LECTURE NOTES ON

Web and Internet Technologies
(15A05605)

III B.TECH II SEMESTER

(JNTUA-R15)

Submitted by: K.Dhanamjay

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VEMU INSTITUTE OF TECHNOLOGY:: P.KOTHAKOTA

Chittoor-Tirupati National Highway, P.Kothakota, Near Pakala, Chittoor (Dt.), AP - 517112

(Approved by AICTE, New Delhi Affiliated to JNTUA Ananthapuramu. ISO 9001:2015 Certified Institute)

UNIT-1

Introduction to web servers:

What is web?

- A collection of cross-linked "websites" which uses URI.
- The consistent use of URIs to represent resources.
- HTTP, HTML, and everything built around them (web).
- This provides to invoke the data across universally over the net.

What is server?

- A server is a **computer or device** on a network that manages network resources.
- Most servers are dedicated. This means that they perform only one task rather than multiple tasks on multiprocessing operating systems, however, a single computer can execute several programs at once.

What is web server?

- A Web server is a **program** that generates and transmits responses to **client requests** for Web resources.
- Handling a client request consists of **several key steps**:
- Parsing the request message
- Checking that the request is authorized
- Associating the URL in the request with a file name
- Constructing the response message
- Transmitting the response message to the requesting client
- server can generate the response message in a variety of ways:
- The server simply retrieves the file associated with the URL and returns the contents to the client.
- The server may invoke a script that communicates with other servers or a back-end database to construct the response message.

Web Site versus Web Server?

Web site and ***Web server*** are different:

- A ***Web site*** consists of a collection of Web pages associated with a particular hostname.
- A ***Web server*** is a program to satisfy client requests for Web resources.

Types of Web Servers:

1. Apache Web Server
2. IIS Server
3. Xampp Server
4. WAMP Server

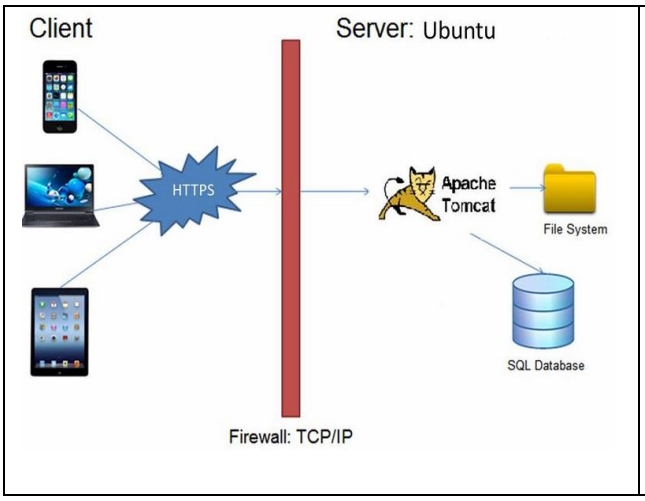
Apache Web Server:

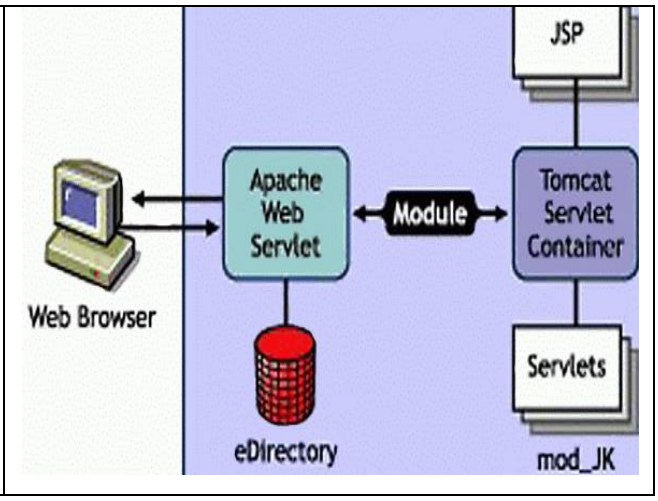
- Apache Web server is the most commonly used http server today. About 80% of all websites and Intranets use Apache web server to deliver their content to requesting Browsers.
- Server side programming languages such as PHP, Perl, Python, Java and many others
- The name **"Apache"** derives from the word **"patchy"** that the Apache developers used to describe early versions of their software.
- The Apache Web server provides a full range of Web server features, including CGI, SSL, and virtual domains. Apache also supports plug-in modules for extensibility. Apache is reliable, free, and relatively easy to configure.
- Apache is free software distributed by the **Apache Software Foundation**. The Apache Software Foundation promotes various free and open source advanced Web technologies.
- It can be downloaded and used completely free of cost. The first version of **Apache web server**, based on the NCSA httpd Web server, was developed in 1995.
- Apache is developed and maintained by an open community of developers under the auspices of the **Apache Software Foundation**.

A Web server is a server that is responsible for accepting HTTP requests from web clients and serving them HTTP responses, usually in the form of web pages containing static (text, images etc) and dynamic (scripts) content. The Apache Web server has been the most popular and widely used Web server for the last decade.

It is used by approximately 50% of all websites. Apache is cross-platform, lightweight, robust, and used in small companies as well as large corporations. Apache is also free and open-source. The Apache Web server has almost endless possibilities, due to its great modularity, which allows it to be integrated with numerous other applications.

Apache Tomcat server architecture:





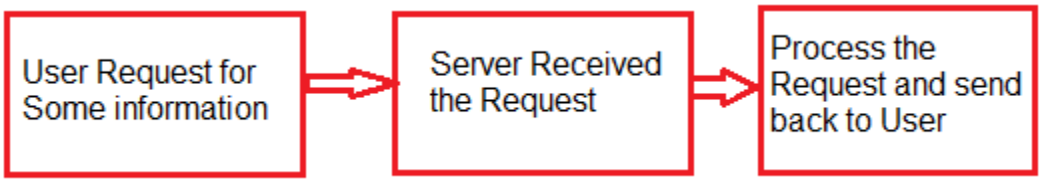
IIS - Internet Information Server

The first Microsoft web server was a research project at the European Microsoft Windows NT Academic Centre (EMWAC), part of the University of Edinburgh in Scotland, and was distributed as freeware. However, since the EMWAC server was unable to handle the volume of traffic going to Microsoft.com, Microsoft was forced to develop its own web server, IIS.

