Unit-5

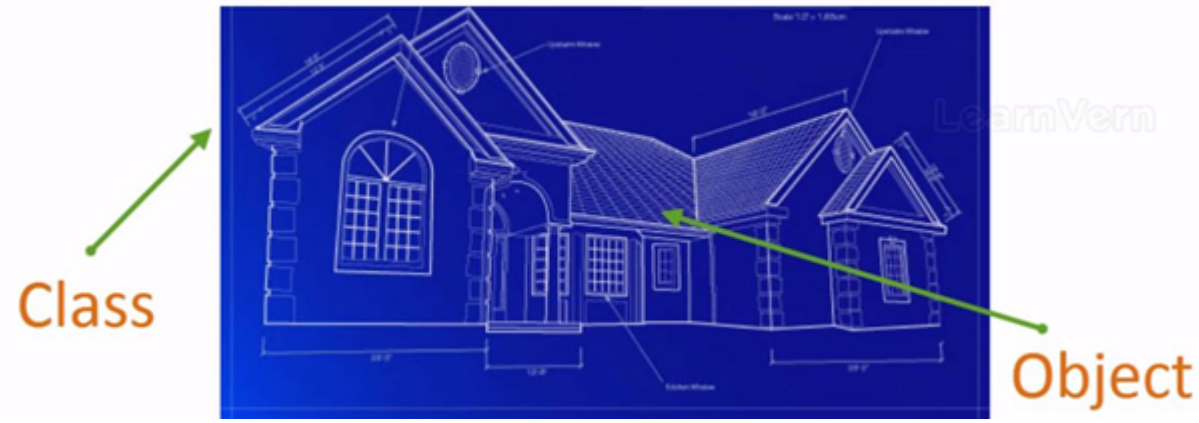
PHP WITH OOPS CONCEPT

Object-oriented programming is a programming model organized around **Object rather than the actions** and **data rather than logic**.

Class:

A class is an entity that determines how an object will behave and what the object will contain. In other words, it is a blueprint or a set of instruction to build a specific type of object.

In PHP, declare a class using the class keyword, followed by the name of the class and a set of curly braces ({}).



This is the blueprint of the construction work that is class, and the houses and apartments made by this blueprint are the objects.

Syntax to Create Class in PHP

```php
<?php
class MyClass
    {
        // Class properties and methods go here
    }
?>
```

Important note:

In PHP, to see the contents of the class, use var_dump(). The var_dump() function is used to display the structured information (type and value) about one or more variables.

Syntax:

```php
var_dump($obj);
```

Object:

A class defines an individual instance of the data structure. We define a class once and then make many objects that belong to it. Objects are also known as an instance.

An object is something that can perform a set of related activities.

Syntax:

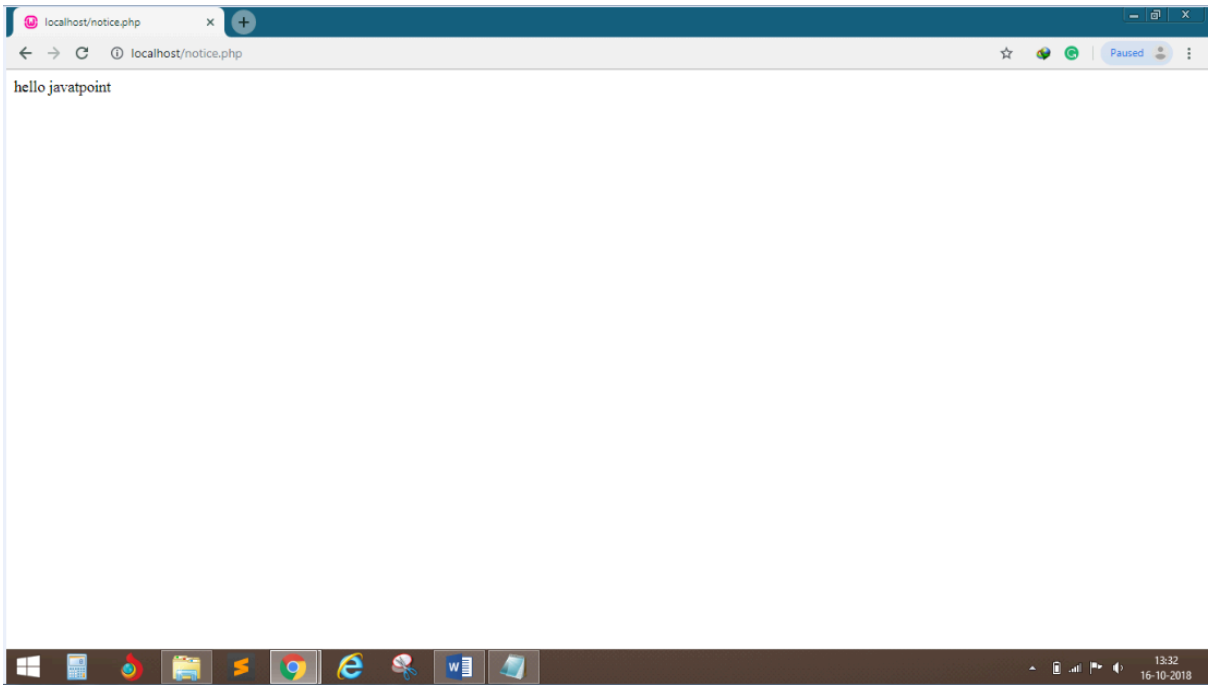
```php
<?php
class MyClass
{
        // Class properties and methods go here
}
$obj = new MyClass;
var_dump($obj);
?>
```

