - If a method has no definition then it is called abstract method.

Points to be Remember:

- We cannot create object of abstract class
- Abstract class and method cannot be final
- Abstract method cannot be private, because private method are never inheritance.
- If a class extends an abstract class then it must define all the abstract methods if any of the abstract method remain undefined in the subclass then the subclass must we declared abstract otherwise throw an error.
- If the abstract method is declared as protected, the function implementation must be defined as either protected or public, but not private (always less or same restricted).

Example:

```
<?php
abstract class Cars{
    public abstract function set_color( $c );
    abstract function set_price( $p );
    public abstract function show();
    function display(){
        echo "<br/>i am not a abstract method";
    }
}
```

```
class Audi extends Cars{
private $color = null;
```



```

private $price = null;
public function set_color( $c )
    {
        $this->color = $c;
    }
public function set_price( $p )
    {
        $this->price = $p;
    }
public function show()
    {
        echo "<br/> car color = ".$this->color;
        echo "<br/> car price = ".$this->price;
    }
}

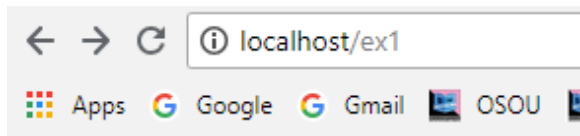
```

```

$audi = new Audi();
$audi->set_color( "Red" );
$audi->set_price( "$55999" );
$audi->show();
$audi->display();
?>

```

OUTPUT:



```

car color = Green
car price = $44993
i am not a abstract method

```

1.11 Differences between abstract class and interface in PHP

Following are some main difference between abstract classes and interface in PHP

1. In abstract classes, this is not necessary that every method should be abstract. But in interface every method is abstract.
2. Multiple and multilevel both type of inheritance is possible in the interface. But single and multilevel inheritance is possible in abstract classes.
3. The method of PHP interface must be public only. A method in an abstract class in PHP could be public or protected both.
4. In an abstract class, you can define as well as declare methods. But in the interface, you can only define your methods.

1.12 Final class and method in PHP

A php class can be declared final to indicate that it cannot be extended; that is, one cannot declare subclasses of a final class. A final class's behavior cannot be changed by extending the class. A final class must be complete.

The class properties cannot be declared final, only non-abstract classes and methods may be declared as final.

Syntax:

```
final class className{ }

final public function functionName(){}
```

Important Point:

- used to declare that a method or class cannot be overridden by a subclass
- cannot be applied to properties
- If the class itself is being defined final then it cannot be extended
- Means of stopping other programmers using the code in unplanned ways

Example:

<?php

```
final class Person {
    public $name;
    public function speak() {
        echo $this->name;
    }
}

class Teacher extends Person {
    public function speak() {
        echo "$this->name is a Teacher who works with OSOU!!!";
    }
}

$geek = new Teacher;
$geek->name = "OSOU";
echo $geek->speak() ;?>
```

OUTPUT:

Fatal error: Class Teacher may not inherit from final class (Person) in C:\AppServ\www\ex1.php on line 12

Example:

<?php

```
class Person {
    public $name;
    final public function speak() {
        echo $this->name;
    }
}
```

```

    }
}
class Teacher extends Person {
    public function speak() {
        echo "$this->name is a Teacher who works with OSOU!!!";
    }
}
$geek = new Teacher;
$geek->name = "OSOU";
echo $geek->speak() ;
?>

```

OUTPUT:

Fatal error: Cannot override final method Person::speak() in
C:\AppServ\www\ex1.php on line 12

1.13 Polymorphism

Polymorphism definition is that Poly means many and morphos means forms. It describes the feature of languages that allows the same word or symbol to be interpreted correctly in different situations based on the context. Object-oriented programs are written so that the methods having the same name works differently in different context.

Polymorphism can be achieved using method overloading and method overriding.

Overloading: Same method name with different signature, since PHP doesn't support method overloading concept

Overriding: When same methods defined in parents and child class with same signature i.e. known as method overriding. A super class and sub class both have methods with same name this is called method overriding.

1.13.1 Method Overriding

Overriding in php is very easy. As we know that overriding is process of modifying the inherited method. So in case of inheritance you only need to create method with same name in your child class which you want to override. Following is example of overriding of method in php.

Example:

```

<?php
class ParentClass {
public function display() {
    echo "Hello i am inside ParentClass display method...";
}
}
class ChildClass extends ParentClass {
    public function display() {
        echo "Display method of parentClass has been overridden<br/>";
    }
}

```

```
        echo "Hello i am inside ChildClass display method...";
    } }
```

```
$myChild = new ChildClass;
echo $myChild->display();
?>
```

OUTPUT:

Display method of parentClass has been overridden
Hello i am inside ChildClass display method...

Another Example:

```
<?php
```

```
class testParent
{
    public function f1()
    {
        echo "f1 method of testParent class: Odisha";
    }
    public function f2()
    {
        echo "f2 method of testParent class: Sambalpur";
    }
}
class testChild extends testParent
{
    function f2($a) //overriding function f2
    {
        echo "<br/>f2 method of testChild class: $a";
    }
}
$a = new testChild();
$a->f1();
$a->f2("OSOU");
?>
```

OUTPUT:

f1 method of testParent class: Odisha
f2 method of testChild class: OSOU

Explanation:

In the above example we have overloaded the function f2() of class testParent with the same function name in testChild class. So when we call f2() method using object of testChild class it will execute the f2() method of testChild class, whereas when we call f1() method using the same object it will execute f1() method of testParent class.

1.14 Let us sum up

The object-oriented programming style in PHP is a great way to build modular, reusable code, letting you create large applications that are relatively easy to maintain.

The major concept of the object oriented programming in PHP is introduced from version 5.

In this unit we learnt about OOPS features, how to implement class and object concept in PHP programming, Access Modifiers, constructor, destructor and many more.

We also learnt Magic Methods in PHP, inheritance, replacement of Multiple Inheritance i.e. interface, Abstract class and how it is different from interface. We got an idea about how to restrict classes from inheritance using final keyword and polymorphism.

In whole series I will use abbreviation OOP for Object Oriented Programming.

1.15 References:

<https://www.techflirt.com/tutorials/oop-in-php/index.html>

<https://intphp.tech-academy.co.uk/>

<http://www.w3webtutorial.com/php/php-oops-classes.php>

1.16 Model Questions

Q.No. 01: What is OOPS Concepts? what are the pillars of OOPS?

Q.No. 02: Write a program in PHP using class and object.

Q.No. 03: What are the advantages of OOPS Concept?

Q.No. 04: What is Access modifiers? What are its types?

Q.No. 05: What is constructor? How to implement constructor in PHP

Q.No. 06: What is static keyword in PHP? How to access static class and member in PHP?

