Web Technology

Unit - 4
The Server Tier

The Server Tier

- **Web server** is a computer where the web content is stored. Basically web server is used to host the web sites but there exists other web servers also such as gaming, storage, FTP, email etc.
- Web site is collection of web pages while web server is a software that respond to the request for web resources.
- Every Website sits on a computer known as a Web server. Every Web server that is connected to the Internet is given a unique address made up of a series of four numbers between 0 and 256 separated by periods. For example, 68.178.157.132 or 68.122.35.127.
- When you register a Web address, also known as a domain name.
- A web server is server software, or hardware dedicated to running said software, that can satisfy World Wide Web client requests. A web server can, in general, contain one or more websites. A web server processes incoming ne-work requests over HTTP and several other related protocols.
- There are four leading web servers - Apache, IIS, lighttpd and Jagsaw.

The Server issues:

Load Limits

- A Web server has defined load limits. It can handle only a limited number of concurrent client connections and it can serve only a certain maximum number of requests per second depending on:
  - its own setting ,hardware and software limitation of OS .
  - HTTP request type ,content origin (static or dynamic) etc.
  - When a Web server is near to or over its limits, it becomes unresponsive**.**

Overload causes:

At any time web servers can be overloaded because of:

- Too much web traffic: when millions of clients connecting to the web site in a short interval overloaded can occur.
- Distributed Denial of Service attacks Computer worms that sometimes cause abnormal traffic because of millions of infected computers Web servers (computers) partial unavailability.
- This can happen because of required or urgent maintenance or upgrade, hardware or software failures, back-end (e.g., database) failures, etc.;

The Server issues:

- Distributed Denial of Service attacks.
- Computer worms that sometimes cause abnormal traffic because of millions of infected computers.
- Web servers (computers) partial unavailability. This can happen because of required or urgent maintenance or upgrade, hardware or software failures, back-end (e.g., database) failures, etc.; in these cases the remaining web servers get too much traffic and become overloaded.
- Internet connection slowdowns, so that client requests are served more slowly and the number of connections increases so much that server limits are reached.

The Server issues:

Anti overloaded technique:

- To partially overcome above load limits and to prevent overload, most Web sites use common techniques. They are as follows : (web server)
  - Managing network traffic, by using:
    - ✓ Firewalls to block unwanted traffic coming from bad IP sources,
    - ✓ HTTP traffic managers to drop, redirect or rewrite requests having bad HTTP patterns.
    - ✓ Bandwidth management and traffic shaping.
  - By deploying web cache techniques.
    - ✓ By adding more hardware resources (i.e. RAM, disks) to each computer.
    - ✓ By using different domain names to serve different (static and dynamic) content by separate Web servers etc.

The Server issues:

Apache HTTP Server :

- This is the most popular web server in the world developed by the Apache Software Foundation.
- Apache web server is an open source software and can be installed on almost all operating systems including Linux, Unix, Windows, FreeBSD, Mac OS X and more.
- We can have Apache with tomcat module to have JSP and J2EE related support.
- Internet Information Services
  - The Internet Information Server (IIS) is a high performance Web Server from Microsoft. This web server runs on Windows NT/2000/2/2/8/12 platforms.
- Lighttpd
  - The lighttpd, pronounced lighty is also a free web server that is distributed with the FreeBSD operating system. This open source web server is fast, secure and consumes much less CPU power.
  - Lighttpd can also run on Windows, Mac OS X, Linux and Solaris operating systems.

The Server issues:

Sun Java System Web Server

- This web server from Suns is suited for medium and large websites. Though the server is free it is not open source. It however, runs on Windows, Linux and Unix platforms. The Sun Java System web server supports various languages, scripts and technologies required for Web 2.0 such as JSP, Prenova Servlets, PHP, Perl, Python, Ruby on Rails, ASP and Coldfusion etc.

Jigsaw Server

- Jigsaw (W3C's Server) comes from the World Wide Web Consortium. It is open source and free and can run on various platforms like Linux, Unix, Windows, Mac OS X Free BSD etc. Jigsaw has been written in Java and can run CGI scripts and PHP programs.