Almost every version of IIS was released either alongside or with a version of Microsoft Windows:

- IIS 1.0 was initially released as a free add-on for Windows NT 3.51.
- IIS 2.0 was included with Windows NT 4.0.

IIS Stands for "Internet Information Services (IIS)" is one of the most popular web servers from Microsoft that is used to host and provide Internet-based services to ASP.NET and ASP Web applications. A web server is responsible for providing a response to requests that come from users. When a request comes from client to server IIS takes that request from users and process it and send response back to users.



The ASP.NET Core framework is the latest generation of Active Server Page (ASP), a server-side script engine that produces interactive WebPages. A request comes in to the IIS server from the web, which sends the request to the ASP.NET Core application, which processes the request and sends its response back to the IIS server and the client who originated the request.

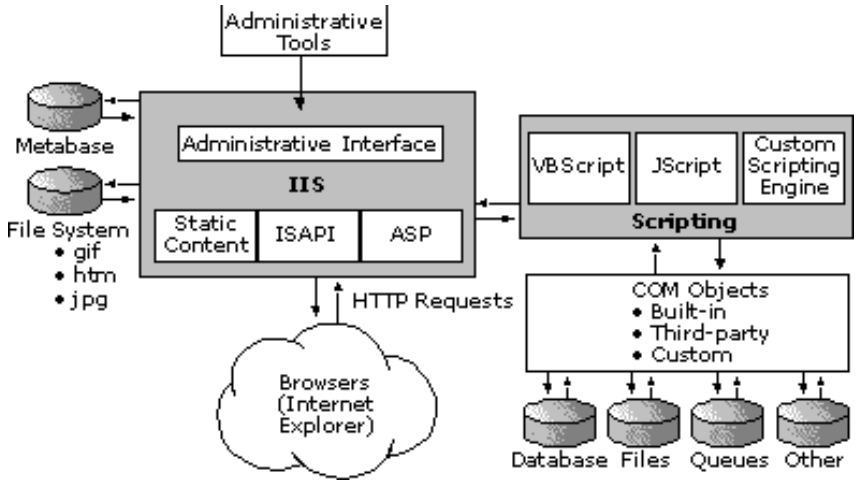
IIS is a web server software package designed for Windows Server. It is used for hosting websites and other content on the Web. Microsoft's Internet Information Services provides a graphical user interface (GUI) for managing websites and the associated users.

It provides a visual means of creating, configuring, and publishing sites on the web. The IIS Manager tool allows web administrators to modify website options, such as default pages, error pages, logging settings, security settings, and performance optimizations.

IIS can serve both standard HTML WebPages and dynamic WebPages, such as ASP.NET applications and PHP pages. When a visitor accesses a page on a static website, IIS simply sends the HTML and associated images to the user's browser. The way you configure an ASP.NET application depends on what version of IIS the application is running on.

When a page on a dynamic website is accessed, IIS runs any applications and processes any scripts contained in the page, and then sends the resulting data to the user's browser.

IIS 7.5 supports HTTP, HTTPS, FTP, FTPS, SMTP and NNTP.



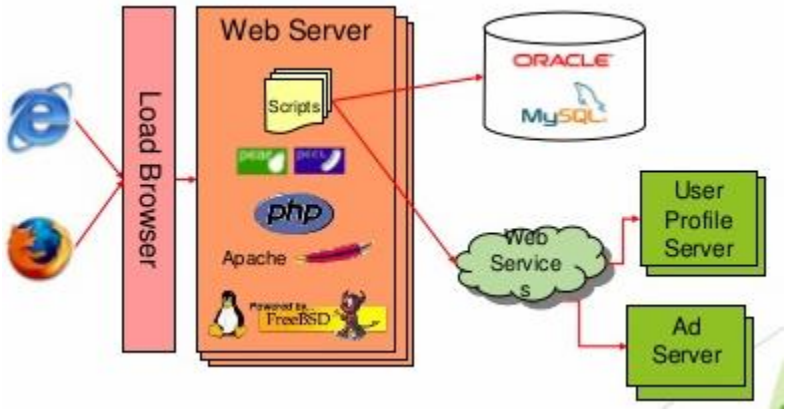
XAMPP Server:

XAMPP is a free and open source cross-platform web server solution stack package developed by Apache Friends. XAMPP consisting mainly of the Apache HTTP Server, MariaDB database and interpreters for scripts written in the PHP and Perl programming languages. XAMPP stands for Cross-Platform (X), Apache (A), MariaDB (M), PHP (P) and Perl (P).

"XAMPP Apache + MariaDB + PHP + Perl". MySQL was replaced with MariaDB on 2015-10-19 and beginning with XAMPP versions 5.5.30 and 5.6.14 effectively. XAMPP has the ability to serve web pages on the World Wide Web. A special tool is provided to password-protect the most important parts of the package.

It is a simple, lightweight Apache distribution that makes it extremely easy for developers to create a local web server for testing and deployment purposes. Everything needed to set up a web server – server application (Apache), database (MariaDB), and scripting language (PHP) – is included in an extractable file. XAMPP is also cross-platform, which means it works equally well on Linux, Mac and Windows. Since most actual web server deployments use the same components as XAMPP, it makes transitioning from a local test server to a live server extremely easy as well.