Q.No. 07: Differentiate between interface and Abstract class.

Q.No. 08: What is Traits in PHP?

Q.No. 09: What is multilevel inheritance? Write a PHP program to implement multilevel inheritance.

Q.No. 10: What is Polymorphism? What is function overloading and overriding in PHP?

UNIT-02

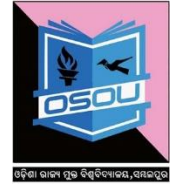
File management and Exception handling

UNIT STRUCTURE

| | |
|---|----|
| 2.1 Introduction | 01 |
| 2.2 What is a file | 01 |
| 2.3 File Formats support by PHP | 01 |
| 2.4 File Operations | 01 |
| 2.4.1 Opening a File | 01 |
| 2.4.2 Closing a File | 02 |
| 2.4.3 Read and Write Files | 03 |
| 2.4.3.1 Reading a File Character by Character | 04 |
| 2.4.3.2 Reading a File Line by Line | 04 |
| 2.4.3.3 Writing a File | 05 |
| 2.4.3.4 Append content to the file in PHP | 06 |
| 2.4.3.5 Copy a file in PHP | 08 |
| 2.4.3.6 Rename a file in PHP | 08 |
| 2.4.3.7 Delete a file in PHP | 09 |
| 2.5 Permission in PHP | 10 |
| 2.5.1 Check File permission | 10 |
| 2.5.2 Changing File permission | 11 |
| 2.6 Error handling in PHP | 12 |
| 2.7 Exception Handling | 16 |
| 2.8 Let Us Sum Up | 20 |
| 2.9 References | 20 |
| 2.10 Model Questions | 20 |

UNIT-02

File management and Exception handling



2.1 introduction

Working with files is an important part of any programming language, and PHP is no different. Whatever your reasons are for wanting to manipulate files, PHP will happily accommodate them through the use of a handful of functions. We will explore different ways of interacting with data files. You'll learn different methods and use cases for reading and writing files in a directory.

2.2 What is a File?

A file is simply a resource for storing information on a computer. Files are stored in directories on a hard drive, and because they retain their data after the computer is shut down, they are a persistent storage mechanism, instead of temporary storage such as RAM. A file can contain any kind of data.

Files are normally used to store data and information like;

- Configuration settings of a program
- Simple data such as contact names against the phone numbers.
- Images, Pictures, Photos, etc.

2.3 File Formats Support by PHP

PHP supports a variety of file formats that include;

text file (.txt file)

log file (.log)

custom_extension file (.abc file)

comma-separated values file (.csv file)

image files (.gif, .jpg, .bmp, .png, .swf, .tiff etc.)

2.4 File Operations

PHP handles file using a set of in-built functions. Some of the functions are, fopen() and fclose(),feof(),fgetc and many more.

Some of the basic operations for file handling are given below.

- Opening file
- Working with file read, write and append
- Closing file

2.4.1 Opening a File

The fopen() function is used to opens a file and returns a file handle associated with the file.

If the fopen() function is unable to open the specified file, it returns FALSE(0).

Syntax:

fopen(filename, mode)

filename: Specifies the name of the file you want to open.

mode: Specifies the mode, how the file is to be used.

Example:

```
<html>
  <head> <title>opening files </title></head>
  <body>
    <?php
      $open = fopen( "http://localhost:80/data.html", "r" );
      $xx = fopen( "data.txt", "r" );
    ?>
  </body>
</html>
```

The file may be opened in one of the following modes:

Mode	Description
r	Open for reading only. the file pointer at the beginning of the file
r+	Read/Write. the file pointer at the beginning of the file
w	file Open for writing only. w mode use to creates a new file if it doesn't exist
w+	Read/Write. w+ mode use to creates a new file if it doesn't exist
a	Append. Opens and writes to the end of the file
a+	Read/Append. Preserves file content by writing to the end of the file
x	Write only. Creates a new file. Returns FALSE and an error if file already exists
x+	Write/Read. Creates a new file. Returns FALSE and an error if file already exists

2.4.2 Closing a File

After performing all file operations of PHP, we need to close it by using the fclose() function. The fclose() function return true on success or false on failure.

Syntax:

fclose(\$filePointer);

Example:

```
<?php
$file = fopen( "fileName.txt", 'w+')
fclose( $file ); ?>
```


2.4.3 Read and Write Files

Once a file is opened using `fopen()` function it can be read using `fread()` function. This function must have two arguments. They are file pointer and the length of the file in bytes.

Steps to read a file with PHP.

- Open a file using `fopen()` function.
- Get the file's length using `filesize()` function.
- Read the file's content using `fread()` function.
- Close the file with `fclose()` function.

Syntax:

```
$char = fread( $filePointer, fileSize );
```

Example:

```
<html>
  <head><title>File Reading using PHP</title> </head>

  <body>
    <?php
      $file = fopen("filename.txt", "r" );

      if( $file == false ) {
        echo ( "Error in opening file" );
        exit();
      }

      $filesize = filesize("filename.txt");
      $filetext = fread( $file, $filesize );
      fclose( $file );

      echo ( "File size : $filesize bytes" );
      echo ( "<pre>$filetext</pre>" );
    ?>
  </body>
</html>
```

Check End-of-file

The php `feof()` method checks if the 'end-of-file' (EOF) has been reached.

The `feof()` function return true when the file pointer has reached the end of the file and returns false otherwise.

Note: `feof()` is useful with `fread()` or `fgetc()` in a while loop when you don't know how long the file is: **`if(feof($file)) echo "pointer reach end of file";`**

2.4.3.1 Reading a File Character by Character

The php `getc()` function read a single character from a file. This method returns just one character from the file it points to, it returns false when file pointer reached the end of the file.

Syntax:

```
$char = fgetc( $file );
```

Note: `fgetc()` function is slow when working with large files.

Note: after a call to this method the file pointer moves to the next character.

Example:

```
<html>
  <head><title>File Reading using PHP</title> </head>

  <body>
    <?php
      $file = fopen("/data.html", "r");
      while( !feof($file) )
      {
          $char = fgetc( $file );
          echo "$char <br/>";
      }
      fclose( $file );
    ?>
  </body>
</html>
```

2.4.3.2 Reading a File Line by Line

The php `fgets()` function read a single line from a file.

If there is no more data to read in the file pointer, then FALSE is returned.

Example:

```
<?php
$handle = fopen("data.text", "r");
if($handle){
    while( ($buffer = fgets($handle, 4090)) !== false){
        echo $buffer;
    }
    if(!feof($handle)){
        echo "error: unexpected fgets() fail \n";
    }
    fclose($handle);
}??>
```

2.4.3.3 Writing a File

The `fwrite()` function used to write binary data to a file.

This function returns the number of bytes written, or `FALSE` on failure.