Introduction to Dynamic Web Content

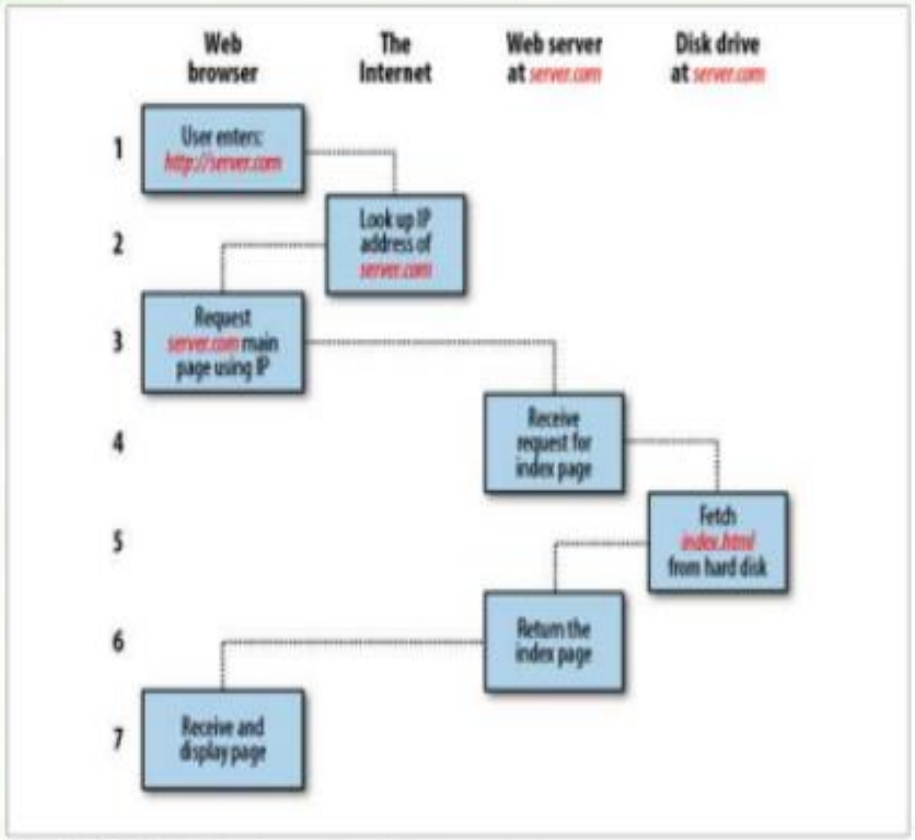


Figure 1-1. The basic client/server request/response sequence

1. You enter http://server.com into your browser's address bar.
2. Your browser looks up the IP address for server.com.
3. Your browser issues a request for the home page at server.com.
4. The request crosses the Internet and arrives at the server.com web server.
5. The web server, having received the request, looks for the web page on its hard disk.
6. The server retrieves the web page and returns it to the browser.
7. Your browser displays the web page.

The Server tier

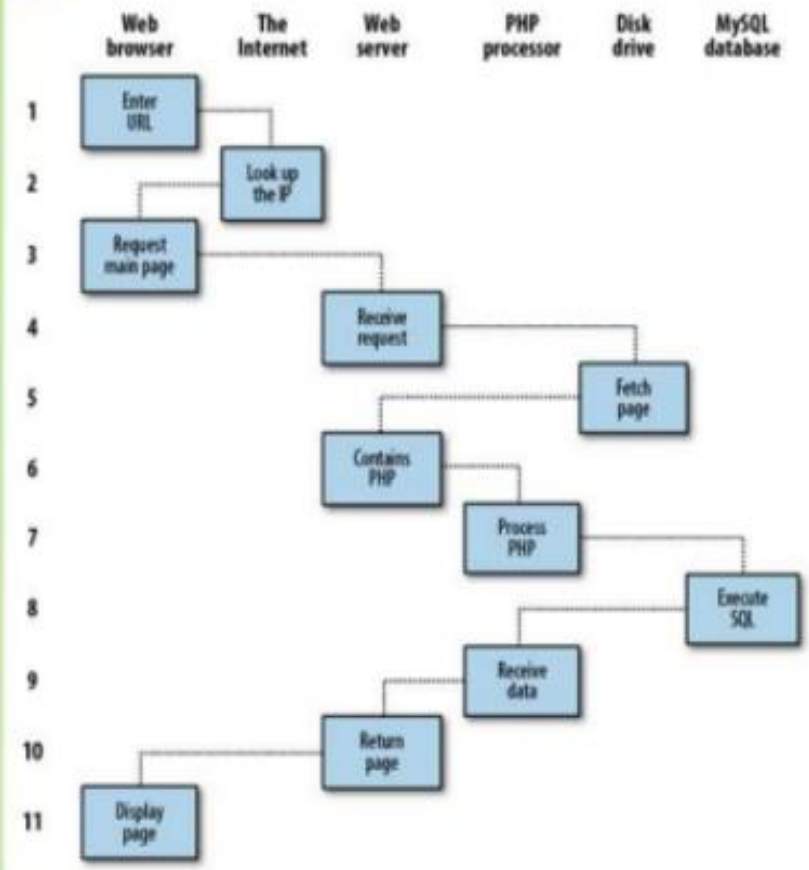


Figure 1-2. A dynamic client/server request/response sequence

1. You enter http://server.com into your browser's address bar.
2. Your browser looks up the IP address for server.com.
3. Your browser issues a request to that address for the web server's home page.
4. The request crosses the Internet and arrives at the server.com web server.
5. The web server, having received the request, fetches the home page from its hard disk.
6. With the home page now in memory, the web server notices that it is a file incorporating PHP scripting and passes the page to the PHP interpreter.
7. The PHP interpreter executes the PHP code.
8. Some of the PHP contains MySQL statements, which the PHP interpreter now passes to the MySQL database engine.
9. The MySQL database returns the results of the statements back to the PHP interpreter.
10. The PHP interpreter returns the results of the executed PHP code, along with the results from the MySQL database, to the web server.
11. The web server returns the page to the requesting client, which displays it.

PHP introduction

Creating a dynamic content

- We can create web content by using Server-side Scripting languages like ASP.Net,C#,PHP etc. Here we are going to discuss PHP Server side scripting.