Architecture:

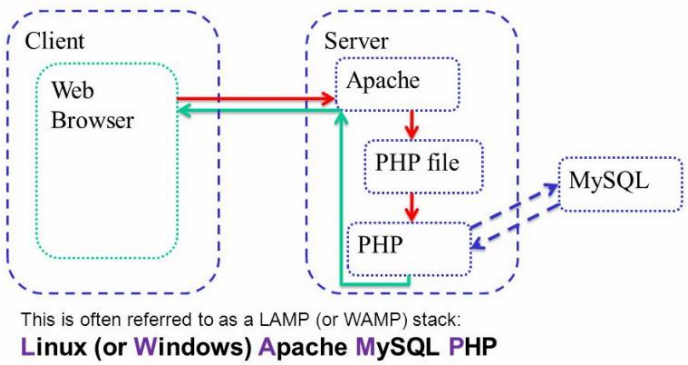


Wamp server:

- WAMP is acronym for Windows operating system, Apache, MySQL and PHP. Here Apache is web server, Mysql is database and PHP is a server side scripting language.
- All of this are open source software. Instead of using WAMP server, we can also install Apache, PHP and mysql separately. But newbie php/mysql programmer doesn't have idea about Apache settings, WAMP server ease the job of PHP settings and other server settings.

- The first version of WAMP server is EasyPHP, which is launched in 1999. WAMP is a variation of LAMP for Windows systems and is often installed as a software bundle (Apache, MySQL, and PHP). It is often used for web development and internal testing, but may also be used to serve live websites.
- The most important part of the WAMP package is Apache (or "Apache HTTP Server") which is used run the web server within Windows. By running a local Apache web server on a Windows machine, a web developer can test webpages in a web browser without publishing them live on the Internet.
- WAMP also includes MySQL and PHP, which are two of the most common technologies used for creating dynamic websites. MySQL is a high-speed database, while PHP is a scripting language that can be used to access data from the database.

Architecture:



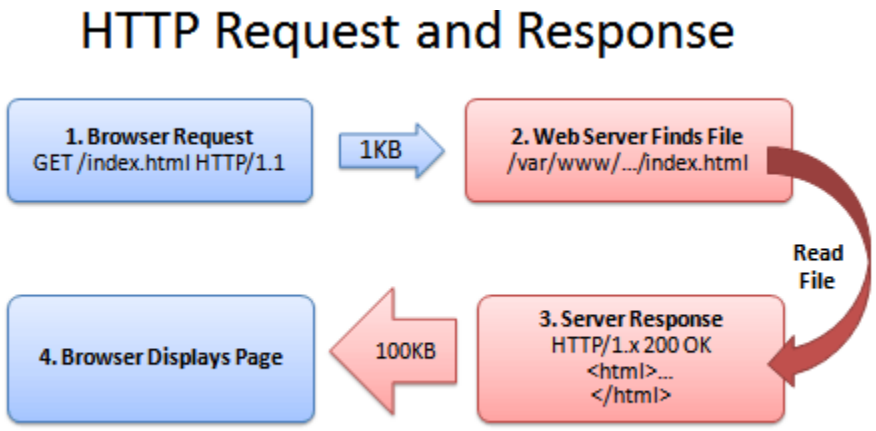
Handling HTTP Request and Response:

What is HTTP?

HTTP stands for HyperText Transfer Protocol. This is a basis for data communication in the internet. The data communication starts with a request sent from a client and ends with the response received from a web server.

- A website URL starting with “http://” is entered in a web browser from a computer (client). The browser can be a Chrome, Firefox, Edge, and Safari, Opera or anything else.
- Browser sends a request sent to the web server that hosts the website.
- The web server then returns a response as a HTML page or any other document format to the browser.
- Browser displays the response from the server to the user.

The symbolic representation of a HTTP communication process is shown in the below picture:



HTTP Request Structure from Client

A simple request message from a client computer consists of the following components:

- A request line to get a required resource, for example a request GET /content/page1.html is requesting a resource called /content/page1.html from the server.
- Headers (Example – Accept-Language: EN).
- An empty line.
- A message body which is optional.

All the lines should end with a carriage return and line feed. The empty line should only contains carriage return and line feed without any spaces.

HTTP Response Structure from Web Server

A simple response from the server contains the following components:

- HTTP Status Code (For example HTTP/1.1 301 Moved Permanently, means the requested resource was permanently moved and redirecting to some other resource).
- Headers (Example – Content-Type: html)
- An empty line.
- A message body which is optional.

All the lines in the server response should end with a carriage return and line feed. Similar to request, the empty line in a response also should only have carriage return and line feed without any spaces.

Example of HTTP Session

Let us take an example that you want to open a page “home.html” from the site “yoursite.com”. Below is how the request from the client browser should look like to get a “home.html” page from “yoursite.com”.

HTTP Request Structure

```
GET /home.html HTTP/1.1
Host: www.yoursite.com
```

The response from the web server should look like below:

HTTP Response Structure

```
HTTP/1.1 200 OK
Date: Sun, 28 Jul 2013 15:37:37 GMT
Server: Apache
Last-Modified: Sun, 07 Jul 2013 06:13:43 GMT
Transfer-Encoding: chunked
Connection: Keep-Alive
Content-Type: text/html; Charset=UTF-8
Webpage Content
```

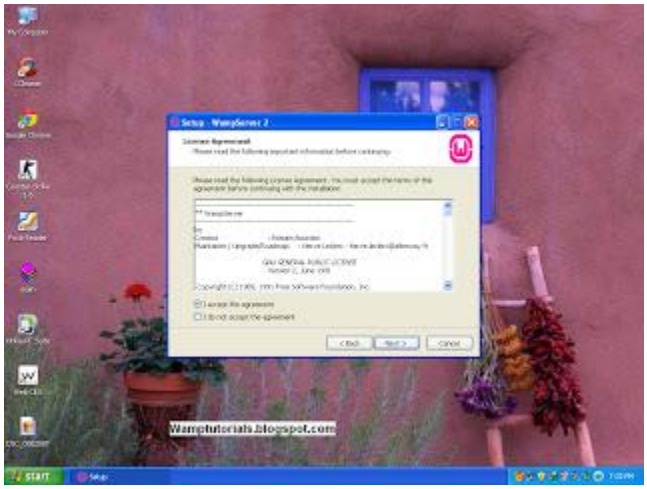
Chunked transfer encoding is a method in which the server responds with a data in chunks and this used in place of Content-Length header. The communication is stopped when a zero length of chunk is received and this method is used in HTTP Version 1.1.

