Introduction to PHP (Hypertext Pre Processor)

Dynamic page

A web page that is dynamically modified before being sent to the browser that requests it.

Normally such pages are associated with rapidly changing information such as share price or number of items in stock in a warehouse.

These are number of technologies used to implement dynamic pages. The most popular currently are Active Server Page (ASP), Java Server Page (JSP), and Hyper Text Pre Processor (PHP).

Authoring a dynamic page consists of writing the HTML first, and then adding the server side script or tags to the HTML to make the page dynamic.

When you view the resulting code, the language appears embedded in the pages HTML accordingly, these languages are known as HTML embedded programming languages.

Server Technology	Application Server	Language
ASP	Microsoft IIS 5.0	VB Script, JavaScript
ASP.NET	Microsoft IIS6.0 with .NET frame work	Visual Basic, C#
PHP	Apache	PHP
JSP	SunONE application server, Apache Tom cat	Java
Cold fusion	Macromedia cold fusion MX7	Cold fusion Markup language (CFML)

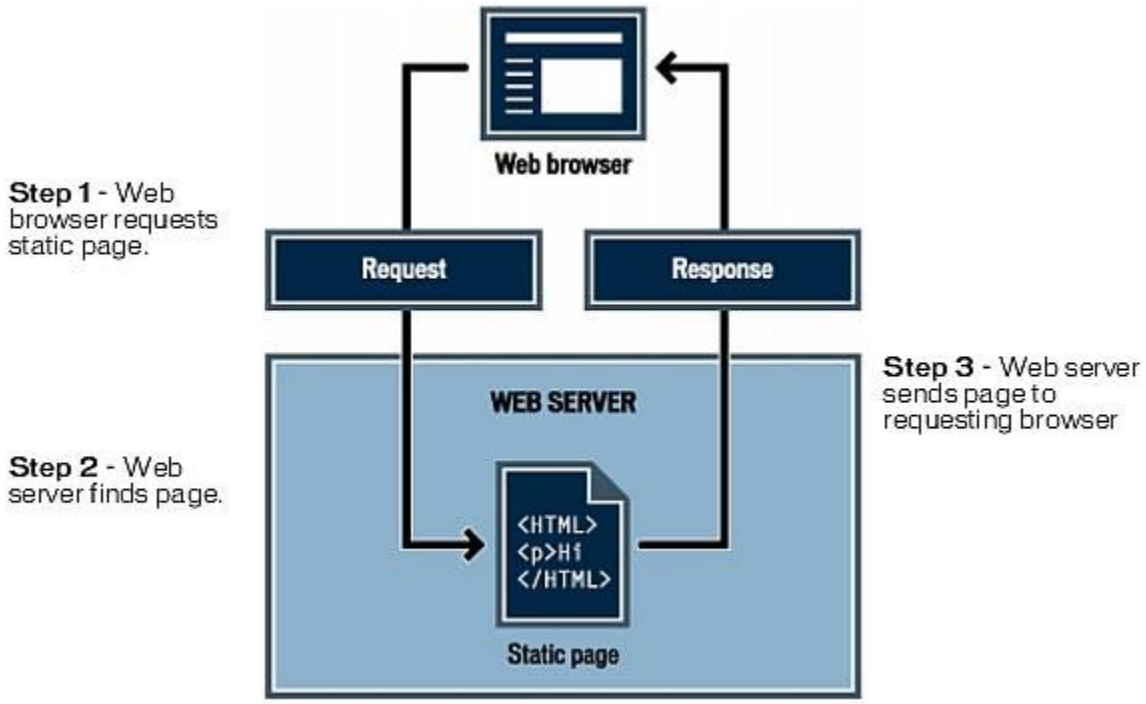
Processing Dynamic Pages

When a web server receives a request for a static web page, a server senses the page directly to the requesting browser.

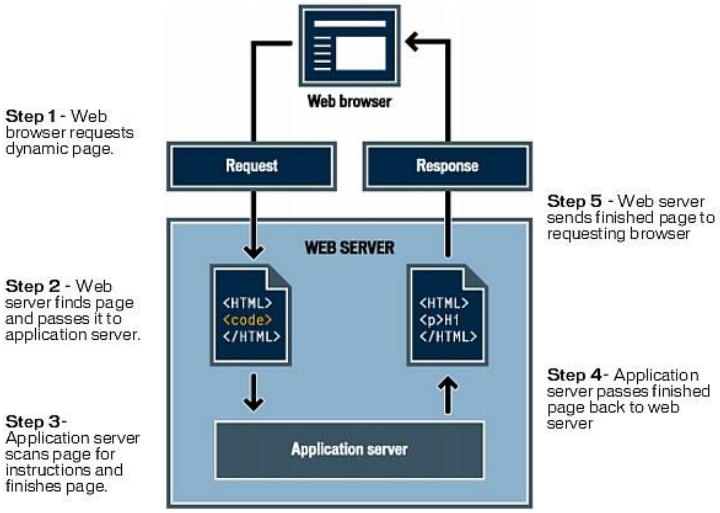
When the web server receives a request for a dynamic page. However, it reacts differently. It passes the page to a special piece of software responsible for finishing the page. This special software is called an application server.