PHP

- The PHP Hypertext Preprocessor (PHP) is a programming language that allows web developers to create dynamic content that interacts with databases. PHP is basically used for developing web based software applications.
- PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages.
- PHP is a widely-used, free, and efficient alternative to competitors such as Microsoft's ASP.
- PHP 7 is the latest stable release. This tutorial uses PHP 7.2.10
- It is powerful enough to be at the core of the biggest blogging system on the web (WordPress)!
- It is deep enough to run the largest social network (Facebook)
- It is also easy enough to be a beginner's first server side language!

PHP introduction

Why PHP?

- PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP supports a wide range of databases
- PHP is free. Download it from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side.

What Can PHP Do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data
- With PHP you are not limited to output HTML. You can output images, PDF files, and even Flash movies. You can also output any text, such as XHTML and XML.

PHP introduction

Advantages of PHP over Other Languages

If you're familiar with other server-side languages like ASP.NET or Java, you might be wondering what makes PHP so special. There are several advantages why one should choose PHP.

Easy to learn: PHP is easy to learn and use. For beginner programmers who just started out in web development, PHP is often considered as the preferable choice of language to learn.

Open source: PHP is an open-source project. It is developed and maintained by a worldwide community of developers who make its source code freely available to download and use.

Portability: PHP runs on various platforms such as Microsoft Windows, Linux, Mac OS, etc. and it is compatible with almost all servers used today such Apache, IIS, etc.

Fast Performance: Scripts written in PHP usually execute or runs faster than those written in other scripting languages like ASP, Ruby, Python, Java, etc.

Vast Community: Since PHP is supported by the worldwide community, finding help or documentation related to PHP online is extremely easy.

You know, some huge websites like Facebook, Yahoo, Flickr, and Wikipedia are built using PHP. Most of the major content management systems (CMS), such as WordPress, Drupal, Joomla and Magento are also built in PHP.

PHP introduction

Difference between client-side scripting vs. Server side scripting

Client Side Scripting	Server Side Scripting
The client-side environment used to run scripts is usually a browser.	The **server-side environment** that runs a scripting language is a web server.
The source code is transferred from the web server to the user's computer over the internet and run directly in the browser.	A user's request is fulfilled by running a script directly on the web server to generate dynamic HTML pages. This HTML is then sent to the client browser.
Advantages to client-side scripting including faster response times, a more interactive application, and less overhead on the web server.	The primary advantage to server-side scripting is the ability to highly customize the response based on the user's requirements, access rights, or queries into data stores.
The Disadvantages of client-side scripting are that scripting languages require more time and effort, while the client's browser must support that scripting language.	The disadvantage of server-side processing is the page postback: it can introduce processing overhead that can decrease performance and force the user to wait for the page to be processed and recreated. Once the page is posted back to the server, the client must wait for the server to process the request and send the page back to the client.

Example
`<script>`
`document.getElementById('hello').innerHTML = 'Hello';`
`</script>`
	Example:
	`<h1 id="hello"><?php echo 'Hello'; ?></h1>`

PHP introduction

Basic Syntax

<?php
// PHP code goes here
?>

```
<!DOCTYPE html>
<html>
<body>
<h1>My first PHP page</h1>
<?php
echo "Hello World!";
?>
</body>
</html>
```

Variable names in PHP are case-sensitive
But Keyword is not.

```
<!DOCTYPE html>
<html>
<body>

<?php
// This is a single-line comment

# This is also a single-line comment

/*
This is a multiple-lines comment block
that spans over multiple
lines
*/

// You can also use comments to leave out parts of a code line
$x = 5 /* + 15 */ + 5;
echo $x;
?>

</body>
</html>
```

Comment example

$a,$A are different
For, for are same.
Creating (Declaring) PHP Variables

```
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
```

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)

Date functions

The PHP Date() Function

- The PHP date() function convert a timestamp to a more readable date and time.

date(*format, timestamp*)

The format parameter in the date() function is required which specifies the format of returned date and time. However the timestamp is an optional parameter, if not included then current date and time will be used.

```php
<?php
$today = date("d/m/Y");
echo $today;
?>
```

03/08/2019

```php
<?php
echo date("d/m/Y") . "<br>";
echo date("d-m-Y") . "<br>";
echo date("d.m.Y");
?>
```

03/08/2019
03-08-2019
03.08.2019

d - Represent day of the month; two digits with leading zeros (01 or 31)
D - Represent day of the week in text as an abbreviation (Mon to Sun)
m - Represent month in numbers with leading zeros (01 or 12)
M - Represent month in text, abbreviated (Jan to Dec)
y - Represent year in two digits (08 or 14)
Y - Represent year in four digits (2008 or 2014)

Date functions

h - Represent hour in 12-hour format with leading zeros (01 to 12)
H - Represent hour in in 24-hour format with leading zeros (00 to 23)
i - Represent minutes with leading zeros (00 to 59)
s - Represent seconds with leading zeros (00 to 59)
a - Represent lowercase ante meridiem and post meridiem (am or pm)
A - Represent uppercase Ante meridiem and Post meridiem (AM or PM)

```php
<?php
echo date("h:i:s") . "<br>";
echo date("F d, Y h:i:s A") . "<br>";
echo date("h:i a");
?>
```

12:08:25
August 03, 2019 12:08:25 PM
12:08 pm

```php
<?php
$timestamp = time();
echo $timestamp."<br>";
echo(date("F d, Y h:i:s", $timestamp));
?>
```