Installing Wamp Server

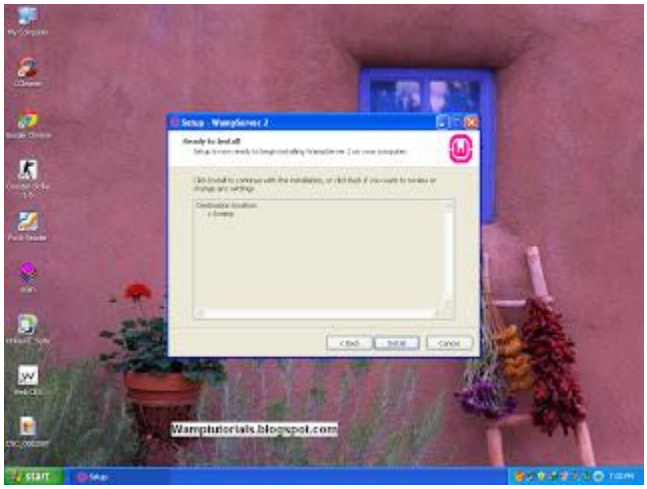
1-open your download file or you can download form the following link

2-Click Next button

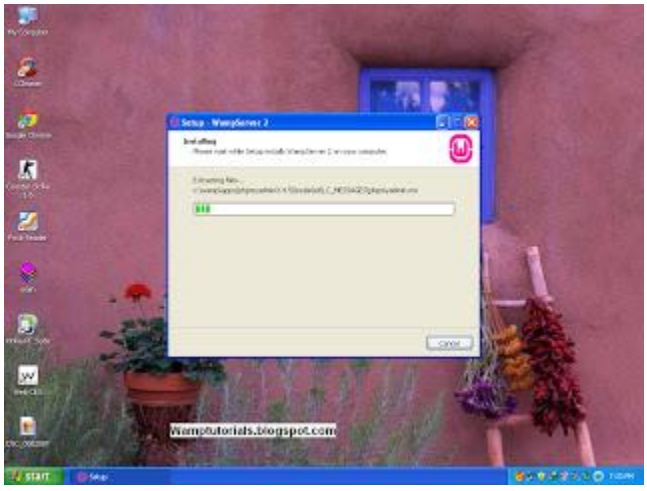
3- Accept the agreement and press next button



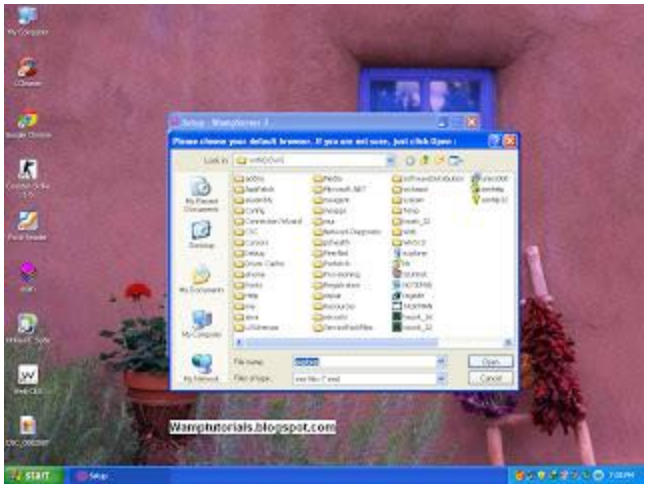
4-Click on the install button



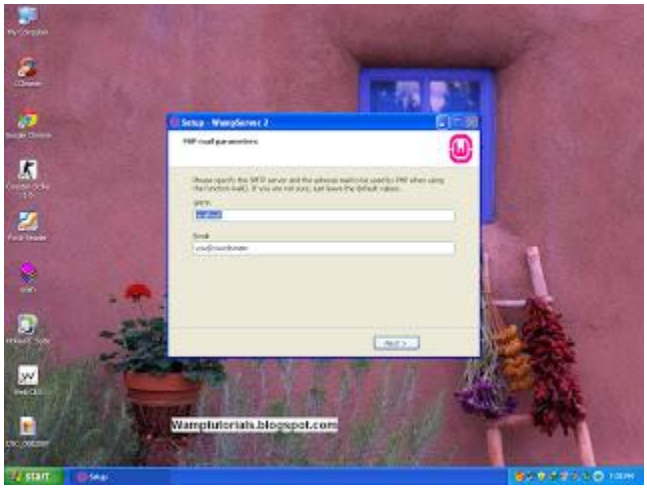
5- Wamp server is being installing



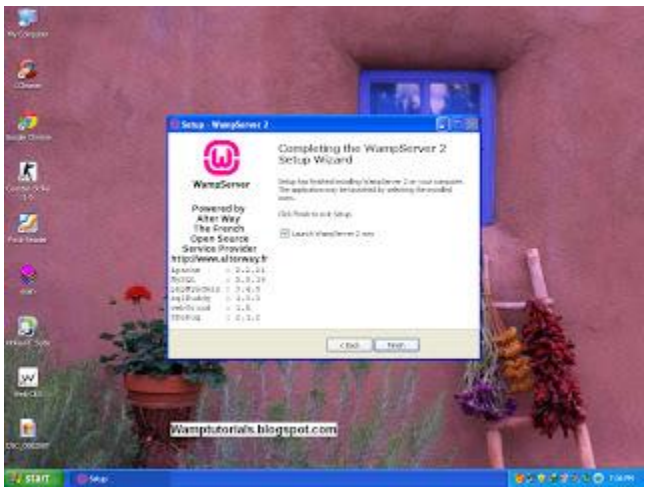
6- Choose your browser you want to use if you want to use internet explorer then press open if you want to use other browse then specify the destination of the browser