Syntax:

`fwrite(file, text, length)`

file: Required. Specifies the open file to write to

text: Required. Specifies the string to write to the open file

length: Optional. Specifies the maximum number of bytes to write.

To perform the write operation, we have two choices to select the mode of the file to be opened. These are,

Mode of Operation	File Mode	File Pointer Position
W	write-only	Start of the file content
W+	read-write	Start of the file content

By using these modes, the entire file content will be cleared and the pointer will focus the start position of the file content. This method is used to change the existing file content, completely.

Example:

```
<?php
$filename = "newfile.txt";
$file = fopen( $filename, "w" );

if( $file == false ) {
    echo ( "Error in opening new file" );
    exit();
}
fwrite( $file, "This is a simple test\n" );
fclose( $file );
?>
<html>
  <head><title>Writing a file using PHP</title></head>
  <body>
    <?php
      $filename = "newfile.txt";
      $file = fopen( $filename, "r" );
```

```

if( $file == false ) {
    echo ( "Error in opening file" );
    exit();
}
$filesize = filesize( $filename );
$filetext = fread( $file, $filesize );
fclose( $file );

    echo ( "File size : $filesize bytes<br/>" );
    echo ( "$filetext <br/>" );
    echo("file name: $filename <br/>");
    ?>
</body>
</html>

```

OUTPUT:

File size: 23 bytes
This is a simple test
file name: newfile.txt

2.4.3.4 Append content to the file in PHP

In append operation, the content of the existing file will not be erased. we can add/append new content with the existing content of the file. Here, file pointer will point end of the file. To append the content in a existing file we need to change the mode to append i.e. a.

Mode of Operation	File Mode	File Pointer Position
a	write-only	End of the file content
A+	read-write	End of the file content

Syntax:

\$filePointer = fopen("hello.txt","a");

Note: In both, write and append mode file will open if exists. Otherwise, a new file will be created for the execution of file write and append.

Example:

```
<?php
$filename = "newfile.txt";
$file = fopen( $filename, "a" );

if( $file == false ) {
    echo ( "Error in opening new file" );
    exit();
}
fwrite( $file, "This is a simple test\n" );
fclose( $file );
?>
<html>
    <head><title>Writing a file using PHP</title></head>
    <body>
        <?php
            $filename = "newfile.txt";
            $file = fopen( $filename, "r" );

            if( $file == false ) {
                echo ( "Error in opening file" );
                exit();
            }
            $filesize = filesize( $filename );
            $filetext = fread( $file, $filesize );
            fclose( $file );

            echo ( "File size : $filesize bytes<br/>" );
            echo ( "$filetext <br/>" );
            echo("file name: $filename <br/>");
        ?>
    </body>
</html>
```

OUTPUT:

```
File size: 51 bytes
This is a simple test
This is a simple test
file name: newfile.txt
```

2.4.3.5 Copy a file in PHP

To copy a file, we need to use copy() function. First, we need to make sure which file to copy by passing as first parameter of the copy() function. Second, you need to specify the file name to copy the file to.

The copy() function returns true if the file was copied successfully, otherwise it returns false.

Syntax:

```
copy( source, dest);
```

Suppose we have to file. One text.html and second is demo.html which locates in the same directory as the script.

Example:

```
<?php
```

```
    $oldf = 'newfile.html';
```

```
    $newf = 'demo.html';
```

```
    if(copy($oldf,$newf))
```

```
        echo 'The file was copied successfully';
```

```
    else
```

```
        echo 'An error occurred during copying the file';
```

```
?>
```

2.4.3.6 Rename a file in PHP

To rename a file, we use the rename() function in PHP. The rename() function renames a file or directory.

Syntax:

```
rename( oldname, newname)
```

Example:

```
<?php
```

```
$fname = 'osou.txt';
```

```
$newfname = 'osoul.bak';
```

```
if(rename($fname,$newfname)){
```

```
    echo sprintf("%s was renamed to %s", $fname, $newfname);
```

```
}else{
```

```
    echo 'An error occurred during renaming the file';
```

```
} ?>
```

This function is also used to move a file to a different directory.

Example:

```
<?php
```

```
$fname = 'osou.txt';  
$newfname = 'C:\Users\osou-18\Desktop\osou1.bak';
```

```
if(rename($fname,$newfname)){  
    echo sprintf("%s was renamed to %s",$fname,$newfname);  
}else{  
    echo 'An error occurred during renaming the file';  
} ?>
```

2.4.3.7 Delete a file in PHP

To delete a file, we can use unlink() function in PHP. The function returns true on successful deletion of file or returns false on failure.

Syntax:

```
unlink( filename );
```

Example:

```
<?php
```

```
$fname = 'C:\Users\osou-18\Desktop\aseem.bak';  
if(unlink($fname)){  
    echo sprintf("The file %s deleted successfully",$fname);  
}  
else{  
    echo sprintf("An error occurred deleting the file %s",$fname);  
}  
?>
```

Note:

copy(), rename() and unlink() functions shows warning-level errors if the file cannot be found therefore it is good practice to check the file exists using the file_exists() function before copying, renaming or deleting it.

2.5 File Permission in PHP

File permissions indicate what we can do with a specific file in the computer i.e., reading, writing or executing the file.

When admin upload files to the webserver by FTP, those files are saved on the server with some specific access permissions. Typically, everyone will be able to "read" the files and subdirectories, that is: view them through the webserver, but only admin will be able to "write" and modify to these files/subdirectories. This type of access control to a file is called File Permission.

2.5.1 Check File permission

PHP has pre-defined functions for checking and changing the file permissions. i.e. `is_readable()`, `is_writable()` and `is_executable()`.

- **is_readable()** returns true if programmer has the permission to read the file, otherwise returns false.
- **is_writable()** returns true if programmer has the permission to write the file, otherwise returns false.
- **is_executable()** returns true if programmer has the permission to execute the file, otherwise returns false.

Example: (How to check Permission of a file)

```
<?php
$name = 'newfile.html';
if(is_readable($fn)){
    echo 'File is readable<br/>';
}
else{
    echo 'File is not readable<br/>';
}
if(is_writable($fn)){
    echo 'File is writable<br/>';}
else{
    echo 'File is not writable<br/>';
}
if(is_executable($fn)){
    echo 'File is executable<br/>';
}
else{
    echo 'File is not executable<br/>';
} ?>
```

OUTPUT:

```
File is readable
File is writable
File is not executable
```


2.5.2 Changing File permission

In PHP, file permissions, or mode can be changed by using `chmod()` function.

Syntax:

`chmod($fname, mode);`

1st, we need to pass the name of the file that we want to set permission. 2nd, we pass the preferred permission.

The `chmod()` function returns true if the permission was successfully assigned otherwise it returns false.