1564834299
August 03, 2019 02:11:39

he mktime() function is used to create the timestamp based on a specific date and time. If no date and time is provided, the timestamp for the current date and time is returned.

```
mktime(hour, minute, second, month, day, year)
```

```php
<?php
// Create the timestamp for a particular date
echo mktime(15, 20, 12, 5, 10, 2014);
?>
```

1399735212

PHP Comments

- In PHP, we use **//** to make a single-line comment or **/*** and ***/** to make a large comment block.

```
<html>
<body>

<?php
//This is a comment

/*
This is
a comment
block
*/
?>

</body>
</html>
```

PHP Variables

- > Variables are used for storing values, like text strings, numbers or arrays.
- > When a variable is declared, it can be used over and over again in your script.
- > All variables in PHP start with a $ sign symbol.
- > The correct way of declaring a variable in PHP:

```
$var_name = value;
```

PHP Variables

```
<?php
$txt="Hello World!";
$x=16;
?>
```

- > In PHP, a variable does not need to be declared before adding a value to it.
- > In the example above, you see that you do not have to tell PHP which data type the variable is.
- > PHP automatically converts the variable to the correct data type, depending on its value.

PHP Variables

- > A variable name must start with a letter or an underscore "_" -- not a number
- > A variable name can only contain alpha-numeric characters, underscores (a-z, A-Z, 0-9, and _)
- > A variable name should not contain spaces. If a variable name is more than one word, it should be separated with an underscore ($my_string) or with capitalization ($myString)

PHP Concatenation

- > The concatenation operator (.) is used to put two string values together.
- > To concatenate two string variables together, use the concatenation operator:

```
<?php
$txt1="Hello World!";
$txt2="What a nice day!";
echo $txt1 . " " . $txt2;
?>
```

PHP Concatenation

- The output of the code on the last slide will be:

```
Hello World! What a nice day!
```

- If we look at the code you see that we used the concatenation operator two times. This is because we had to insert a third string (a space character), to separate the two strings.

PHP Operators

•Operators are used to operate on values. There are four classifications of operators:

- > Arithmetic
- > Assignment
- > Comparison
- > Logical

PHP Operators

Arithmetic Operators

Operator	Description	Example	Result
+	Addition	x=2 x+2	4
-	Subtraction	x=2 5-x	3
*	Multiplication	x=4 x*5	20
/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

PHP Operators

Assignment Operators

Operator	Example	Is The Same As
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
.=	x.=y	x=x.y
%=	x%=y	x=x%y

PHP Operators

Comparison Operators

Operator	Description	Example
==	is equal to	5==8 returns false
!=	is not equal	5!=8 returns true
<>	is not equal	5<>8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

PHP Operators

Logical Operators

Operator	Description	Example
&&	and	x=6 y=3 (x < 10 && y > 1) returns true
||	or	x=6 y=3 (x==5 || y==5) returns false
!	not	x=6 y=3 !(x==y) returns true

PHP Conditional Statements

- > Very often when you write code, you want to perform different actions for different decisions.
- > You can use conditional statements in your code to do this.
- > In PHP we have the following conditional statements...

PHP Conditional Statements

- **if** statement - use this statement to execute some code only if a specified condition is true
- **if...else** statement - use this statement to execute some code if a condition is true and another code if the condition is false
- **if...elseif....else** statement - use this statement to select one of several blocks of code to be executed
- **switch** statement - use this statement to select one of many blocks of code to be executed

PHP Conditional Statements

- The following example will output "Have a nice weekend!" if the current day is Friday:

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri") echo "Have a nice weekend!";
?>

</body>
</html>
```

PHP Conditional Statements

- Use the **if....else** statement to execute some code if a condition is true and another code if a condition is false.

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri")
  echo "Have a nice weekend!";
else
  echo "Have a nice day!";
?>

</body>
</html>
```

PHP Conditional Statements

- If more than one line should be executed if a condition is true/false, the lines should be enclosed within curly braces **{ }**

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri")
  {
  echo "Hello!<br />";
  echo "Have a nice weekend!";
  echo "See you on Monday!";
  }
?>

</body>
</html>
```

PHP Conditional Statements

- The following example will output "Have a nice weekend!" if the current day is Friday, and "Have a nice Sunday!" if the current day is Sunday. Otherwise it will output "Have a nice day!":

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri")
  echo "Have a nice weekend!";
elseif ($d=="Sun")
  echo "Have a nice Sunday!";
else
  echo "Have a nice day!";
?>

</body>
</html>
```

PHP Conditional Statements

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

```
switch (n)
{
case label1:
  code to be executed if n=label1;
  break;
case label2:
  code to be executed if n=label2;
  break;
default:
  code to be executed if n is different from both label1 and label2;
}
```

PHP Conditional Statements

- For switches, first we have a single expression n (most often a variable), that is evaluated once.

- The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed.

- Use break to prevent the code from running into the next case automatically. The default statement is used if no match is found.

PHP Conditional Statements

```
<html>
<body>

<?php
switch ($x)
{
case 1:
  echo "Number 1";
  break;
case 2:
  echo "Number 2";
  break;
case 3:
  echo "Number 3";
  break;
default:
  echo "No number between 1 and 3";
}
?>

</body>
</html>
```

PHP Arrays

- > An array variable is a storage area holding a number or text. The problem is, a variable will hold only one value.
- > An array is a special variable, which can store multiple values in one single variable.