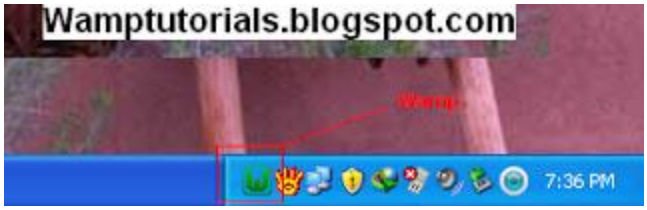
7-Select the SMTP you want to use :e.g: if you want to use your name then put your name in the first box and then replace the "localhost" with your name in the second box



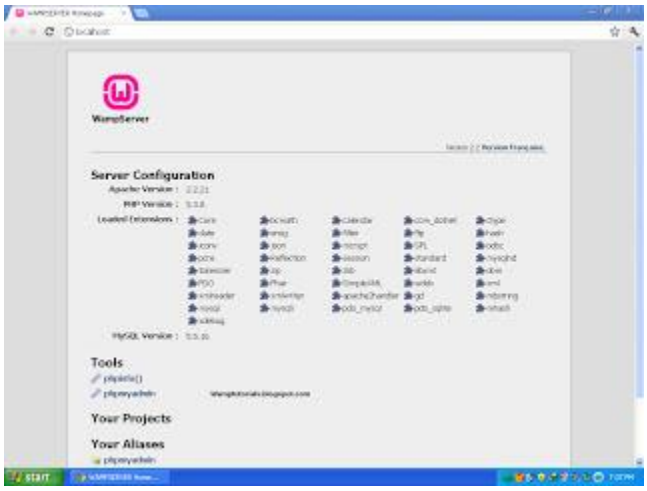
8- Wamp server has been install click finish.



9- When you launch Wamp it appears in your tray



10- Type 'localhost ' in your browser URL box
Wamp configuration page will be appeared.



11.Now you can use it.

Unit-III Introduction to PHP & Advanced Concepts

1) The problem with other Technologies (Servlets and JSP):

Servlet : Servlet is a server side program which runs based on request and response architecture.

- When a client made a request to the server from the browser, the server bypass the request to the servlet which can handles the request.

Problems with Servlets:

- In many java Servlet based applications, processing the request and generating the response are both handled by a single servlet class.
- Changing the look and feel of an application or adding support for a new type of client requires servlet code to be updated and recompiled.
- If your application is build on using servlet technology, it is very difficult for enhancement and bug fixing.
- It is time taking to design an application interface using web page development tools because HTML elements are embedded in to the servlet code manually.
- **To over come this we use JSP:**

JSP(Java ServerPages):

- Java Server Pages is a technology for developing web pages that include dynamic content.
- JSP not only contains standard markup language elements like HTML tags but also contains special JSP elements which allow the server to insert the dynamic content in the page.
- When a user requests a JSP page, servers executes JSP elements and merge the results with static parts of the page and sends the dynamically composed page back to the browser.

Problems with JSP:

- JSP pages must be compiled on the server when first accessed and produces a noticeable delay when accessing the JSP page for the first time.
- JSP pages require about double the disk space to hold the page (along with JSP page, resultant class file is stored in server).
- Database connectivity is not as easy as it should be.
- JSP requires a compiler to be shipped with the web server.
- To deploy and run Java Server Pages, a compatible web server with a servlet container, such as Apache Tomcat is needed.

2) DOWNLOADING INSTALLING AND CONFIGURING PHP:

To Install the Apache and PHP on windows (apache 2 is considered)

Apache installation:

- Start the apache installer by double clicking the icon which contains the name as “apache_x.x.xx-win 32-x86-no-ssl.msi”.
- The installation process begins with a welcome screen. Take a moment to read the screen and then click Next.
- The license agreement is displayed next. Carefully read through the license. Assuming that you agree with the license stipulations, click Next.
- A screen containing various items pertinent to the Apache server is displayed next, read through this information and then click Next.

Apache installation contd..

- You will be prompted for various items pertinent to the server’s operation[1].
- You’ll also be prompted as to whether Apache should run as a service for all users or only for the current user[2]. When you’re finished, click Next.
- You are prompted for a Setup Type: Typical or Custom.
- You’re prompted for the Destination folder. By default, this is C:\Program Files\Apache Group. When you are finished click Next.

Click the install button for installation of Apache.

To Download the PHP

- Go to the official website http://www.php.net.
- At the bottom right corner of the page click on the link “mirror sites” which will redirect to http://www.php.net/mirrors.php

- Now Select the link based on country code and click on the downloads link. For example if the country is India, select the link(s) of country code "in".
- Now click on downloads link at the top of the page.
- Once you are in downloads page, choose one of the available distributions.
  - **Source** → If Linux is your target server platform
  - **Windows zip package** → Plan to use PHP in conjunction with Apache on Windows
- Download the appropriate distribution as per your target server platform.

PHP Installation and Configuration:

Extract the PHP files:

Step 1 : Extract the PHP zip and copy to C:\ directory.

Php-5.2.6-Win32.zip

Step 2 : Rename the folder to php thus **C:\php.**

Configure PHP.ini:

Step 3: Look for the file **php.ini** and open php.ini with your favorite text editor

Step 4 : Look for **display_errors = Off** on line 372. With this directive set to off, errors that occur during the execution of scripts will no longer be displayed so that we should make it on as follows.

display_errors = On.

Step 5 : Open **httpd.conf.txt** on C:\Program Files\Apache Software Foundation\Apache2.2\conf and Edit the Apache httpd.conf Configuration File.

Configure PHP as an Apache module:

Step 6: Add PHP module to last line of list of modules on Apache configuration file.

