The Server Tier I

Web server

Web server is a computer system which receives the HTTP requests via TCP, which are used to distribute information on world wide web. Commonly a web browser or web crawler initiates communication by making HTTP request for a specific resource and the server response with the content of that resource or an error message if unable to do so. The main function of a web server is to store, process and deliver web pages to clients. The communication between client and server takes place using the Hypertext Transfer Protocol (HTTP). Pages delivered are most frequently HTML documents, which may include images,stylesheets, and scripts in addition to text content.

Web servers are not always used for serving the world wide web. They can also be found embedded in devices such as printers,routers, webcams and serving only a local network

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 cause:

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.; 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

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.

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 Serverside scripting.

PHP

PHP stands for Hypertext Preprocessor. It is a server-side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.

Simple example of PHP:

```
<html>
<head>
<title>Hello World</title>
</head>
<body>
<?php
echo " Today is " . date("l") . ". ";
?>
</body>
</html>
```

- d - Represents the day of the month (01 to 31)
- m - Represents a month (01 to 12)
- Y - Represents a year (in four digits)
- l (lowercase 'L') - Represents the day of the week

https://www.w3schools.com/php/php_date.asp

As mentioned earlier, PHP is embedded in HTML.

Variables in PHP

All variables in PHP always begins with the dollar sign ($) and are followed by a concise, meaningful name.PHP has a total of eight data types which we use to construct variables. They are:

- Integers
- Doubles
- Booleans
- NULL
- Strings
- Arrays
- Objects
- Resources

The first five are simple types and other are compound types.
Simple example of declaring variables in PHP:

```
$ int_var=12345;
$ name="ronaldo";
$my_var=NULL;
```

etc

Rules for naming variables:

- Variable names must begin with a letter or underscore character.
- A variable name can consist of numbers, letters, underscores but you cannot use characters like + , - , % , (,) . & , etc

Comment in PHP

A comment is the portion of a program that exists only for the human reader.there are two types of comment in PHP,Single line comment and multi line comment.

single line comment: they are used for short explanation.

multiline comment:They are generally used to provide detailed explanations when necessary.Here is the example of comment

```
<?
# This is a single line comment
// This is a single line comment too. Each style comments only
/* This is a
Example of
Multiline
comment
*/
echo "comments are not displayed"
?>
```

Using control flow to control dynamic content generation

we can use different technique to control flow for the dynamic content. Some of them are defined below:

For loop :

It is used when you know how many times you want to execute a statement or block of statement.

syntax:

```
for(start value, condition, increment/decrement){
//statements
}
```

While loop

The while statement will execute a block of code if and as long as a test expression is true.

syntax:

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

do while loop

The do...while statement will execute a block of code at least once. it then will repeat the loop as long as a condition is true.

Syntax:

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

for each loop

The foreach statement is used to loop through arrays. For each pass the value of the current array element is assigned to $value and the array pointer is moved by one and in the next pass next element will be processed.

Syntax

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

Switch cases

it evaluates given expression then search a lable to match the resulting value. If a matching value is found then the code associated with the matching label will be executed or if none of the lable matches then statement will execute any specified default code.

syntax:

```
switch (expression){
case1:
//code to be executed if expression is matched with case 1;
break;
case 2:
// code to be executed if expression is matched with case2;
break;
.....
....
default:
//code to be executed if none of the case is matched with expression
}
```

If else statement

If you want to execute some code if a condition is true and another code if a condition is false, use the if....else statement.

Syntax

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

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.