PHP Arrays

- If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
$cars1="Saab";
$cars2="Volvo";
$cars3="BMW";
```

PHP Arrays

- > However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?
- > The best solution here is to use an array.
- > An array can hold all your variable values under a single name. And you can access the values by referring to the array name.
- > Each element in the array has its own index so that it can be easily accessed.

PHP Arrays

- In PHP, there are three kind of arrays:
- **> Numeric array** - An array with a numeric index
- **> Associative array** - An array where each ID key is associated with a value
- **> Multidimensional array** - An array containing one or more arrays

PHP Numeric Arrays

- > A numeric array stores each array element with a numeric index.
- > There are two methods to create a numeric array.

PHP Numeric Arrays

- In the following example the index is automatically assigned (the index starts at 0):

```
$cars=array("Saab","Volvo","BMW","Toyota");
```

- In the following example we assign the index manually:

```
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
```

PHP Numeric Arrays

- In the following example you access the variable values by referring to the array name and index:

```
<?php
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
echo $cars[0] . " and " . $cars[1] . " are Swedish cars.";
?>
```

- The code above will output:

```
Saab and Volvo are Swedish cars.
```

PHP Associative Arrays

- > With an associative array, each ID key is associated with a value.
- > When storing data about specific named values, a numerical array is not always the best way to do it.
- > With associative arrays we can use the values as keys and assign values to them.

PHP Associative Arrays

- In this example we use an array to assign ages to the different persons:

```
$ages = array("Peter"=>32, "Quagmire"=>30, "Joe"=>34);
```

- This example is the same as the one above, but shows a different way of creating the array:

```
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";
```

PHP Associative Arrays

The ID keys can be used in a script:

```
<?php
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";

echo "Peter is " . $ages['Peter'] . " years old.";
?>
```

The code above will output:

```
Peter is 32 years old.
```

PHP Multidimensional Arrays

- In a multidimensional array, each element in the main array can also be an array.

- And each element in the sub-array can be an array, and so on.

PHP Multidimensional Arrays

In this example we create a multidimensional array, with automatically assigned ID keys:

```
$families = array
  (
  "Griffin"=>array
  (
  "Peter",
  "Lois",
  "Megan"
  ),
  "Quagmire"=>array
  (
  "Glenn"
  ),
  "Brown"=>array
  (
  "Cleveland",
  "Loretta",
  "Junior"
  )
  );
```

PHP Multidimensional Arrays

The array above would look like this if written to the output:

```
Array
(
[Griffin] => Array
   (
   [0] => Peter
   [1] => Lois
   [2] => Megan
   )
[Quagmire] => Array
   (
   [0] => Glenn
   )
[Brown] => Array
   (
   [0] => Cleveland
   [1] => Loretta
   [2] => Junior
   )
)
```

PHP Multidimensional Arrays

Lets try displaying a single value from the array above:

```
echo "Is " . $families['Griffin'][2] .
" a part of the Griffin family?";
```

The code above will output:

```
Is Megan a part of the Griffin family?
```

PHP Loops

- > Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal lines in a script we can use loops to perform a task like this.
- > In PHP, we have the following looping statements:

PHP Loops

- **while** - loops through a block of code while a specified condition is true
- **do...while** - loops through a block of code once, and then repeats the loop as long as a 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

PHP Loops - While

- The while loop executes a block of code while a condition is true. The example below defines a loop that starts with
- i=1. The loop will
- continue to run as
- long as i is less
- than, or equal to 5.
- i will increase by 1
- each time the loop
- runs:

```
<html>
<body>

<?php
$i=1;
while($i<=5)
  {
  echo "The number is " . $i . "<br />";
  $i++;
  }
?>

</body>
</html>
```

PHP Loops - While

Output:

```
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

PHP Loops – Do ... While

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

- The next example defines a loop that starts with i=1. It will then increment i with 1, and write some output. Then the condition is checked, and the loop will continue to run as long as i is less than, or equal to 5:

PHP Loops – Do ... While

```
<html>
<body>

<?php
$i=1;
do
  {
  $i++;
  echo "The number is " . $i . "<br />";
  }
while ($i<=5);
?>

</body>
</html>
```

PHP Loops – Do ... While

Output:

```
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
```

PHP Loops - For

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

Syntax

```
for (init; condition; increment)
  {
  code to be executed;
  }
```

PHP Loops - For

- Parameters:
- **> init**: Mostly used to set a counter (but can be any code to be executed once at the beginning of the loop)
- **> condition**: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- **> increment**: Mostly used to increment a counter (but can be any code to be executed at the end of the loop)

PHP Loops - For

- The example below defines a loop that starts with i=1. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs:

```
<html>
<body>

<?php
for ($i=1; $i<=5; $i++)
  {
  echo "The number is " . $i . "<br />";
  }
?>

</body>
</html>
```

PHP Loops - For

Output:

```
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

PHP Loops - Foreach

```
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) - so on the next loop iteration, you'll be looking at the next array value.

PHP Loops - Foreach

- The following example demonstrates a loop that will print the values of the given array:

```
<html>
<body>

<?php
$x=array("one","two","three");
foreach ($x as $value)
  {
  echo $value . "<br />";
  }
?>

</body>
</html>
```

Output:

```
one
two
three
```

PHP Functions

- > We will now explore how to create your own functions.
- > To keep the script from being executed when the page loads, you can put it into a function.
- > A function will be executed by a call to the function.
- > You may call a function from anywhere within a page.