The application server reads the code on the page, finishes the page according to the instructions in the code, and then removes the code from the page. The result is a static page that the application server passes back to the web server, which then sense the page to the requesting browser gets when the page arrives in page HTML.

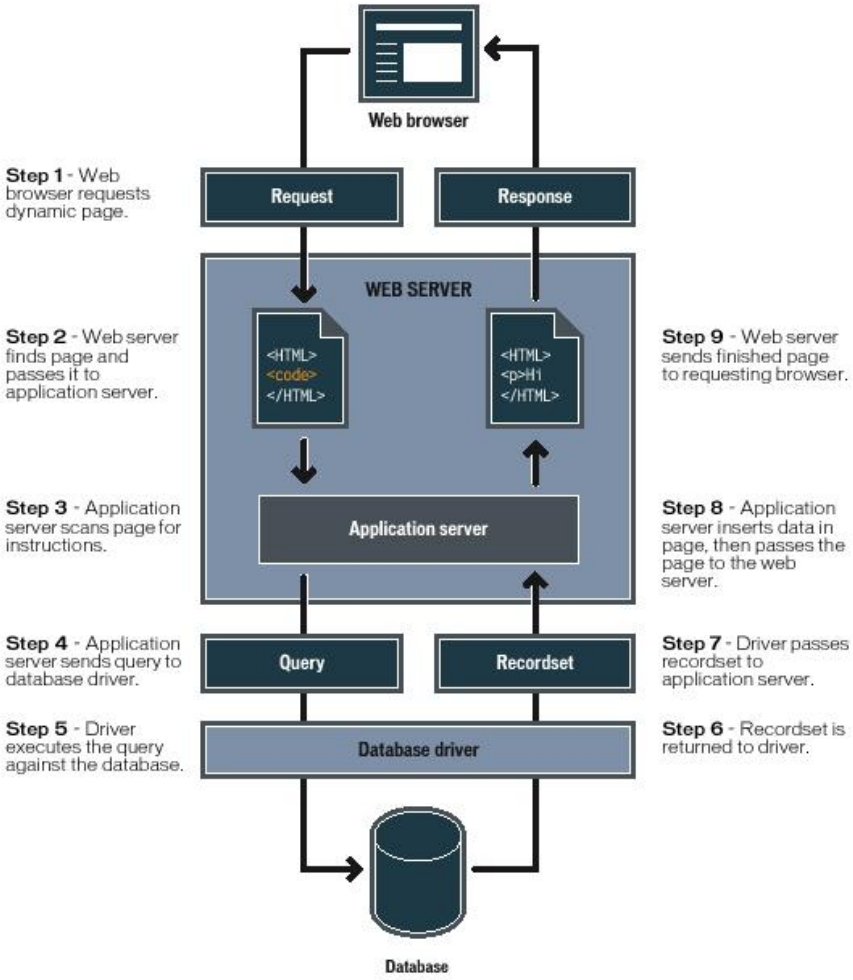
Processing static web pages



Processing dynamic web pages



Processing dynamic web pages with database



PHP

PHP: Hypertext Preprocessor is a general-purpose programming language originally designed for web development. It was originally created by Rasmus Lerdorf in 1994; the PHP reference implementation is now produced by The PHP Group.

PHP is a popular general-purpose scripting language that is especially suited to web development.

PHP Features

There are given many features of PHP.

- **Performance**: Script written in PHP executes much faster than those scripts written in other languages such as JSP & ASP.
- **Open Source Software**: PHP source code is free available on the web, you can developed all the version of PHP according to your requirement without paying any cost.
- **Platform Independent**: PHP are available for WINDOWS, MAC, LINUX & UNIX operating system. A PHP application developed in one OS can be easily executed in other OS also.
- **Compatibility**: PHP is compatible with almost all local servers used today like Apache, IIS etc.
- **Embedded**: PHP code can be easily embedded within HTML tags and script.