A file permission is represented by an octal number that contains three digits:

- The first digit specifies the owner file permission.
- The second digit specifies the owner group file permission.
- The third digit specifies for everyone's file permission.

Value	Permission
0	cannot read, write or execute
1	can only execute
2	can only write
3	can write and execute
4	can only read
5	can read and execute
6	can read and write
7	can read, write and execute

Example:

```
<?php
```

```
if(chmod("newfile.html", 0600))
    echo "Read and write for owner, nothing for everybody else<br/>";
if(chmod("newfile.html", 0644))
    echo "Read and write for owner, read for everybody else<br/>";
if(chmod("newfile.html", 0755))
    echo "Everything for owner, read and execute for others<br/>";
if(chmod("newfile.html", 0750))
    echo "Everything for owner, read and execute for others<br/>";
?>
```

NOTE: 0 before mode value (664) to request PHP to interpret it as an octal number.

2.6 Error handling in PHP

Errors are the most common event a developer or programmer faces when programming. Errors can be categorized as syntactical, run-time, or logical.

- **syntax error:** missing the semicolon at the end of a statement.
- **run-time error:** trying to connect to a database when the server is down
- **logic error:** providing incorrect data to a variable

Error handling in PHP is a mechanism to improve application's security, appearance, and debugging capabilities.

Some error handling methods are:

- die() statement
- Custom error handler
- Error reporting

2.6.1 Error Handling Using the die() function

When we open a file using 'fopen' function to read and write operation, we use the code:

```
<?php $rw = fopen("fileName.txt", "r+"); ?>
```

If the file does not exist 'fopen' function throw a run-time error look like:

Warning: fopen(fileName.txt) [function.fopen]:failed to open stream: No such file or directory in C:\AppServ\www\data.php on line 1

So to solve this problem we can use die() method.

Example:

```
<?php
if( !file_exists("fileName.txt")){
    die("User.Txt File Not Found );
}
else{
    $rw = fopen("fileName.txt", "r+");
}
?>
```

If the file does not exist the die function display a simple message: **"User.Txt File Not Found"**

2.6.2 Custom Error Handling

It is possible to override PHP's default mechanism for handling errors. This option gives the programmer/ developer full control over what actions to take when an error is raised.

In this process we create a special function that can be called when an error occurs in PHP.

This function must be able to handle a minimum of two parameters (error_level, error_message) but can accept up to five parameters.

Syntax:

error_function(error_level, error_message, error_file, error_line, error_context)

Parameter	Description
error_level	Required. Specifies the error reporting level for the error. Must be a value number.
error_message	Required. Specifies the error message.
error_file	Optional. Specifies the name of the file in which the error occurred.
error_line	Optional. Specifies the line number at which the error occurred
error_context	Optional. Specifies an array containing every variable, their values, in use when the error occurred

Example:

```
<?php
function error_Handler( $error_level, $error_message, $error_line)
{
    echo "Error : [" . $error_level . "] Error Message :". $error_message;
    echo "<br/> Line Number At Which Error Occured :". $error_line;
}
?>
```

Example:

```
<?php
function error_Handler( $error_level, $error_message, $error_line)
{
    echo "<br/><b>Error :</b> ". $error_level.
    "<br/><b>Error Message :</b>". $error_message.
    "<br/><b>Error line:</b>". $error_line;
}

set_error_handler("error_Handler");
$data= 30/0; // at this point run-time error will trigger
echo( $data );
?>
```

OUTPUT:

Error : 2

Error Message :Division by zero

Error line:C:\AppServ\www\ex1.php

2.6.3 Trigger Errors

Programmer can trigger an error of a specific level using `trigger_error()` function.

Syntax:

```
trigger_error( error_msg, error_level)
```

error_msg: The designated error message. It's limited to 1024bytes in length.

error_type: Possible error types:-

E_USER_ERROR - Fatal user-generated run-time error. Execution of the script is halted. Value is 256

E_USER_WARNING - Non-fatal user-generated run time error. Script is not halted. Value is 512

E_USER_NOTICE - Default. User-generated run time notice. Value is 1024

This function is useful when you need to generate a particular response to an exception at runtime.

Example:

```
<?php  
function error_Handler( $error_level, $error_message, $error_line)  
{  
echo "<br/>Error : [". $error_level ."] Error Message:". $error_message;  
}  
set_error_handler("error_Handler");  
$test = 2;  
if( $test>1 ){  
    trigger_error("value must be less than one", E_USER_ERROR);  
}  
?>
```

OUTPUT: Error: [256] Error Message: value must be less than one

Example:

If we replace the above highlighted line with below line

```
trigger_error("value must be less than one", E_USER_WARNING);
```

Then Output will be: Error: [512] Error Message: value must be less than one

Example:

Similarly If we replace the above highlighted line with below line

```
trigger_error("value must be less than one", E_USER_NOTICE);
```

Then Output will be

Error: [1024] Error Message: value must be less than one

2.6.4 Error Logging

The `error_log()` function sends an error to a specified file or a remote destination.

By default, PHP sends an error log to the server logging system or a file, depending on how the `error_log` configuration is set in the `php.ini` file.

Syntax:

```
error_log( msg, type, destination, headers);
```

Example:

```
<?php
```

```
function error_Handler( $error_level, $error_message, $error_line)
{
    echo "<br/>Error: [". $error_level ."] Error Message:". $error_message;
    error_log("Error :[$error_level] Error Message :
    {$error_message}",1,"demo@osou.ac.in", "From:
    ak.patel@osou.ac.in");
}
set_error_handler("error_Handler");
$test = 2;
if( $test>1){
    trigger_error("value must be less than one", E_USER_WARNING);
}
?>
```

OUTPUT:

The output of the code above should be like this:

Error: [256] Error Message: value must be less than one

The mail received from the code above looks like:

Error: [256] Error Message: value must be less than one

2.7 Exception Handling

Exception handling mechanism can change the normal flow of the code execution, if the specified error occurs.

PHP exception handling mechanism to improve your application's security, appearance, debugging capabilities, and make more robust.

2.7.1 Exception Class in PHP

Exception is the base class for all exceptions.

The exception class offers six different methods to access information about what caused the problem look methods:

Method	Description
getMessage()	Returns the exception message
getCode()	Returns the exception error code
getLine()	Returns the line number in which the exception occurred
getFile()	Returns the file in which the exception occurred
getTrace()	Gets the stack trace
getTraceAsString()	Gets the stack trace as a string

2.7.2 Try . . . Catch and throw

Exception handling is carried out by the use of the keywords throw, try, catch.

Try: - The code that may throw an exception is placed within the try block. Each try must have at least one corresponding catch or finally block (or one of both).

Catch: - The code to handle the exception is placed within the catch block. Multiple catch blocks can be used to catch different classes of exceptions.