Add **LoadModule php5_module "C:/php/php5apache2_2.dll".**

Step 7 : Look for the line **DirectoryIndex index.html** which can be found on

<IfModule dir_module>

Change it to **DirectoryIndex index.php index.html.** So that Apache will serve the file index.php

Step 8 : Look for the following lines under **<IfModule mime_module> section:**

- AddType application/x-compress .Z
- AddType application/x-gzip .gz .tgz

now append the following file to the next line.

AddType application/x-httpd-php .php

Step 9 : At the last line add **PHPIniDir <php.ini directory>** in order for the Apache to know the php.ini configuration. Save the file.

Step 10: Restart the Apache Server. It shows messages if there is any wrong in configuration.

Test a PHP file:

Step 10 : To test if Apache and PHP work together, make a file with the following codes on your text editor and save with the desired name and with **.php extension.**

For ex :

```
<?php
        phpinfo();
?>
```

3) The anatomy of a PHP Page:

Before going to learn PHP must and should have a basic understanding of the following:

• HTML	• CSS	• JavaScript	• JQuery

What is PHP?

- PHP originally stands for "Personal Home Page"
- Later Personal Home Page renamed as "PHP: Hypertext Preprocessor"
- Developed by Rasmus Lerdorf was the first version of PHP way back in 1994.
- PHP is a widely-used, open source scripting language
- PHP is a server side scripting language.
- PHP is an interpreted language, i.e. there is no need for compilation.
- PHP is an open-source scripting language.
- PHP is free to download and use
- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code are executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension ".php"

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

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

PHP Features: There are given many features of PHP.

- **Performance**: Script written in PHP executes much faster then 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.

Overview of PHP Data types and Concepts

4) Overview of PHP Concepts:

Basic PHP Syntax:

A PHP script can be placed anywhere in the document.A PHP script starts with **<?php** and ends with **?>**

```
<?php
// PHP code goes here
?>
```

The default file extension for PHP files is **".php".**

Example:

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

PHP Echo:

PHP echo is a language construct not a function, so you don't need to use parenthesis with it. But if you want to use more than one parameters, it is required to use parenthesis.The **syntax of PHP echo** is given below:

```
void echo ( string $arg1 [, string $... ] )
```

PHP echo statement can be used to print string, multi line strings, escaping characters, variable, array etc.

PHP echo: printing string

```
1. <?php
2. echo "Hello by
    PHP echo";
3. ?>
```

PHP echo: printing multi line string

```
1. <?php
2. echo "Hello by PHP
   echo
3. this is multi line
4. text printed by
5. PHP echo statement
6. ";
7. ?>
```

PHP echo: printing escaping characters

```
1. <?php
2. echo "Hello escape
   \"sequence\" chara
   cters";
3. ?>
```

PHP echo: printing variable value

```
1. <?php
2. $msg="Hello JavaTpo
   int PHP";
3. echo "Message is: $ms
   g";
4. ?>
```

PHP Print:

Like PHP echo, PHP print is a language construct, so you don't need to use parenthesis with the argument list. Unlike echo, it always returns 1.

The syntax of PHP print is given below:

int print(string $arg)

PHP print statement can be used to print string, multi line strings, escaping characters, variable, array etc.

PHP print: printing string

```
1. <?php
2. print "Hello by PHP print ";
3. print ("Hello by PHP print()");
4. ?>
```

PHP Variables:

A variable in PHP is a name of memory location that holds data. A variable is a temporary storage that is used to store data temporarily.In PHP, a variable is declared using $ sign followed by variable name.

Syntax of declaring a variable in PHP is given below:

$variablename=value;

PHP Variable: Declaring string, integer and float

```
1. <?php
2. $str="hello string";
3. $x=200;
4. $y=44.6;
5. echo "string is: $str <br/>";
6. echo "integer is: $x <br/>";
7. echo "float is: $y <br/>";
8. ?>
```

PHP Variable: Sum of two variables

```
1. <?php
2. $x=5;
3. $y=6;
4. $z=$x+$y;
5. echo $z;
6. ?>
```

PHP Variable: case sensitive

In PHP, variable names are case sensitive. So variable name "color" is different from Color, COLOR, COLor etc.

```
1. <?php
2. $color="red";
3. echo "My car is " . $color . "<br>";
4. echo "My house is " . $COLOR . "<br>";

5. echo "My boat is " . $coLOR . "<br>";
6. ?>
```

PHP Variable: Rules

PHP variables must start with letter or underscore only.PHP variable can't be start with numbers and special symbols.

```
1. <?php
2. $a="hello";//letter (valid)
3. $_b="hello";//underscore (valid)
4.
5. echo "$a <br/> $_b";
6. ?>
```

```
1. <?php
2. $4c="hello";//number (invalid)
3. $*d="hello";//special symbol (invalid)
4.
5. echo "$4c <br/> $*d";
6. ?>
```

PHP: Loosely typed language

PHP is a loosely typed language; it means PHP automatically converts the variable to its correct data type.

PHP $ and $$ Variables:

The **$var** (single dollar) is a normal variable with the name var that stores any value like string, integer, float, etc.The **$$var** (double dollar) is a reference variable that stores the value of the $variable inside it.

```
1. <?php
2. $x = "abc";
3. $$x = 200;
4. echo $x."<br/>";
5. echo $$x."<br/>";
6. echo $abc;
7. ?>
```

PHP Constants

PHP constants are name or identifier that can't be changed during the execution of the script. PHP constants can be defined by 2 ways:

1. Using define() function
2. Using const keyword

PHP constants follow the same PHP variable rules. For example, it can be started with letter or underscore only.Conventionally, PHP constants should be defined in uppercase letters.

PHP constant: define()

Let's see the syntax of define() function in PHP.

1. define(name, value, case-insensitive)

where as name: specifies the constant name
value: specifies the constant value
case-insensitive: Default value is false. It means it is case sensitive by default.