Creating a Session

Simply by calling the session_start() function you can create a new session. Session variables are set with the PHP global variable: $_SESSION.

syntax :

```
session_start();
```

This is the initialization of session. Each and every page at the top of script, we must initiate the session,if we need any session value.

syntax:

```
$_SESSION['session_name]="value";
```

example:

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// Set session variables
$_SESSION["name"] = "rabin";
$_SESSION["address"] = "kathmandu";
?>
</body>
</html>
```

Retrieving the value from session

we can retrieve the value in any page if we store the value in sessio,session carries the values from one page to another.

syntax:

```
$_SESSION['session_name']
```

example:

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// Echo session variables that were set on previous page
echo "hello " . $_SESSION["name"] . ".<br>";
echo "your address is " . $_SESSION["adderss"] .;
?>
</body>
</html>
```

Removing session and session variables

we can also destroy our session after we finished its requirement.to destroy a specific session variable simply unset the corresponding key of the $_SESSION array.Use session_distroy() function to destroy all the session variables. If you want to destroy a single session variable then useunset() function.

syntax:

```
session_destroy();//destroy all the session
unset($_SESSION['session_name']);//unset single variable
```

example:

```
<?php
session_start();
```

```
?>
<!DOCTYPE html>
<html>
<body>
<?php
// remove name variables
unset($_SESSION['name']);
// destroy the session
session_destroy();
?>
</body>
</html>
```

Advantages and disadvantages of session

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

The Server Tier II

Error Handling

The default error handling in PHP is very simple. An error message with filename line number and a message describing the error is sent to the browser.

When creating scripts and web applications, error handling is an important part. If your code lacks error checking code, your program may look very unprofessional and you may be open to security risks.

In PHP there are different error handling methods they are :

- Simple "die()" statements
- Custom errors and error triggers
- Error reporting

Using die() function

if the required operation cant be performed then we use die() statement to handle these type of error.If the file does not exist you might get an error like this:

Warning: fopen(welcome.txt) [function.fopen]: failed to open stream:

No such file or directory in C:\webfolder\hello.php on line 2

To prevent the user from getting an error message like the one above, we handle these type of error like this:

```
<?php
```

```
if(!file_exists("welcome.txt")) {
die("File not found");
} else {
$file=fopen("welcome.txt","r");
}
?>
```

now, if the file name "welcome.txt" does not exist we get an error like this :

File not found.

simply, stopping the scripting is not always the right way to go ,lets take a look at another handling error technique.

Custom errors

Creating a custom error handler is quite simple. We simply 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 and error message) but can accept up to five parameters (optionally: file, line-number, and the error context):

Syntax:

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

where

- **error_level :-**Required. Specifies the error report level for the user-defined error. Must be a value number. See table below for possible error report levels
- **error_message**:-Required. Specifies the error message for the user-defined error
- **error_file :-**Optional. Specifies the filename in which the error occurred
- **error_line :-**Optional. Specifies the line number in which the error occurred
- **error_context ;-**Optional. Specifies an array containing every variable, and their values, in use when the error occurred

example:

```
<html>
<head><title>custom error</title>
<style type ="text/css">
.notice{color:yellow;}
.warning{color:red;}
</style>
</head>
<body>
<?php
set_error_handler("myhandler");
error_reporting(E_ALL);
//generate some error
echo $var;
echo 1/0;
//custom error handler
function myhandler($type,$msg,$file,$line,$context){
$type="an error occur on line.$line while processing your request."
switch($type){
case E_NOTICE:
```

```
echo"<div class=\"notice\">$text</div>";
break;
case E_WARNING:
echo"<div class=\"warning\">$text<div>";
break:
}
}
?>
</body>
</html>
```

Error reporting

we can control which error are the displayed in script output with a built-in function called error_reporting().

eg:

```
<?php//only display fatal errors
error_reporting(E_ERROR);
echo 1/0;
?>
```

we can also use error_reporting() to **turn off** run time fatal errors.

eg:

```
<? php
//only display warnings
error_reporting(E_WARNING);
echo somefunc();
?>
```

it is important to realize that error_reporting() doesnot automatically make our script error free.all it does is hide error of a certain type .we can also pass error_reporting() a combination of error levels ,to customize errors..

eg:

```
<? php
//only display notice and fatal errors
error_reporting(E_NOTICE|E_ERROR);
echo $var; //notice
echo 1/0; //warning
echo myfun(; //fatal
?>
```

Error Types:

Notice: These are small, non-critical errors that PHP encounters while executing a script.that dont stop PHP from executing a script. for example,:accessing a variable that has not yet been defined

Warnings:seriuos errors that require attention but still dont stop script to execute.eg: reading a file that doesnt exist in given path ,division by zero.