PHP syntax

PHP's syntax and semantics are similar most other programming languages. (C and Java) with the addition that all PHP code contain with a tag. All PHP code must be contain with the following.

<?php ?> OR <script language= "php"> </script>

Ex :-

```
<html>
<head>
        <title>Welcome</title>
</head>
<body>
        <?php
                echo "Hello world";
        ?>
</body>
</html>
```

If you save this file and place it on PHP enabled server and load it up in your web browser, then you should see "Hello world" displayed.

PHP echo and print Statements

echo and **print** are more or less the same. They are both used to output data to the screen.

The differences are small: echo has no return value while print has a return value of 1 so it can be used in expressions. echo can take multiple parameters (although such usage is rare) while print can take one argument. echo is marginally faster than print.

The echo Statement

The echo statement can be used with or without parentheses: echo or echo().

```
<?php
echo "<h2>PHP is Fun!</h2>";
echo "Hello world!<br>";
echo "I'm about to learn PHP!<br>";
echo "This ", "string ", "was ", "made ", "with multiple parameters.";
?>
```

The PHP print Statement

The print statement can be used with or without parentheses: print or print().

```
<?php
print "<h2>PHP is Fun!</h2>";
print "Hello world!<br>";
print "I'm about to learn PHP!";
?>
```

PHP variable

A variable is a means of a storing a value. A variable you can be used throughout your code, instead of having to type out the actual value over and over again. In PHP you define a variable with the following form.

```
$abc=123
$name=”Raja”
$ok=false
```

Note :- Variable names are case sensitive, so use the exact same capitalization when using a variable.

```
<?php
        $name=”Raja”;
        $add=”Batticaloa”;
        echo “Your name is “.$name;
        echo “Your address  is “.$add;
?>
```

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

Rules for PHP variables:

- A variable starts with the $ sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive ($age and $AGE are two different variables)

PHP Data Types

Variables can store data of different types, and different data types can do different things.

PHP supports the following data types:

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL

PHP Operators

Operators are used to perform operations on variables and values.

PHP divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Increment/Decrement operators
- Logical operators

PHP Conditional Statements

Very often when you write code, you want to perform different actions for different conditions. You can use conditional statements in your code to do this.

In PHP we have the following conditional statements:

- if statement - executes some code if one condition is true
- if...else statement - executes some code if a condition is true and another code if that condition is false
- if...elseif...else statement - executes different codes for more than two conditions
- switch statement - selects one of many blocks of code to be executed

PHP - The if Statement

The if statement executes some code if one condition is true.

Syntax

```
if (condition) {
   code to be executed if condition is true;
}
```

```
<?php
$t = date("H");

if ($t < "20") {
   echo "Have a good day!";
}
?>
```

PHP - The if...else Statement

The if...else statement executes some code if a condition is true and another code if that condition is false.

Syntax

```
if (condition) {
   code to be executed if condition is true;
} else {
   code to be executed if condition is false;
}
```

```
<?php
$t = date("H");

if ($t < "20") {
   echo "Have a good day!";
} else {
   echo "Have a good night!";
}
?>
```

PHP - The if...elseif...else Statement

The if...elseif...else statement executes different codes for more than two conditions.

Syntax

```
if (condition) {
   code to be executed if this condition is true;
} elseif (condition) {
   code to be executed if first condition is false and this condition is true;
} else {
   code to be executed if all conditions are false;
}
```

```
<?php
$t = date("H");

if ($t < "10") {
   echo "Have a good morning!";
} elseif ($t < "20") {
   echo "Have a good day!";
} else {
   echo "Have a good night!";
```

```
}
?>
```

The PHP switch Statement

Use the switch statement to **select one of many blocks of code to be executed**.

```
<?php
$favcolor = "red";

switch ($favcolor) {
  case "red":
    echo "Your favorite color is red!";
    break;
  case "blue":
    echo "Your favorite color is blue!";
    break;
  case "green":
    echo "Your favorite color is green!";
    break;
  default:
    echo "Your favorite color is neither red, blue, nor green!";
}
?>
```

PHP Loops

Often when you write code, you want the same block of code to run over and over again a certain number of times. So, instead of adding several almost equal code-lines in a script, we can use loops.

Loops are used to execute the same block of code again and again, as long as a certain condition is true.

In PHP, we have the following loop types:

- while - loops through a block of code as long as the specified condition is true
- do...while - loops through a block of code once, and then repeats the loop as long as the specified condition is true
- for - loops through a block of code a specified number of times
- foreach - loops through a block of code for each element in an array

The PHP while Loop

The while loop executes a block of code as long as the specified condition is true.