Example of class and object:

```php
<?php
class demo
{
        private $a= "hello javatpoint";
         public function display()
        {
         echo $this->a;
        }
}
$obj = new demo();
    $obj->display();
```

```
12. ?>
```

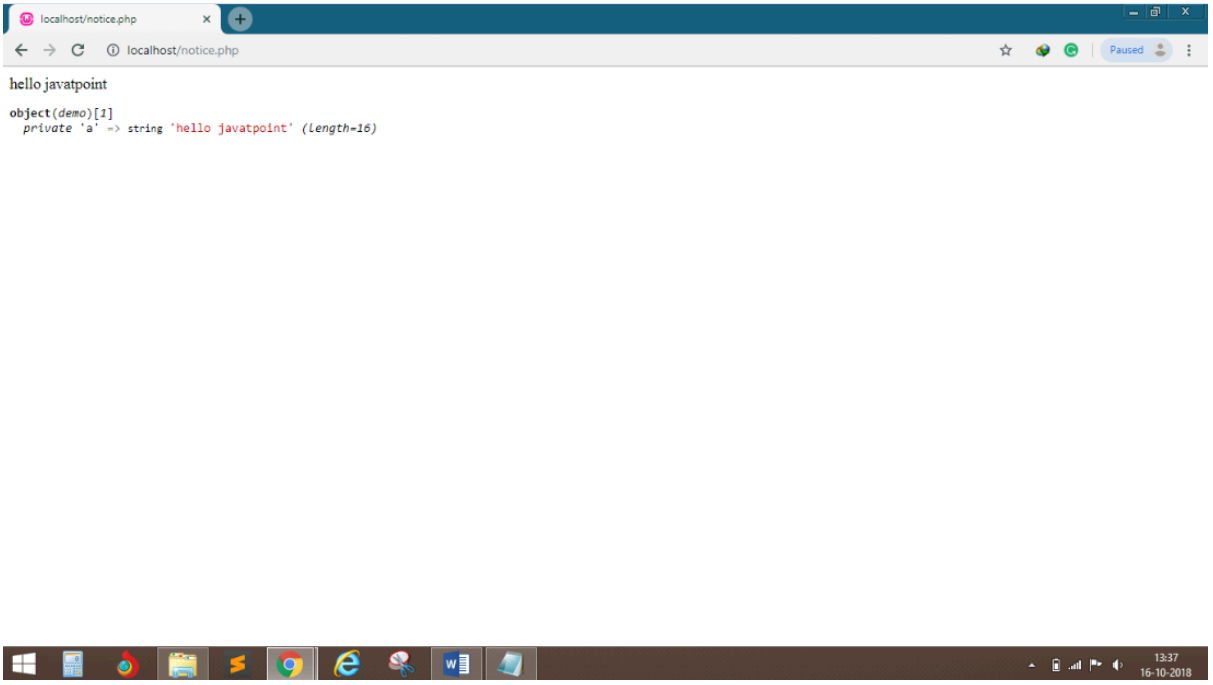
Output:



Example 2: Use of var_dump($obj);

```
1.  <?php
2.  class demo
3.  {
4.          private $a= "hello javatpoint";
5.          public function display()
6.          {
7.          echo $this->a;
8.          }
9.  }
10. $obj = new demo();
11.      $obj->display();
12.     var_dump($obj);
13. ?>
```

Output:



Inheritance

It is a concept of accessing the features of one class from another class. If we inherit the class features into another class, we can access both class properties. We can extends the features of a class by using 'extends' keyword.