Let's see the example to define PHP constant using define().

```
1. <?php
2. define("MESSAGE","Hello JavaTpoint PHP");
3. echo MESSAGE;
4. ?>
```

```
1. <?php
2. define("MESSAGE","Hello JavaTpoint PHP",true);//not case sensitive
3. echo MESSAGE;
4. echo message;
5. ?>
```

```
1. <?php
2. define("MESSAGE","Hello JavaTpoint PHP",false);//case sensitive
3. echo MESSAGE;
4. echo message;
5. ?>
```

PHP constant: const keyword

The const keyword defines constants at compile time. It is a language construct not a function.It is bit faster than define().It is always case sensitive.

```
1. <?php
2. const MESSAGE="Hello const by JavaTpoint PHP";
3. echo MESSAGE;
4. ?>
```

PHP Case Sensitivity:

In PHP, all keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are NOT case-sensitive.

```
<!DOCTYPE html>
<html>
<body>
<?php
ECHO "HelloWorld!<br>";
echo "HelloWorld!<br>";
EcHo "HelloWorld!<br>";
?>
</body>
</html>
```

However; all variable names are case-sensitive.

```
<!DOCTYPE html>
<html>
<body>
<?php
$color = "red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?></body></html>
```

PHP Variables Scope:

In PHP, variables can be declared anywhere in the script.

The scope of a variable is the part of the script where the variable can be referenced/used.

PHP has three different variable scopes:

- local
- global
- static

Global and Local Scope

A variable declared **outside** a function has a **GLOBAL SCOPE** and can only be accessed outside a function:

Example

```
<?php
$x = 5; // global scope
function myTest() {
   // using x inside this function will generate an error
   echo "<p>Variable x inside function is: $x</p>";
}
myTest();
echo "<p>Variable x outside function is: $x</p>";
?>
```

A variable declared **within** a function has a **LOCAL SCOPE** and can only be accessed within that function:

Example

```
<?php
function myTest() {
   $x = 5; // local scope
   echo "<p>Variable x inside function is: $x</p>";
}
myTest();
// using x outside the function will generate an error
echo "<p>Variable x outside function is: $x</p>";
?>
```

PHP The static Keyword:

Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.To do this, use the static keyword when you first declare the variable:

Example

```
<?php
function myTest() {
   static $x = 0;
   echo $x;
   $x++;
}
myTest();
myTest();
myTest();
?>
```

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.

5) PHP Data Types:

PHP data types are used to hold different types of data or values. PHP supports 8 primitive data types that can be categorized further in **3 types:**

1. Scalar Types
2. Compound Types
3. Special Types

PHP Data Types: Scalar Types	PHP Data Types: Compound Types	PHP Data Types: Special Types
There are 4 scalar data types in PHP. 1. boolean 2. integer 3. float 4. string	There are 2 compound data types in PHP. 1. array 2. object	There are 2 special data types in PHP. 1. resource 2. NULL

PHP String:

A string is a sequence of characters, like "Hello world!".A string can be any text inside quotes. You can use single or double quotes:

```
<?php
$x = "Hello world!";
$y = 'Hello world!';
echo $x;
echo "<br>";
echo $y;
?>
```

PHP Integer:

An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.

Rules for integers:

- An integer must have at least one digit
- An integer must not have a decimal point
- An integer can be either positive or negative
- Integers can be specified in three formats: decimal (10-based), hexadecimal (16-based - prefixed with 0x) or octal (8-based - prefixed with 0)

In the following example $x is an integer. The PHP **var_dump() function** returns the data type and value:

```
<?php
$x = 5985;
var_dump($x);
?>
```

PHP Float:

A float (floating point number) is a number with a decimal point or a number in exponential form.

In the following example $x is a float. The PHP var_dump() function returns the data type and value:

```
<?php
$x = 10.365;
var_dump($x);
?>
```

PHP Boolean:

A Boolean represents two possible states: TRUE or FALSE.

```
$x = true;
$y = false;
```

Booleans are often used in conditional testing.

PHP Array:

An array stores multiple values in one single variable.In the following example $cars is an array. The PHP var_dump() function returns the data type and value:

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

PHP Object:

An object is a data type which stores data and information on how to process that data.In PHP, an object must be explicitly declared.First we must declare a class of object. For this, we use the class keyword. A class is a structure that can contain properties and methods:

```
<?php
class Car {
   function Car() {
      $this->model = "VW";
   }
}
// create an object
$herbie = new Car();
// show object properties
echo $herbie->model;
?>
```

PHP NULL Value:

Null is a special data type which can have only one value: NULL.A variable of data type NULL is a variable that has no value assigned to it.If a variable is created without a value, it is automatically assigned a value of NULL.Variables can also be emptied by setting the value to NULL:

```
<?php
$x = "Hello world!";
$x = null;
var_dump($x);
?>
```

PHP Resource:

The special resource type is not an actual data type. It is the storing of a reference to functions and resources external to PHP.A common example of using the resource data type is a database call.

6) PHP Strings: A string is a sequence of characters, like "Hello world!". i.e. used to store and manipulate text. There are 4 ways to specify string in PHP.

- single quoted
- double quoted

Single Quoted PHP String

We can create a string in PHP by enclosing text in a single quote. It is the easiest way to specify string in PHP.

```
1. <?php
2. $str='Hello text within single quote';
3. echo $str;
4. ?>
```

Double Quoted PHP String

In PHP, we can specify string through enclosing text within double quote also. But escape sequences and variables will be interpreted using double quote PHP strings.

```
1. <?php
2. $str="Hello text within double quote";
3. echo $str;
4. ?>
```

PHP String Functions: There are number of string handling functions are as follows:

Some commonly used functions to manipulate strings.