PHP Functions

- A function will be executed by a call to the function.

```
function functionName()
{
code to be executed;
}
```

- > Give the function a name that reflects what the function does
- > The function name can start with a letter or underscore (not a number)

PHP Functions - Parameters

The following example will write different first names, but equal last name:

```
<html>
<body>

<?php
function writeName($fname)
{
echo $fname . " Refsnes.<br />";
}

echo "My name is ";
writeName("Kai Jim");
echo "My sister's name is ";
writeName("Hege");
echo "My brother's name is ";
writeName("Stale");
?>

</body>
</html>
```

PHP Functions

- A simple function that writes a name when it is called:

```
<html>
<body>

<?php
function writeName()
{
echo "Kai Jim Refsnes";
}

echo "My name is ";
writeName();
?>

</body>
</html>
```

PHP Functions - Parameters

- Adding parameters...
- > To add more functionality to a function, we can add parameters. A parameter is just like a variable.
- > Parameters are specified after the function name, inside the parentheses.

PHP Functions - Parameters

Output:

```
My name is Kai Jim Refsnes.
My sister's name is Hege Refsnes.
My brother's name is Stale Refsnes.
```

PHP Functions - Parameters

```
<html>
<body>

<?php
function writeName($fname,$punctuation)
{
echo $fname . " Refsnes" . $punctuation . "<br />";
}

echo "My name is ";
writeName("Kai Jim",".");
echo "My sister's name is ";
writeName("Hege","!");
echo "My brother's name is ";
writeName("Ståle","?");
?>

</body>
</html>
```

This example adds different punctuation.

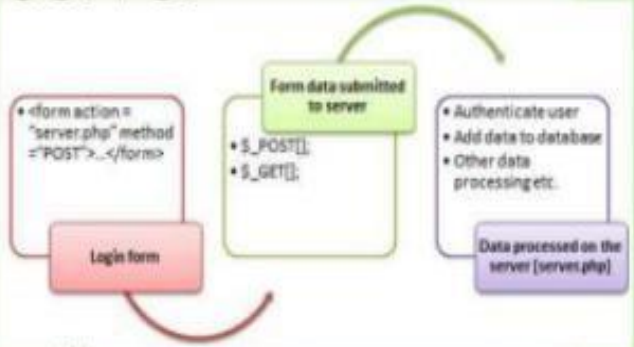
PHP Functions - Parameters

Output:

```
My name is Kai Jim Refsnes.
My sister's name is Hege Refsnes!
My brother's name is Ståle Refsnes?
```

Methods of Sending Information to Server

A web browser communicates with the server typically using one of the two HTTP (Hypertext Transfer Protocol) methods; GET,POST,REQUEST. In GET method the data is sent as URL parameters that are usually strings of name and value pairs separated by ampersands (&).



http://www.mechicampus.edu.np/action.php?name=mohan&age=24

- Since the data sent by the GET method are displayed in the URL, it is possible to bookmark the page with specific query string values.
- The GET method is not suitable for passing sensitive information such as the username and password, because these are fully visible in the URL query string as well as potentially stored in the client browser's memory as a visited page.
- Because the GET method assigns data to a server environment variable, the length of the URL is limited. So, there is a limitation for the total data to be sent.
- $REQUEST contains: $COOKIE, $GET, and $POST variables.

Methods of Sending Information to Server

form method="post" action="action.php">
data goes here .....
</form>
Now , In this case action.php is destination page where user will have records .
in PHP : there are 3 methods to retrieve data from receiving page.
1) $_GET
2) $_POST
3) R_REQUEST
$_GET :
$_GET is a super global array which is an inbuilt array. Which collects values from a form sent with method="get" as well it collects values from URL also. (e.g. http://www.mmc.com/index.php?data=123)
You can print $_GET with inbuilt function in php i.e (print_r($_GET)).
IMP : Information sent from this method will be visible in (address bar of any browser) and has limitation on characters .
As per W3C Community limitation can be extended up to 4000 characters
Myth: Search services will not index anything with a "?" in the URI.
Myth: URIs cannot be longer than 256 characters
for details : visit : About $_GET limitation :
You can read more details about $_GET.
$_POST:
$_POST is a super global array which is an inbuilt array. Which collects values from a form sent with method="post".
You can print $_GET with inbuilt function in php i.e (print_r($_POST)).
IMP : Information received from POST method is always invisible in (address bar of any browser) . It has more limit on sending characters to action page.
This limitation can be set in php.ini file
Note: We can change by setting the POST_MAX_SIZE in the php.ini file . by default we will find this value with 8MB in most of the php.ini file.
$_REQUEST:
$_REQUEST is a super global array which is an inbuilt array. Which collects values from a form sent with either method="post" or method="get".
You can print $_REQUEST with inbuilt function in php i.e (print_r($_REQUEST)).
IMP : One of the best way of getting data on action page . here we did not require to check that which method we have used to pass data from sending page. Even $_REQUEST get records from the COOKIE also.