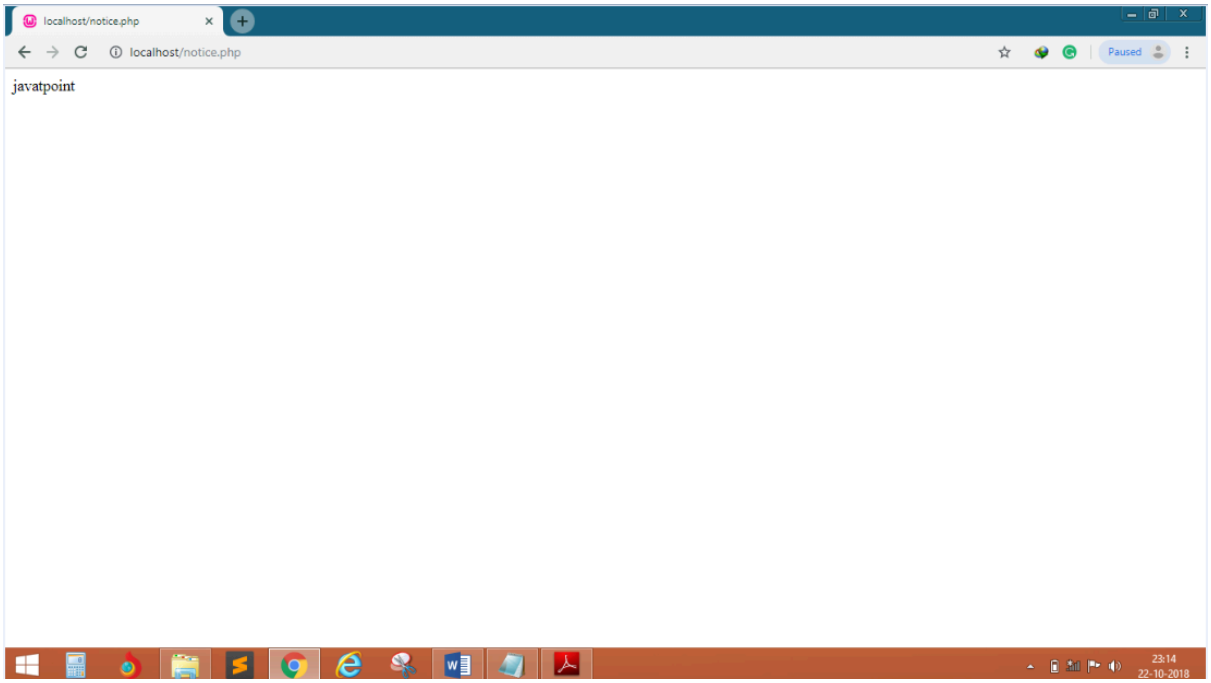
- It supports the concept of **hierarchical classification**.
- Inheritance has three types, **single, multiple and multilevel Inheritance**.
- **PHP** supports only **single inheritance**, where only one class can be **derived from single parent class**.
- We can simulate multiple inheritance by using **interfaces**.