Syntax

```
while (condition is true) {
   code to be executed;
}
```

```
<?php
$x = 1;

while($x <= 5) {
   echo "The number is: $x <br>";
   $x++;
}
?>
```

The PHP do...while Loop

The do...while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.

Syntax

```
do {
   code to be executed;
} while (condition is true);
```

```
<?php
$x = 1;

do {
   echo "The number is: $x <br>";
   $x++;
} while ($x <= 5);
?>
```

The PHP for Loop

The for loop is used when you know in advance how many times the script should run.

Syntax

```
for (init counter; test counter; increment counter) {
   code to be executed for each iteration;
}
```

Parameters:

- *init counter*: Initialize the loop counter value
- *test counter*: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- *increment counter*: Increases the loop counter value

```
<?php
for ($x = 0; $x <= 10; $x++) {
   echo "The number is: $x <br>";
}
?>
```

The PHP foreach Loop

The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.

Syntax

foreach ($*array* as $*value*) {
  code to be executed;
}

For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element.

```
<?php
$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $value) {
  echo "$value <br>";
}
?>
```

PHP Arrays

An array stores multiple values in one single variable:

Create an Array in PHP

In PHP, the array() function is used to create an array:

array();

In PHP, there are three types of arrays:

- **Indexed arrays** - Arrays with a numeric index
- **Associative arrays** - Arrays with named keys
- **Multidimensional arrays** - Arrays containing one or more arrays

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo count($cars);
?>
```

PHP - Form Process

Form processing is a very important feature in PHP. It is through the uses of forms that users interact with your web pages and through which you have collect information for personalizing pages for your visitors.

HTML forms controls allow users to input data and make selections and provide a mechanism by which users interact a web page. A form is useful to extent that it is backed while a processing script coded on a web page or by a program running on a web server.

Information entered on to a form it submitted for processing by clicking the form's submit button. The information in the form fields is then transmitted to the page identify in the "ACTION" of the <form> tag. When this URL request and the accompanying form data arrive at the server, the targeted page is retrieved and the data are made available to it for processing.

```
<html>
        <head><title>Hello</title></head>
<body>
<form action=”process.php” method=”POST”>
        First name : <input type =”text” name=”fname”><br/>
        Last name : <input type =”text” name=”lname”><br/>
        Email : <input type =”text” name=”mail”><br/>
        Message : <textarea  name=”msg” cols=”30”
        rows=”5”></textarea><br/>
        <input type=”submit” name=”submit” value=”send”>
</form>
</body>
</html>
```

- Save form1.html

```
<?php
echo “Your first name is : “.$_POST[“fname”].”<br/>”;
echo “Your last name is : “.$_POST[“lname”].”<br/>”;
echo “Your email is : “.$_POST[“mail”].”<br/>”;
echo “Your message is : “.$_POST[“msg”];
?>
```

- Save process.php at the same location.
- After that call the form1.html in the browser.

When the submit button in the above example is clicked, all of the form data entered by the user is passed to the page, process.php, for processing.

Since this form is using the POST method, all form data is passed using the PHP $_POST variable. Each input control is uniquely identified by the value assigned to its name attribute value becomes the index value of the $_POST array.

Since the PHP $_POST and $_GET variables use the value associated with the form controls name attribute. It is important that all form elements be assigned a unique name value.

If the GET method is used instead of the POST method, the $_POST array is replaced by the $_GET array. It is also possible to take advantage of the $_REQUEST array.

Although both specify the manner in which form information is transmitted to the page identify in the action attribute the POST method is recommended.

POST

Information from the form is transmitted as a separate data stream the service associated with the processing of the form has site effect, the method should be POST. The POST method does protect integrity of the data since it can't be viewed.

- Appends form-data inside the body of the HTTP request (data is not shown is in URL)
- Has no size limitations
- Form submissions with POST cannot be bookmarked

GET

Information from the form is appended to the action URL creating a query string. If the processing of a form has no lasting obserble effect on the state of the world, then the form method should be GET. Many database searchers have no visible site effects and make ideal applications of query forms.

- Appends form-data into the URL in name/value pairs
- The length of a URL is limited (about 3000 characters)
- Never use GET to send sensitive data! (will be visible in the URL)
- Useful for form submissions where a user wants to bookmark the result
- GET is better for non-secure data, like query strings in Google

--------- *Our class room marks processing program should be here …….*

PHP – MySQL

What is MySQL?

MySQL is a relational database management system. The data in MySQL is stored in database objects called tables. A table is a collection of related data entries and it consists of columns and rows.

Database are useful when storing information categorically. A company may have a database with the following tables, employees, products, customers and orders.

Queries

A query is a question or a request. With MySQL, we can query a database for specific information. And have a record set returned.

PHP Connect to MySQL

Before we can access data in the MySQL database, we need to be able to connect to the server:

In PHP, this is done with the Mysqli_connect() function.

Syntax :-

mysqli_connect(*host, username, password, dbname, port, socket*)

host	Optional. Specifies a host name or an IP address
username	Optional. Specifies the MySQL username
password	Optional. Specifies the MySQL password
dbname	Optional. Specifies the default database to be used
port	Optional. Specifies the port number to attempt to connect to the MySQL server
socket	Optional. Specifies the socket or named pipe to be used