Throw: - Trigger an exception. Thrown object must be an instance of the Exception class or a subclass of Exception.

There are three forms of the 'try' statement

<pre>try{ //throw__ statement } catch(Exception e){ // catch__statement } catch{ }</pre>	<pre>try { //throw__ statement } catch(Exception e){ // catch__statement }</pre>	<pre>try { //try __ statements } finally { //finally__statement }</pre>
--	--	---

Note: - If an exception is not caught, a PHP Fatal Error will be issued with an "Uncaught Exception" message. **The finally block only work in PHP 5.5 and later versions.**

Example:

<?php

```
function exc($value){
if($value<=0){
    throw new Exception("the value is less than zero");
    }
}
try{
    exc(-23);
    echo "statement execute only when the exc method not throw the exception";
}
catch( Exception $e){
    echo "exception handler correct exception";
    echo "<br/>error message:- ". $e->getMessage();
}
?>
```

OUTPUT:

```
exception handler correct exception
error message :- the value is less than zero
```

Explanation:

The exc() method is called in a 'try' block, first the exc() method check the value, if value is less than zero then exception is thrown. The catch block retrieves the exception and creates an object (\$e) containing the exception information. The error message access from the exception object by calling \$e->getMessage().

2.7.3 Top Level Exception Handler

The set_exception_handler() function is used to sets the default exception handler if an exception is not caught within a try/catch block.

Syntax:

```
set_exception_handler( "handler" )
```

handler is the name of the function to called when an uncaught exception occurs.

Note: This function must be defined before calling `set_exception_handler()`.

Example:

<?php

```
function handler( $exception ){  
    echo "i am exception handler";  
    echo "<br/>error message :- ". $exception->getMessage();  
}  
set_exception_handler( 'handler' );  
throw new Exception("FileNotFoundException");  
?>
```

OUTPUT:

i am exception handler
error message :- FileNotFoundException

2.7.4 User Defined Exception Handler

User or programmer can create their own custom exceptions. This allows programmer to add their own methods and properties to the exception objects, which can help to make error reporting even more rich and useful to users and developers.

Example:

<?php

```
class DivisionByZero extends Exception{  
    public function displayMessage(){  
        $value ="error message:- " . $this->getMessage();  
        return $value;  
    }  
    public function file(){  
        $v="file path:- " . $this->getFile();  
        return $v;  
    }  
}  
$var = -2 ;  
$msg = "division by zero exception";
```

```
try{  
    if($var<0){  
        throw new DivisionByZero($msg);  
    }  
}
```

```
Catch (DivisionByZero $e)  
{  
    echo $e->displayMessage();  
    echo "<br/>". $e->file();  
}  
?>
```


OUTPUT:

error message:- division by zero exception
file path:- C:\AppServ\www\ex1.php

Example:**<?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();
}

?>
```

OUTPUT:

Message: Value must be 1 or below

2.8 Let Us Sum Up

Files are stored in directories on a hard drive, and because they retain their data after the computer is shut down, they are a persistent storage mechanism, instead of temporary storage such as RAM. A file can contain any kind of data. In this unit we learnt about file, its need in storing data and different operation that can possible with File. This unit also gives idea about the process of read a file character by character and line by line, it also explain about file [permission and checking of file permission.

Also we learnt about error handling and exception handling concept in PHP.

2.9 References:

<http://www.w3webtutorial.com/php/php-read-write-file.php>

<http://www.zentut.com/php-tutorial/php-file-operations/>

<https://phpspot.com/php/php-file-handling/>

<https://websistent.com/php-include-vs-require/>

2.10 Model Questions

Q. No. 01. What is File? What are the File Formats Support by PHP?

Q. No. 02. What are the different operation that can be possible in File?

Q. No. 03. Write a program to read a file line by line.

Q. No. 04. What is file permission in PHP? How to Check File permission?

Q. No. 05. What is error handling in PHP? What are the methods to handle error in PHP?

Q. No. 06. What is Exception Handling in PHP? How to handle Exception using try, catch block?

Q. No. 07. How to create user defined exception? Explain with an example.

UNIT-03

Database connectivity in PHP

UNIT STRUCTURE

3.1 Introduction	01
3.2 Introduction to MySQL	01
3.2.1 What Can MySQL Do?	02
3.2.2 Why MySQL use with PHP	02
3.2.3 Features of MySQL	02
3.3 Communication between PHP and MySql server	03
3.3.1 Create a connect to the MySql server	03
3.3.1.1 Using mysql extension	04
3.3.1.2 Using mysqli extension (MySQL improved)	07
3.3.1.3 Using PDO extension (PHP Data Object)	07
3.3.2 Create Database and Tables in MySql	08
3.3.2.1 Create Table, Primary Keys and Auto Increment Fields	09
3.3.3 Insert Data into MySql Server	10
3.3.4 Mysql SELECT Statement	11
3.3.5 Update MySql Records	13
3.3.6 Delete MySql Records	14
3.4 Example database access from Webpage	15
3.5 Let us Sum Up	17
3.6 References	17
3.7 Model Questions	17

UNIT-03

Database connectivity in PHP

Learning Objective

By the end of this unit we will learn how to:

- Get a connection to a MySQL database from within PHP.
- Use a particular database.
- Send a query to the database.
- Parse the query results.
- Check for data errors.
- Build HTML output from data results.

3.1 Introduction

One of the most common applications of PHP is to provide a Web interface for accessing a database. The database may hold information about user postings for a forum, account and order information for a vendor, or raw data for a statistics site.

Because databases are so common, there are a special category of programs written specifically for managing databases, called database management systems. Some of the more popular commercial DBMS packages are Oracle and Microsoft SQL Server, but there are also two prominent open-source DBMS packages, MySQL and PostgreSQL.

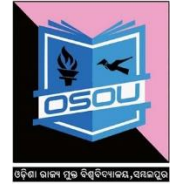
PHP is well known for its smooth database integration, especially with MySQL. It's actually quite easy to connect to a MySQL database from within PHP. Once you've established the connection, you can send SQL commands to the database and receive the results as data you can use in your PHP program. So, we'll use MySQL here, since it is easily available and used quite frequently for Web pages in conjunction with PHP.

3.2 Introduction to MySQL

MySQL is the most popular database system used with PHP. The kinds of database that can be accessed using PHP are known as relational databases. In a relational database information is stored in a number of two-dimensional structures called tables.

MySQL is a freely available open source Relational Database Management System (RDBMS) that uses Structured Query Language (SQL).

A database is a collection of tables (made up of columns and rows) that stores information. Most databases are created, updated, and read using SQL (Structured Query Language). There are few commands in SQL, which are used to perform above operation.



SQL was designed to be written a lot like the English language, which makes it very user friendly. But SQL is still extremely capable, even if it takes some thought to create more elaborate SQL statements with only the handful of available terms.

Command	Purpose
ALTER	Modifies an existing table
CREATE	Creates a database or table
DELETE	Deletes records from a table
DROP	Deletes a database or table
INSERT	Adds records to a table
SELECT	Retrieves records from a table
UPDATE	Updates records in a table

3.2.1 What Can MySQL Do?

- MySQL can create new databases
- MySQL can create new tables in a database
- MySQL can create stored procedures in a database
- MySQL can retrieve data from a database
- MySQL can insert records in a database
- MySQL can update records in a database
- MySQL can delete records from a database
- MySQL can execute queries against a database
- MySQL can set permissions on tables, procedures, and views.

3.2.2 Why MySQL use with PHP

- MySQL is an open source database system that runs on a server.
- MySQL is very fast performance, reliable, and easy to use
- MySQL supports Structured Query Language (SQL)
- MySQL manage easily
- MySQL compiles on a number of platforms.
- MySQL database can able store any kind of data types (text, images, media, audio files).

3.2.3 Features of MySQL

MySQL is a freely available open source Relational Database Management System (RDBMS) that uses Structured Query Language (SQL).

- Uses a very fast thread-based memory allocation system.
- Supports server secure layer (SSL)
- Cross-Platform support
- Full-Text indexing and searching using MyISAM engine
- Embedded database library

Security

- Privilege and password system that is very flexible and secure and that enables host-based verification.

Scalability and Limits

- MySQL support for large database that contain 50 million records.

3.3 Communication between PHP and MySQL server

When we worked with text files in previous unit, “File Management,” we saw that some functions, such as `fwrite()` and `fgets()`, require that we first create a file pointer using `fopen()`. This pointer then acts as a reference to that open file. We use a similar process when working with databases.

Basically there are three steps for communication with database server from PHP.

1. Create a connect to the MySQL server
2. Perform the operation (select, insert, update, modify, delete etc.)
3. Close the database connection

3.3.1 Create a connect to the MySQL server

PHP provides us with three main ways to connect to MySQL databases:

- `mysql`
- `mysqli` (MySQL improved)
- `pdo` (PHP Data Object)

mysql: The `mysql` extension for PHP is incredibly old. Not only did development stop long ago on `mysql`, but it was deprecated as of PHP 5.5.0, and has been officially removed in PHP 7.0.

mysqli (MySQL improved): is a relational database driver used in the PHP to provide an interface with MySQL databases. `MySQLi` is an improved version of the older PHP MySQL driver, offering various benefits. It features both procedural (function-oriented) and object-oriented interfaces. However, if you know you're only ever going to work with MySQL, and you want to squeeze the most out of MySQL's power from your PHP scripts, then `mysqli` is a good choice.

PDO (PHP Data Object): This is an object-oriented extension that sits between the MySQL server and the PHP engine. We can use the same extension to talk to lots of other database systems like (MySQL, PostgreSQL, Oracle).

Here in this unit we will mainly emphasis on `mysqli` (MySQL improved).

3.3.1.1 Using mysql extension

First, we have to establish a connection to the MySQL database server. This connection is then used as the access point for any future commands.

The **syntax** for connecting to a database is

\$dbc = mysql_connect (hostname, username, password, dbname, port, socket)

This method returns an object which represents the connection to a MySQL server.

The database connection is established using at least three arguments:

The hostname, which is almost always “localhost”. The username; and the password for that username. If you’re using a database through a hosting company, the company will most likely provide you with the host name, username, and password to use.

Parameter	Description
hostname	Specifies a host name or an IP address.
username	Specifies the MySQL user name.
password	Specifies the MySQL password.
dbname	Specifies the default MySQL database name to be used. It is optional. (Optional)

Closing a Database Connection

Once we are done working with a database, we can close the connection, just the same way as we close an open file:

Syntax:

mysql_close(\$dbc);

This method return true on success or false on failure

The PHP script will automatically close the database connection when the script terminates, but it’s considered good form to formally close the connection once it’s no longer needed.

Example:

Step-01

Open a new PHP document in your text editor (Notepad++) or IDE, to be named **mysql_connect.php**:

```
<html>
  <head>
    <title>Connect to MySQL DB for First Time</title>
    <h1>Odisha State Open University</h1>
  </head>
</body>
```

Step-02

Start the section of PHP code

```
<?php
```

Step03

Initialize the parameters i.e. hostname, username, password, dbname.

```
$host = "localhost";
$username = "root";
$password = "osou123";
$dbname = "dbName";
```

Step-04

Connect to MySQL, and report on the results:

```
$dbc = mysql_connect($host, $username, $password, $dbname);
```

```
if($dbc){
    print '<p>Successfully connected to MySQL!</p>';
    mysql_close($dbc);
}
else {
    print '<p style="color: red;">Could not connect to MySQL.</p>';
}
```

By placing the connection attempt as the condition in an if-else statement, you make it easy to report on whether the connection worked.

If a connection was established, a positive message is printed and then the connection is closed. Otherwise, a message stating the opposite is printed, and there is no need to close the database connection (because it wasn't opened).

Step-05

Complete the PHP code and the HTML page:

```
?>
</body>
</html>
```


Step-06

Save the file as “mysql_connect.php”, place it in the proper directory of your PHP-enabled computer, and test it in your Web browser.

If PHP has support for MySQL and the username/password/host combination you used was correct, you should see this simple message:

“Successfully connected to MySQL!”

If you see results, like below:

```
Warning: mysql_connect() [function.mysql-connect]: Access denied for user 'root'@'localhost' (using password: YES) in C:\AppServ\www\ex1.php on line 7
unable to connect to database what error: Access denied for user 'root'@'localhost' (using password: YES)
```

Double check the username and password values. They should match up with those provided to you by your Web host or those you used to create the user.

If you see call to undefined function mysql_connect..., your version of PHP doesn't support MySQL.

Example:

```
<html>
  <head>
    <title>Connect to MySQL DB for First Time</title>
    <h1>Odisha State Open University</h1>
  </head>
</body>
<?php
    $host = "localhost";
    $username = "root";
    $password = "osou123";
    $dbname = "dbName";

    $dbc = mysql_connect($host, $username, $password, $dbname);

    if($dbc){
        print '<p>Successfully connected to MySQL!</p>';
        mysql_close($dbc);
    }
    else {
```

```

        print '<p style="color: red;">Could not connect to MySQL.</p>';
    }
?>
</body>
</html>

```

3.3.1.2 Using mysqli extension (MySQL improved)

The PHP `mysqli_connect()` function is used to open a new connection to MySQL server.

This method returns an object which represents the connection to a MySQL server.

Syntax:

```
mysqli_connect(host, username, password, databasename, port, socket);
```

The `mysqli_close()` function can be used to close a connection to a database. This method return true on success or false on failure.

Syntax:

```
mysqli_close( $object );
```

Example:

```

<?php
    $host = "localhost";
    $username = "root";
    $password = "osou123";
    $dbname = "dbName";

    $object = mysqli_connect( $host, $username, $password, $dbname);

    if( $object ){
        echo "able to connect to database";
        mysqli_close( $object );
    }
    else {
        echo "unable to connect to database what error: ". mysqli_connect_error();
    }
?>

```

3.3.1.3 Using PDO extension (PHP Data Object)

PDO is a database connection abstraction library which is built into PHP since 5.1.0, which provides a common interface to talk with many different databases.

For example, you can use basically identical code to interface with MySQL or SQLite:

To establish a connection to a database program, create a new PDO object. You pass the PDO constructor a string that describes the database you are connecting to, and it returns an object that you use in the rest of your program to exchange information with the database program.

Syntax:

```
$pdo = new PDO('mysql:host=example.com;dbname=database', 'user', 'password');
```

The string passed as the first argument to the PDO constructor is called a data source name (DSN). It begins with a prefix indicating what kind of database program to connect to, then has a :, then some semicolon-separated key=value pairs providing information about how to connect.

Example:

```
<?php
$pdo=new PDO('mysql:host = example.com;dbname=database', 'user', 'password');
    if($pdo){
        print '<p>Successfully connected to MySQL!</p>';
    }
    else {
        print '<p style="color: red;">Could not connect to MySQL.</p>';
    }
?>
```

Here in this unit we will mainly emphasis on mysqli (MySQL improved).

3.3.2 Create Database and Tables in MySql

The PHP mysqli_query() function with the CREATE DATABASE query is used to create a new database in MySQL. This method return true on success false on failure.

Syntax:

```
mysqli_query( connection, query);
```

connection: Required. Specifies the MySQL connection to use.

query: Required. Specifies the query string.

In the example below, create a database named "**dbName**":

Example:

```
<?php

$_host = "localhost";
$_username = "root";
$_password = "osou123";

$object = mysqli_connect( $_host, $user_name, $pass_word);

if( $object){
    echo "connect to database success";
}
else {
    echo "connection to database error -- : ". mysqli_connect_error();
}

//create a database
$query = "CREATE DATABASE dbName";

if ( mysqli_query( $object, $query) ){
    echo " the dbName database created successfully";
}
else{
    echo "Error :- ". mysqli_error($object);
}
?>
```

3.3.2.1 Create Table, Primary Keys and Auto Increment Fields

The PHP `mysqli_query()` function with the `CREATE TABLE` mysql query is used to create a new table in MySQL.

This method return true on success false on failure.

Example:

```
<?php

$host = "localhost";
$username = "root";
$password = "osou123";
$dbname = "dbName";

$object = mysqli_connect( $host, $username, $password, $dbname);

if( $object ){
    echo "able to connect to database";
}
}
```

```

        else {
            echo "unable to connect to database what error: ". mysqli_connect_error();
        }

//create a table

$query = "CREATE TABLE Users(
    UserID INT NOT NULL AUTO_INCREMENT,
    FirstName VARCHAR(255) NOT NULL,
    LastName VARCHAR(255),
    Salary FLOAT,
    PRIMARY KEY(UserID) )";

if ( mysqli_query( $object, $query) ){
    echo " Create users table Successfully done...";
    mysqli_close( $object );
}
else{
    echo "Error :- ". mysqli_error($object);
}
?>

```

3.3.3 Insert Data into MySql Server

The MySQL INSERT INTO statement is used to insert new record in a table.

The PHP use mysqli_query() function to execute the INSERT INTO statement and send query to a MySQL server.

Example:

```

<?php

$host = "localhost";
$username = "root";
$password = "osou123";
$dbname = "dbName";

$object = mysqli_connect( $host, $username, $password, $dbname);

if( $object ){
    echo "able to connect to database";
}
else {
    echo "unable to connect to database what error: ". mysqli_connect_error();
}

//Inserting data into a table
$query = "INSERT INTO Users (FirstName, LastName, Salary) VALUES
('aseem', 'patel', 50000)";

```

```

$query1 = "INSERT INTO Users (FirstName,LastName)
          VALUES ('ABCD', 'PQRS')";

if ( mysqli_query( $object, $query) ){
    echo " Data Inserted into users table Successfully done...";
    mysqli_close( $object );
}
else{
    echo "Error :- ". mysqli_error($object);
}
?>

```

3.3.4 Mysql SELECT Statement

The MySQL SELECT statement allows you to retrieve zero or more rows from tables or views.

The SELECT statement returns a result that is a combination of columns and rows, which is also known as a result-set.

The data can be fetched from Users tables by executing SELECT statement through PHP function mysqli_query().

The mysqli_fetch_array() function fetches a result row as an associative array.

Note: Field names returned by this function are case-sensitive.

The mysqli_fetch_array() returns an array of strings that corresponds to the fetched row or NULL.

Example:

```
<?php
```

```

$host = "localhost";
$username = "root";
$password = "osou123";
$dbname = "dbName";

$object = mysqli_connect( $host, $username, $password, $dbname);

if( $object ){
    echo "able to connect to database";
}
else {
    echo "unable to connect to database what error: ". mysqli_connect_error();
}
//Select data from the users Table
$query = "SELECT * FROM Users";

```

```

$array = mysqli_query( $object, $query);

if( ! $array ){
    echo "Could not get data from Table : ".mysqli_error( $object);
}
while( $rows = mysql_fetch_array( $array )){
    echo ( $rows['FirstName']);
    echo "\n". $rows['LastName'];
    echo "\n". $rows['Salary'];
}
mysqli_close( $object );
?>

```

3.3.4.1 MySQL SELECT Statement using Where Clause

The MySQL WHERE clause is used to filter records

The WHERE clause allows you to specify exact rows to select based on given conditions.

In the example below, selects all rows from the "Users" table where "LastName = 'patel' ":

Example:

```
<?php
```

```

$host = "localhost";
$username = "root";
$password = "osou123";
$dbname = "dbName";
$object = mysqli_connect( $host, $username, $password, $dbname);

```

```

if( $object ){
    echo "able to connect to database";
}
else {
    echo "unable to connect to database what error: ". mysqli_connect_error();
}

```

//Select data from the users Table using Where Clause

```

$query = "SELECT * FROM Users WHERE LastName='patel'";
$array = mysqli_query( $object, $query);
if( ! $array ){
    echo "Could not get data from Table : ".mysqli_error( $object);
}
while( $rows = mysql_fetch_array( $array )){
    echo ( $rows['FirstName']);
    echo "\n". $rows['LastName'];
    echo "\n". $rows['Salary'];
}
mysqli_close( $object );
?>

```

3.3.5 Update MySql Records

The MySQL UPDATE statement is used to update existing data in tables. It can be used to change column values of a single row, group of rows or all rows in a tables.

Syntax:

```
UPDATE table_name1, table_name2, ...
    SET column_name1 = expression_value,
    column_name2 = expression_value, ...
    WHERE condition ;
```

Note: The PHP use mysqli_query() function to execute the UPDATE statement and send query to a MySQL server.

The following example update data in 'Users' table that we have created earlier:

Example:

```
<?php
```

```
$host = "localhost";
$username = "root";
$password = "osou123";
$dbname = "dbName";
$object = mysqli_connect( $host, $username, $password, $dbname);

if( $object ){
    echo "able to connect to database";
}
else {
    echo "unable to connect to database what error: ". mysqli_connect_error();
}

//Update existing data from the users Table using Where Clause

$query = "UPDATE Users SET FirstName='Sibananda' Salary=70000
        WHERE Salary =50000";
$array = mysqli_query( $object, $query);

if( ! $array ){
    echo "Could not get Updated data in Table : ".mysqli_error( $object);
}
else{
    echo "Data Successfully Updated in Users table...";
}
mysqli_close( $object );
?>
```


3.3.6 Delete MySql Records

The MySQL DELETE statement allows you to remove records from not only one table but also multiple tables using a single DELETE statement.

Syntax:

```
DELETE FROM table_name WHERE condition;
```

Note: The mysql WHERE keyword specifies which record should be deleted. If you omit the WHERE statement, in this situation all table records will be deleted.

The following example Delete data in 'Users' table that we have created earlier:

Example:

<?php

```
$host = "localhost";
$username = "root";
$password = "osou123";
$dbname = "dbName";
$object = mysqli_connect( $host, $username, $password, $dbname);

if( $object ){
    echo "able to connect to database";
}
else {
    echo "unable to connect to database what error: ". mysqli_connect_error();
}

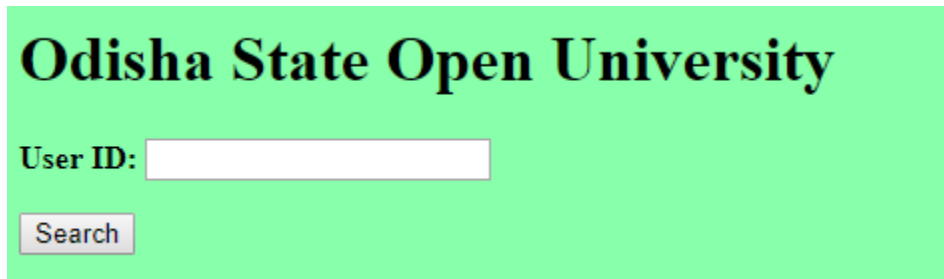
//Update existing data from the users Table using Where Clause

$query = "DELETE From Users WHERE Salary =70000";
$array = mysqli_query( $object, $query);

if( ! $array ){
    echo "Could not get Deleted data in Table : ".mysqli_error( $object);
}
else{
    echo "Data Successfully Deleted from Users table...";
}
mysqli_close( $object );
?>
```

3.4 Example database access from Webpage

Suppose we want to write a script that allows a user of our Web site to view the information about a particular participant. The Web site user would see a form such as the following.



This form can be generated by the following HTML code.

File: index.html

```
<html>
  <head>
    <title>Connect to MySQL DB for First Time</title>
    <h1> Odisha State Open University</h1>
  </head>
<body bgcolor="#88FFAA">
<form method="post" action="user.php">
  <p><b>User ID:</b> <input type="text" name="userid" /></p>
  <p><input type="submit" value="Search" /></p>
</form>
</body>
</html>
```

The file **user.php** would be the following

File: user.php

```
<?php
```

```
$host = "localhost";
$username = "root";
$password = "osou123";
$dbname = "dbName";
$object = mysqli_connect( $host, $username, $password, $dbname);

if( $object ){
  echo "able to connect to database";
}
```

```

else {
    echo "unable to connect to database what error: ". mysqli_connect_error();
}

$query = "SELECT * FROM Users WHERE userid = '$form_userid' ";
$rows = mysqli_query( $object, $query);

if( !$rows){
    echo "Could not get data from Table : ".mysqli_error( $object);
}
elseif (mysqli_num_rows($rows) == 0) {

    $error = "No such user name found";
}
else{
    $error = FALSE;
    $first_name = mysqli_result($rows, 0, 1);
    $last_name = mysqli_result($rows, 0, 2);
    $salary = mysqli_result($rows, 0, 3);
}
?>
<html>
    <head><title>User information</title>
    </head>
<body>
    <?php
        if($error) {
            echo "<h1>Error accessing user information</h1>\n";
            echo "<p>$error</p>\n";
        }
        else{
            echo "<h1>Information about $form_userid</h1>\n";
            echo "<p>User ID: <tt>$form_userid</tt></p>\n";
            echo "<p>First_name: $first_name</p>\n";
            echo "<p>Last_name: $last_name</p>\n";
            echo "<p>Salary: <tt>$salary</tt></p>\n";
        }
    ?>
</body>
</html>

```

3.5 Let us Sum Up

PHP is well known for its smooth database integration, especially with MySQL. It's actually quite easy to connect to a MySQL database from within PHP. Once you've established the connection, you can send SQL commands to the database and receive the results as data you can use in your PHP program. So, we'll use MySQL here, since it is easily available and used quite frequently for Web pages in conjunction with PHP.

In this Unit we learnt about the MySQL database if features and why we need to use MySQL in PHP. Also understand how data is organized in a database, Establish a database connection, creating a table, insert data, select data, update and delete data in the database

3.6 References:

1. <http://www.toves.org/books/php/ch07-db/index.html>
2. <https://www.phptherightway.com/#databases>
3. <http://www.w3webtutorial.com/php/php-mysql-connection.php>
4. <http://www.allitebooks.com/?s=php>

3.7 Model Questions

Q.N. 01: Why MySQL use in PHP? What are the features MySQL?

Q.N. 02: What are the different way to connect MySQL Database?

Q.N. 03: Write a program to connect MySQL Database in PHP.

Q.N. 04: Write a PHP program to create a table "users" using MySQL.

Q.N. 05: Write a PHP program to insert data in table "users" using MySQL.

Q.N. 06: Write a PHP program to select data from a table "users" using MySQL.

Q.N. 07: Write a PHP program to update data in a table "users" using MySQL.

Q.N. 08: Write a PHP program to delete a table "users" using MySQL.

Q.N. 09: Write a PHP program to access database from Webpage.

Q.N. 10: Write a PHP program to search details of an employee using it's a employee code in MySQL.