Example 1

```
1.  <?php
2.  class a
3.     {
4.        function fun1()
5.        {
6.           echo "javatpoint";
7.        }
8.     }
9.     class b extends a
10.    {
11.       function fun2()
12.       {
```

```
13.             echo "SSSIT";
14.         }
15.     }
16.     $obj= new b();
17.     $obj->fun1();
18. ?>
```

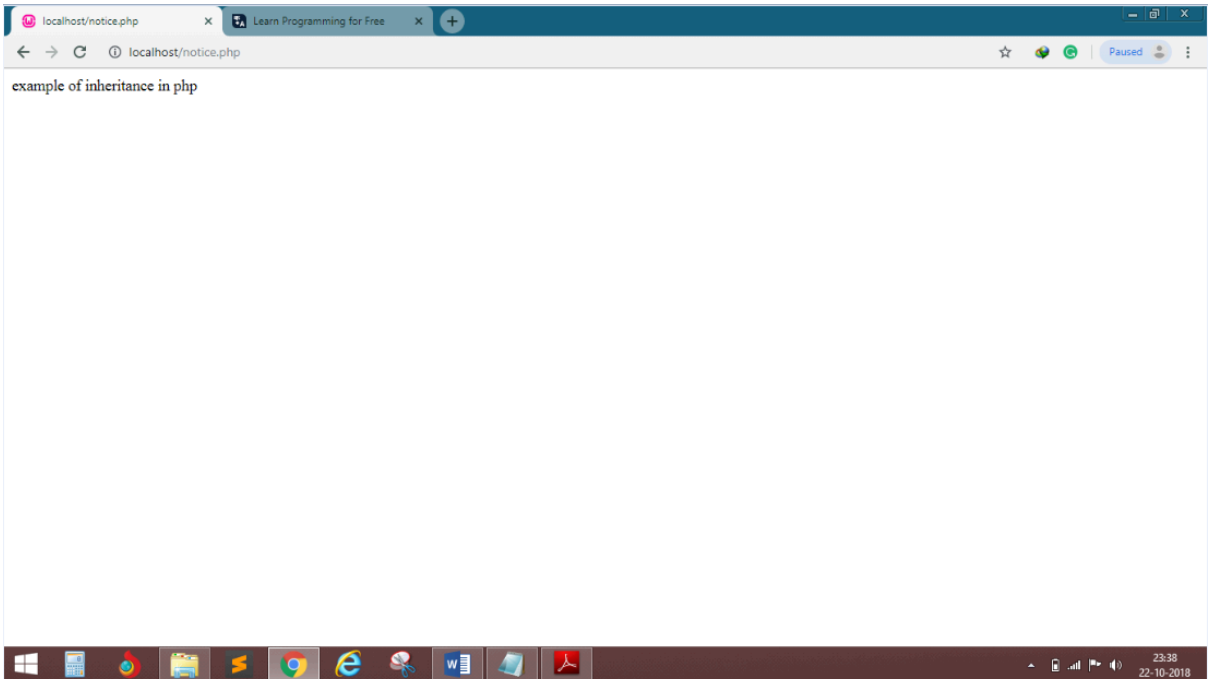
Output:



Example 2

```
1.  <?php
2.      class demo
3.      {
4.          public function display()
5.          {
6.              echo "example of inheritance  ";
7.          }
8.      }
9.      class demo1 extends demo
10.     {
11.         public function view()
12.         {
13.             echo "in php";
14.         }
15.     }
16.     $obj= new demo1();
17.     $obj->display();
18.     $obj->view();
19. ?>
```

Output:



PHP Form Handling

We can create and use forms in PHP. To get form data, we need to use PHP superglobals $_GET and $_POST.

The form request may be get or post. To retrieve data from get request, we need to use $_GET, for post request $_POST.

PHP Get Form

Get request is the default form request. The data passed through get request is visible on the URL browser so it is not secured. You can send limited amount of data through get request.

Let's see a simple example to receive data from get request in PHP.

File: form1.html

```
<form action="welcome.php" method="get">
Name: <input type="text" name="name"/>
<input type="submit" value="visit"/>
</form>
```

File: welcome.php

```
<?php
$name=$_GET["name"];//receiving name field value in $name variable
echo "Welcome, $name";
```

```
?>
```

PHP Post Form

Post request is widely used to submit form that have large amount of data such as file upload, image upload, login form, registration form etc.

The data passed through post request is not visible on the URL browser so it is secured. You can send large amount of data through post request.

Let's see a simple example to receive data from post request in PHP.