```
$con = mysqli_connect("localhost","my_user","my_password","my_db");
```

In the following example we store the connection in a variable. [$con]. The "die" part will be executed if the connection fails.

The connection will be closed automatically when the script ends. To close connection before used the mysqli_close( ) function.

```php
<?php

    $servername = "localhost";
    $username = "username";
    $password = "password";

    // Create connection
    $conn = mysqli_connect($servername, $username,
    $password);
    // Check connection
    if (!$conn) {
      die("Connection failed: " . mysqli_connect_error());
    }
    echo "Connected successfully";
?>
```

PHP MySQL INSERT Query

The INSERT INTO statement is used to insert new rows in a database table.

Let's make a SQL query using the INSERT INTO statement with appropriate values, after that we will execute this insert query through passing it to the PHP mysqli_query() function to insert data in table.

```php
<?php
// Connection
    $link = mysqli_connect("localhost", "root", "", "demo");

// Check connection
    if($link === false){
        die("ERROR: Could not connect. " .
        mysqli_connect_error()); }

// Attempt insert query execution
```

```
    $sql = "INSERT INTO persons (first_name, address, email)
    VALUES ('Peter', 'Batticaloa', 'peter@gmail.com')";

    if(mysqli_query($link, $sql)){
        echo "Records inserted successfully."; }
    else{
        echo "ERROR: Could not able to execute $sql. "
        .mysqli_error($link); }

// Close connection
        mysqli_close($link);
?>
```

PHP MySQL SELECT Query

The SQL SELECT statement is used to select the records from database tables. Let's make a SQL query using the SELECT statement, after that we will execute this SQL query through passing it to the PHP mysqli_query() function to retrieve the table data.

```
<?php
// Connection
    $link = mysqli_connect("localhost", "root", "", "demo");

// Check connection
    if($link === false){
        die("ERROR: Could not connect. " .
        mysqli_connect_error()); }

// Attempt select query execution
$sql = "SELECT * FROM persons";
if($result = mysqli_query($link, $sql)){
    if(mysqli_num_rows($result) > 0){
        echo "<table>";
```

```
            echo "<tr>";
                echo "<th>id</th>";
                echo "<th>first_name</th>";
                echo "<th>address</th>";
                echo "<th>email</th>";
            echo "</tr>";
        while($row = mysqli_fetch_array($result)){
            echo "<tr>";
                echo "<td>" . $row['id'] . "</td>";
                echo "<td>" . $row['first_name'] .
            "</td>";
                echo "<td>" . $row['address'] . "</td>";
                echo "<td>" . $row['email'] . "</td>";
            echo "</tr>";
        }
        echo "</table>";

// Free result set
        mysqli_free_result($result);
}
        else{
            echo "No records matching your query were
        found.";
            }
}
else{
    echo "ERROR: Could not able to execute $sql. " .
    mysqli_error($link);
}
// Close connection
mysqli_close($link);
?>
```

MySQLi Functions	
mysqli_affected_rows()	Returns the number of affected rows in the previous MySQL operation.
mysqli_close()	Closes a previously opened database connection.
mysqli_connect_errno()	Returns the error code from the last connect call.
mysqli_connect_error()	Returns the error description from the last connection error.
mysqli_connect()	Opens a new connection to the MySQL server.
mysqli_error()	Returns the last error message for the most recent MySQLi function call.
mysqli_fetch_array()	Fetches a result row as an associative, a numeric array, or both.
mysqli_fetch_assoc()	Fetches a result row as an associative array.
mysqli_free_result()	Frees the memory associated with a result.
mysqli_num_rows()	Returns the number of rows in a result set.
mysqli_query()	Performs a query on the database.