GET vs POST

	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data. Use multipart encoding for binary data
History	Parameters remain in browser history	Parameters are not saved in browser history
Restrictions on data length	Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters)	No restrictions
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Security	GET is less secure compared to POST because data sent is part of the URL Never use GET when sending passwords or other sensitive information!	POST is a little safer than GET because the parameters are not stored in browser history or in web server logs
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL

PHP Include and Require Files

- The include() and require() statement allow us to include the code contained in a PHP file within another PHP file.
- Including a file produces the same result as copying the script from the file specified and pasted into the location where it is called.
- You can save a lot of time and work through including files —Just store a block of code in a separate file and include it wherever you want using the include() and require() statements instead of typing the entire block of code multiple times.
- A typical example is including the header, footer and menu file within all the pages of a website.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Tutorial Republic</title>
</head>
<body>
<?php include "header.php"; ?>
<?php include "menu.php"; ?>
    <h1>Welcome to Our Website!</h1>
    <p>Here you will find lots of useful information.</p>
<?php include "footer.php"; ?>
</body>
</html>
```

PHP Include and Require Files

Difference Between include and require Statements

- The only difference is — the include() statement will only generate a PHP warning but allow script execution to continue if the file to be included can't be found, whereas the require() statement will generate a fatal error and stops the script execution.
- **Tip**: It is recommended to use the require() statement if you're including the library files or files containing the functions and configuration variables that are essential for running your application, such as database configuration file.

- The **include_once** and **require_once** statements will only include the file once even if asked to include it a second time i.e. if the specified file has already been included in a previous statement, the file is not included again.

Session and State

- Web is Stateless, that means the web server does not know who you are and what you do, because the HTTP address doesn't maintain state.
- If user inserts some information, and move to the next page, that data will be lost and user would not able to retrieve the information so we need to store the information about user.
- Session provides that facility to store information on server memory. Session variables hold information about one single user, and are available to all pages in one application. By default, session variables last until the user closes the browser or destroy the session variable in php code.
- A PHP session stores data on the server rather than user's computer. In a session based environment, every user is identified through a unique number called **session identifier** or SID. This unique session ID is used to link each user with their own information on the server like emails, posts, etc.

Session and State

Advantages

- It helps to maintain user states and data to all over the application.
- It can easily be implemented and we can store any kind of object.
- Stores every client data separately.
- Session is secure and transparent from user.

Disadvantages

- Performance overhead in case of large volume of user, because of session data stored in server memory.
- Overhead involved in serializing and De-Serializing session Data. because In case of StateServer and SQLServer session mode we need to serialize the object before store.

Session and State

```php
<?php
session_start();
?>
<html>
<body>
<?php
 $_SESSION["user"] = "krish";
 $_SESSION["pass"] = session_id();
  echo "Session information are successfully set.<br/>";
  echo "<a href='session2.php'>Visit next page</a>"; }
?>
</body>
</html>
```

```php
<?php
session_start();
?>
<html>
<body>
<a href="sessionout.php">Clear Session</a> <br>
<?php
echo "User is: ".$_SESSION["user"].
    "<br>Password: ".$_SESSION["pass"]."<br>";
?>
</body>
</html>
```

```php
<?php
session_start();
//session_destroy();
//Is Used To Destroy All Sessions

if(isset($_SESSION['user'])) //specific
{
  unset($_SESSION['user']);
  unset($_SESSION['pass']);
}
  echo "Clear Session successfully!";
 ?>
```

Handling Errors

- An Error is said to have occurred when a piece of code returns unexpected result or stops abruptly due to some incorrect code for example division by zero, or infinite loop etc.
- Sometimes an application will not run as it supposed to do, resulting in an error. There are a number of reasons that may cause errors, for example:
- The Web server might run out of disk space.
- A user might have entered an invalid value in a form field.
- The file or database record that you were trying to access may not exist.
- The application might not have permission to write to a file on the disk.
- A service that the application needs to access might be temporarily unavailable.
- These types of errors are known as runtime errors, because they occur at the time the script runs. They are distinct from syntax errors that need to be fixed before the script will run.
- A professional application must have the capabilities to handle such runtime error gracefully. Usually this means informing the user about the problem more clearly and precisely..

Handling Errors

- **Error Types in php:**

- **Notice:** These are small, non-critical errors that PHP encounters while executing a script that don't stop PHP from executing a script. for example,: accessing a variable that has not yet been defined
- **Warnings:** serious errors that require attention but still don't stop script to execute.eg: reading a file that doesn't exist in given path ,division by zero.
- **Fatal errors:** syntax error or critical errors that stop execution .example :calling a non-existent function, instantiating an object of a non-existent class.

- **Error report level:** These error report levels are the different types of error the user-defined error handler can be used for:

E_WARNING,E_NOTICE,E_USER_ERROR,E_USER_WARNING, E_USER_NOTICE, ERECOVERABLEERROR, E_ALL

Handling Errors

- **In PHP,** We can set PHP configurations to show error messages in the browser or hide them.
- PHP provides multiple ways to handle errors

- Using die() function
- conditional statements
- Using Custom Error Handlers
- PHP error reporting

- **die() function:** The die() function in PHP is used to display any message and exit out of current script at the same time. Hence once an error condition is detected, die() function can be used to exit the code execution.