File: form1.html

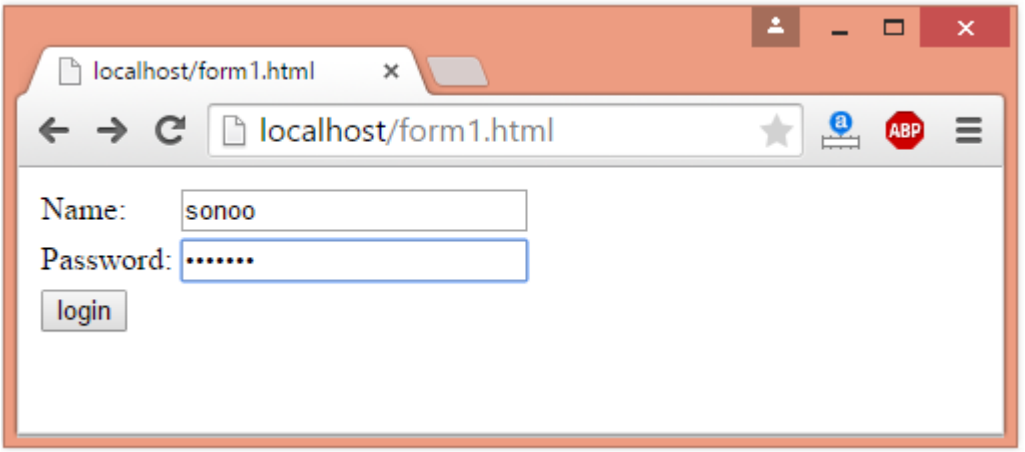
```
<form action="login.php" method="post">
<table>
<tr><td>Name:</td><td> <input type="text" name="name"/></td></tr>
<tr><td>Password:</td><td> <input type="password" name="password"/></td></tr>
<tr><td colspan="2"><input type="submit" value="login"/>  </td></tr>
</table>
</form>
```

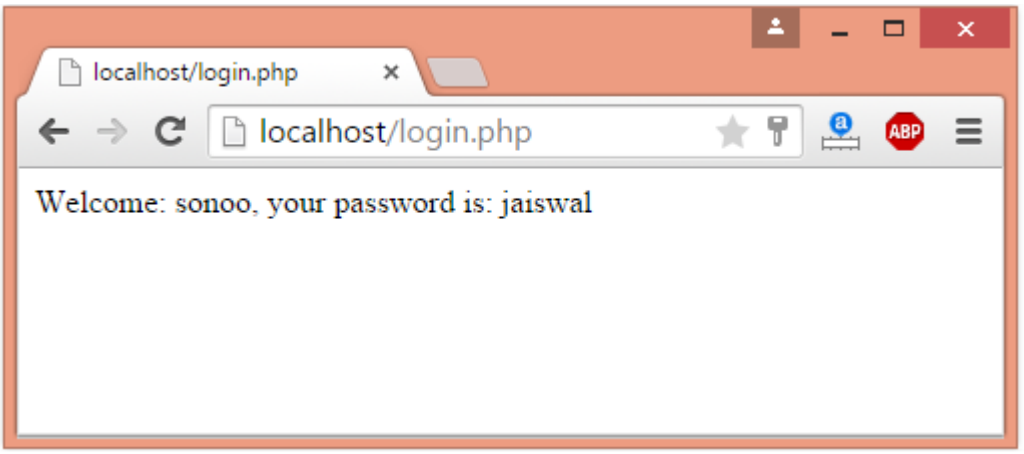
File: login.php

```
<?php
$name=$_POST["name"];//receiving name field value in $name variable
$password=$_POST["password"];//receiving password field value in $password vari
able

echo "Welcome: $name, your password is: $password";
?>
```

Output:

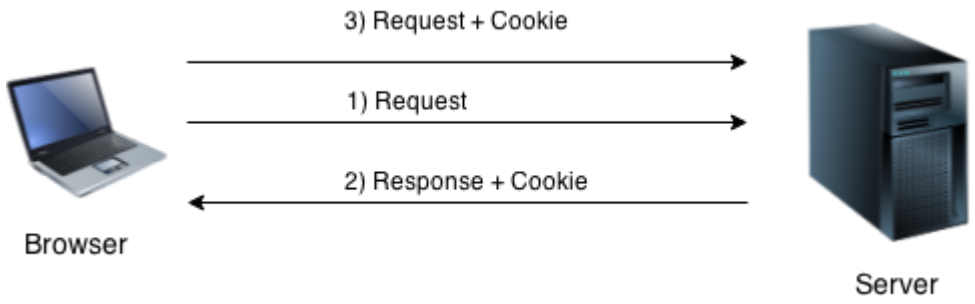




PHP Cookie

PHP cookie is a small piece of information which is stored at client browser. It is used to recognize the user.