Fatal errors:syntax error or critical errors that stop execution .eg :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

E_RECOVERABLE_ERROR

E_ALL

Exception handling

Exception handling is used to change the normal flow of the code execution.if a specified error(exceptional) condition occurs.This condition is called an exception.

This is what normally happens when an exception is triggered:

- The current code state is saved
- The code execution will switch to a predefined (custom) exception handler function
- Depending on the situation, the handler may then resume the execution from the saved code state, terminate the script execution or continue the script from a different location in the code

Basic use of exception

When an exception is thrown, the code following it will not be executed, and PHP will try to find the matching "catch" block.

eg:

```
<? 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
checkNum(2);
?>
```

The code above will generate an error like this:

```
Fatal error: Uncaught exception 'Exception'
with message 'Value must be 1 or below' in C:\webfolder\test.php:6
Stack trace: #0 C:\webfolder\test.php(12):
checkNum(28) #1 {main} thrown in C:\webfolder\test.php on line 6
```

The above error generates because try and catch is not defined to handle the error.

Try throw and catch

To avoid the error from the example above, we need to create the proper code to handle an exception.

Proper exception code should include:

1. Try - A function using an exception should be in a "try" block. If the exception does not trigger, the code will continue as normal. However, if the exception triggers, an exception is "thrown"
2. Throw - This is how you trigger an exception. Each "throw" must have at least one "catch"
3. Catch - A "catch" block retrieves an exception and creates an object containing the exception information

Let's try to trigger an exception with valid code:

```
<?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();
}
?>
```

Explanation of above example:

1. The checkNum() function is created. It checks if a number is greater than 1. If it is, an exception is thrown
2. The checkNum() function is called in a "try" block
3. The exception within the checkNum() function is thrown
4. The "catch" block retrieves the exception and creates an object ($e) containing the exception information
5. The error message from the exception is echoed by calling $e->getMessage() from the exception object

However, one way to get around the "every throw must have a catch" rule is to set a top level exception handler to handle errors that slip through.

Creating a custom Exception class

Creating a custom exception handler is quite simple .We simply create a special class with functions that can be called when an exception occurs in PHP.the class must be an extension of the Exception class.The custom exception class inherits the properties from PHP's exception class and you can add custom functions to it.

eg:

```
<?php
```

```
class customException extends Exception {
public function errorMessage() {
//error message
$errorMsg = 'Error on line '.$this->getLine().' in '.$this->getFile()
.': <b>'.$this->getMessage().'</b> is not a valid E-Mail address';
return $errorMsg;
}
}
$email = "someone@example...com";
try {
//check if
if(filter_var($email, FILTER_VALIDATE_EMAIL) === FALSE) {
//throw exception if email is not valid
throw new customException($email);
}
}
catch (customException $e) {
//display custom message
echo $e->errorMessage();
}
?>
```

The new class is a copy of the old exception class with an addition of the errorMessage() function. Since it is a copy of the old class, and it inherits the properties and methods from the old class, we can use the exception class methods like getLine() and getFile() and getMessage().

explanation of above example:

The code above throws an exception and catches it with a custom exception class:

1. The customException() class is created as an extension of the old exception class. This way it inherits all methods and properties from the old exception class
2. The errorMessage() function is created. This function returns an error message if an e-mail address is invalid
3. The $email variable is set to a string that is not a valid e-mail address
4. The "try" block is executed and an exception is thrown since the e-mail address is invalid
5. The "catch" block catches the exception and displays the error message

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.

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 tagsThe 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, 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>