```php
<?php

    $fileName = "noerror.txt";
    // opening the file
    fopen($fileName, "r")
    or die("Unable to open file $fileName");

?>
```

```php
<?php

    function division($numerator, $denominator) {
        if($denominator != 0) {
            echo $numerator / $denominator;
        }
        else {
            echo "Division by zero(0) no allowed!";
        }
    }

    division(7, 0);
?>
```

Handling Errors

- Handling Errors Using Custom Error Handlers: PHP provide custom method to display any message or execute any code when error occurs. All we have to do is set our method as the default error handler for PHP using the function set_error_handler()

```php
<?php
function error_handler_test($error_no,$error_msg)
{
    echo "OOPs Something Unexpected happen..";
    echo "Possible reason:".$error_no.$error_msg;
    echo "we are working on it";
    return true;
}
set_error_handler("error_handler_test");
echo $r;
?>
```

Handling Errors

- Error reporting PHP provides default error reporting mechanism which can be utilized to display messaged on screen when an error occurs. We can use PHP function error_reporting() to set which errors to show and which errors to hide.

```php
<?php
// Turn on error reporting for all types of errors
error_reporting(E_ALL);


echo $undefinedVariable;

// Example 2: E_WARNING
$file = 'nonexistent_file.txt';
$fileHandle = fopen($file, 'r'); // This will generate a warning because the file does not exist

// Example 3: E_ERROR
function divide($a, $b) {
    return $a / $b; // This will cause a division by zero error if $b is 0
}

echo divide(10, 0); // This will generate a fatal error (division by zero)
?>
```

Handling Errors

- **Exception in php:** An exception is a signal that indicates some sort of exceptional event or error has occurred. Exceptions can be caused due to various reasons, for example, database connection or query fails, file that you're trying to access doesn't exist. exception handling is the object-oriented method for handling errors, which provides more controlled and flexible form of error reporting

```
<?php    try {
            // Code that may throw an exception
            $result = 10 / 0;
         } catch (Exception $e) {
            // Catching and handling the exception
            echo "Caught exception: " . $e->getMessage();
         } finally {
            // Optional: Code that will be executed regardless of an exception
            echo "Finally block executed.";
         }
?>
```

Tag libraries:

- In a Web application, a common design goal is to separate the display code from business logic. Java tag libraries are one solution to this problem. Tag libraries allow you to isolate business logic from the display code by creating a Tag class (which performs the business logic) and including an HTML-like tag in your JSP page.
- When the Web server encounters the tag within your JSP page, the Web server will call methods within the corresponding Java Tag class to produce the required HTML content.
- The java Server page API allows us to define custom JSP tags that look like HTML or XML tags and tag library is a set of user defined tags that implement custom behaviour. The taglib directive declares that your jsp page uses a set of custom tags, identifies the location of the library,and provides a means for identifying the custom tags in your JSP page.

Tag libraries:

- **syntax of taglib directive:**

<%@taglib prefix="prefixOfTag" uri="uri"%>

- Where the uri attribute value resolves to a location the container understands and the prefix attribute informs a container what bits of markup are custom actions.
- Creating custom tag:
- A custom tag is a user-defined JSP language element. When a JSP page containing a custom tag is translated into a servlet, the tag is converted to operations on an object called a tag handler. The Web container then invokes those operations when the JSP page's servlet is executed.JSP tag extensions let you create new tags that you can insert directly into a JavaServer Page just as you would the built-in tags.
- The JSP 2.0 specification introduced Simple Tag Handlers for writing these custom tags.
- To write a custom tag you can simply extend SimpleTagSupport class and override the doTag() method,

Tag libraries:

- where you can place your code to generate content for the tag.
- Consider you want to define a custom tag named <ex:Hello> and you want to use it in the following fashion without a body: <ex:Hello />
- To create a custom JSP tag, you must first create a Java class that acts as a tag handler. So let us create HelloTag class as follows

```
package com.customTagExample;
import javax.servlet.jsp.tagext.*;
import javax.servlet.jsp.*;
import java.io.*;
public class HelloTag extends SimpleTagSupport {
public void doTag() throws JspException, IOException
{
JspWriter out = getJspContext().getOut();
out.println("Hello Custom Tag!");
}
}
```

Above code has simple coding where doTag() method takes the current JspContext object using getJspContext() method and uses it to send "Hello Custom Tag!" to the current JspWriter object.

Let us compile above class and copy it in a directory available in environment variable CLASSPATH. Finally create following tag library file: <Tomcat-Installation-Directory>webapps\ROOT\WEB-INF\custom.tld.

```
<taglib>
<tlib-version>1.0</tlib-version>
<jsp-version>2.0</jsp-version>
<short-name>Example TLD</short-name>
<tag>
<name>Hello</name>
<tag-class>com.customTagExample.HelloTag</tag-class>
<body-content>empty</body-content>
</tag>
</taglib>
```

Now it's time to use above defined custom tag Hello in our JSP program as follows:

```
<%@ taglib prefix="ex" uri="WEB-INF/custom.tld"%>
<html>
<head>
<title>A sample custom tag</title>
</head>
```

```
<body>
<ex:Hello/>
</body>
</html>
```

Try to call above JSP and this should produce following result:

Hello Custom Tag!

Accessing the Tag Body:

You can include a message in the body of the tag as you have seen with standard tags. Consider you want to define a custom tag named <ex:Hello> and you want to use it in the following fashion with a body:

<ex:Hello>

This is message body

</ex:Hello>