Cookie is created at server side and saved to client browser. Each time when client sends request to the server, cookie is embedded with request. Such way, cookie can be received at the server side.



In short, cookie can be created, sent and received at server end.

Exception Handling in PHP

Exception handling is a powerful mechanism of PHP, which is used to handle runtime errors (runtime errors are called exceptions). So that the normal flow of the application can be maintained.

The main purpose of using exception handling is **to maintain the normal execution of the application**.

What is an Exception?

An exception is an unexcepted outcome of a program, which can be handled by the program itself. Basically, an exception disrupts the normal flow of the program. But it is different from an error because an exception can be handled, whereas an error cannot be handled by the program itself.

In other words, - "An unexpected result of a program is an exception, which can be handled by the program itself." Exceptions can be thrown and caught in PHP.

Why needs Exception Handling?

PHP provides a powerful mechanism, exception handling. It allows you to handle runtime errors such as IOException, SQLException, ClassNotFoundException, and more. A most popular example of exception handling is - divide by zero exception, which is an arithmetic exception.

> ***Note: Exception handling is required when an exception interrupts the normal execution of the program or application.***

Exception handling is almost similar in all programming languages. It changes the normal flow of the program when a specified error condition occurs, and this condition is known as exception. PHP offers the following keywords for this purpose:

try -

The try block contains the code that may have an exception or where an exception can arise. When an exception occurs inside the try block during runtime of code, it is caught and resolved in catch block. The try block must be followed by catch or finally block. A try block can be followed by minimum one and maximum any number of catch blocks.

catch -

The catch block contains the code that executes when a specified exception is thrown. It is always used with a try block, not alone. When an exception occurs, PHP finds the matching catch block.

throw -

It is a keyword used to throw an exception. It also helps to list all the exceptions that a function throws but does not handle itself.

Remember that each throw must have at least one "catch".

finally -

The finally block contains a code, which is used for clean-up activity in PHP. Basically, it executes the essential code of the program.

What happens when an exception is triggered -

- The current state of code is saved.
- The execution of the code is switched to a predefined exception handler function.
- Depending on the situation, the handler can halt the execution of program, resume the execution from the saved code state, or continue the execution of the code from another location in the code.

Advantage of Exception Handling over Error Handling

Exception handling is an important mechanism in PHP, which have the following advantages over error handling -

Grouping of error types -

In PHP, both basic and objects can be thrown as exception. It can create a hierarchy of exception objects and group exceptions in classes and also classify them according to their types.

Keep error handling and normal code separate -

In traditional error handling, if-else block is used to handle errors. It makes the code unreadable because the code for handling errors and conditions got mixed. Within the try-catch block, exception keeps separate from the code and code become readable.

FRONT END AND BACK END DATABASE CONNECTIVITY.

Front end code:

<html>

<head>

```
<title>Registration</title>
</head>
<body bgcolor="orange">
<center><h1>Registration</h1>
<form action="welcome.php"
method="post">
ENTER NAME:<input type="text"
name="Name"></br>
ENTER PASSWORD:<input type="text"
name="Password"></br>
ENTER EMAIL:<input type="text"
name="Email"></br>
ENTER COUNTRY:<input type="text"
name="Country"></br>
<input type="submit" value="REGISTER">
<input type="reset" value="RESET">
</form>
</center>
```

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

Backend Code: (PHP)

```
<?php
$Name=$_POST['Name'];
$Password=$_POST['Password'];
$Email=$_POST['Email'];
$Country=$_POST['Country'];
$conn = new mysqli('localhost:3306','root','','nd');
if($conn->connect_error)
{
die('Connection failed :'.$conn->connect_error);
}
else
{
$stmt=$conn->prepare("insert into nt(Name, Password, Email, Country) value(?,?,?,?)");
```

```
$stmt->bind_param('ssss',$Name,
$Password, $Email, $Country);

$stmt->execute();

echo "Welcome: $Name Registration
Successfully....";

$stmt->close();

$conn->close();

}

?>
```