Get The Length of a String:

The PHP strlen() function returns the length of a string. The example below returns the length of the string "Hello world!":

```
<?php
echo strlen("Hello world!"); // outputs 12
?>
```

Count The Number of Words in a String:

The PHP str_word_count() function counts the number of words in a string:

```
<?php
echo str_word_count("Hello world!"); // outputs 2
?>
```

Reverse a String:

The PHP strrev() function reverses a string:

```
<?php
echo strrev("Hello world!"); // outputs !dlrow olleH
?>
```

Search For a Specific Text Within a String:

The PHP strpos() function searches for a specific text within a string.If a match is found, the function returns the character position of the first match. If no match is found, it will return FALSE.The example below searches for the text "world" in the string "Hello world!":

```
<?php
echo strpos("Hello world!", "world"); // outputs 6
?>
```

Replace Text Within a String:

The PHP str_replace() function replaces some characters with some other characters in a string.The example below replaces the text "world" with "Dolly":

```
<?php
echo str_replace("world", "Dolly", "Hello world!"); // outputs Hello Dolly!
?>
```

PHP ucwords() function:

The ucwords() function returns string converting first character of each word into uppercase.

Syntax

string ucwords (string $str)

Example

```
<?php
$str="my name is Sonoo jaiswal";
$str=ucwords($str);
echo $str;
?>
```

PHP strtolower() function:

The strtolower() function returns string in lowercase letter.

Syntax

string strtolower (string $string)

Example

```
<?php
$str="My name is KHAN";
$str=strtolower($str);
echo $str;
?>
```

PHP strtoupper() function:

The strtoupper() function returns string in uppercase letter.

Syntax

string strtoupper (string $string)

Example

```
<?php
$str="My name is KHAN";
$str=strtoupper($str);
echo $str;
?>
```

PHP ucfirst() function:

The ucfirst() function returns string converting first character into uppercase. It doesn't change the case of other characters.

Syntax

```
string ucfirst ( string $str )
```

Example

```
<?php
$str="my name is KHAN";
$str=ucfirst($str);
echo $str;
?>
```

PHP lcfirst() function:

The lcfirst() function returns string converting first character into lowercase. It doesn't change the case of other characters.

Syntax

```
string lcfirst ( string $str )
```

Example

```
<?php
$str="MY name IS KHAN";
$str=lcfirst($str);
echo $str;
?>
```

7) 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

- String operators
- Array operators

PHP Arithmetic Operators:

The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc.

Operator	Name	Example	Result
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y
*	Multiplication	$x * $y	Product of $x and $y
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power (Introduced in PHP 5.6)

PHP Assignment Operators:

The PHP assignment operators are used with numeric values to write a value to a variable.The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.

Assignment	Same as...	Description
x = y	x = y	The left operand gets set to the value of the expression on the right
x += y	x = x + y	Addition
x -= y	x = x - y	Subtraction
x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus

PHP Comparison Operators: The PHP comparison operators are used to compare two values (number or string):

Operator	Name	Example	Result
==	Equal	$x == $y	Returns true if $x is equal to $y
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type

>	Greater than	$x > $y	Returns true if $x is greater than $y
<	Less than	$x < $y	Returns true if $x is less than $y
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y

PHP Increment / Decrement Operators:
The PHP increment operators are used to increment a variable's value.The PHP decrement operators are used to decrement a variable's value.

Operator	Name	Description
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

PHP Logical Operators:
The PHP logical operators are used to combine conditional statements.

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

PHP String Operators:
PHP has two operators that are specially designed for strings.

Operator	Name	Example	Result
.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2
.=	Concatenation assignment	$txt1 .= $txt2	Appends $txt2 to $txt1

PHP Array Operators:
The PHP array operators are used to compare arrays.

Operator	Name	Example	Result
+	Union	$x + $y	Union of $x and $y
==	Equality	$x == $y	Returns true if $x and $y have the same key/value pairs
===	Identity	$x === $y	Returns true if $x and $y have the same key/value pairs in the same order and of the same types
!=	Inequality	$x != $y	Returns true if $x is not equal to $y
<>	Inequality	$x <> $y	Returns true if $x is not equal to $y
!==	Non-identity	$x !== $y	Returns true if $x is not identical to $y

8) PHP Expressions and Conditional Statements:

An ***expression*** is a bit of PHP that can be evaluated to produce a value. The simplest expressions are literal values and variables. A literal value evaluates to itself, while a variable evaluates to the value stored in the variable.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 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!"; } ?>

PHP - The switch Statement:
The switch statement is used to perform different actions based on different conditions. Use the switch statement to **select one of many blocks of code to be executed**.

Syntax: switch (*n*) { case *label1:* *code to be executed if n=label1;* break; case *label2:* *code to be executed if n=label2;* break; case *label3:* *code to be executed if n=label3;* break; ... default: *code to be executed if n is different from all labels;* }	<?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!"; } ?>

9) 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 code-lines in a script, we can use loops to perform a task like this.
In PHP, we have the following looping statements:

- 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 "; $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 "; $x++; } while ($x <= 5); ?>	<?php $x = 6; do { echo "The number is: $x "; $x++; } while ($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;* } 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 "; } ?>

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;* }	<?php $colors = array("red", "green", "blue", "yellow"); foreach ($colors as $value) { echo "$value "; } ?>

10) PHP Functions:
PHP function is a piece of code that can be reused many times. It can take input as argument list and return value. There are thousands of built-in functions in PHP.

Advantage of PHP Functions:

Code Reusability: PHP functions are defined only once and can be invoked many times, like in other programming languages.

Less Code: It saves a lot of code because you don't need to write the logic many times. By the use of function, you can write the logic only once and reuse it.

Easy to understand: PHP functions separate the programming logic. So it is easier to understand the flow of the application because every logic is divided in the form of functions.

PHP User-defined Functions:

We can declare and call user-defined functions easily. Let's see the syntax to declare user-defined functions.

Syntax:

```
function functionname(){
//code to be executed
}
```

```
<?php
function sayHello(){
echo "Hello PHP Function";
}
sayHello();//calling function
?>
```

Note: Function name must be start with letter and underscore only like other labels in PHP. It can't be start with numbers or special symbols.

PHP Function Arguments:

We can pass the information in PHP function through arguments which is separated by comma.

PHP supports **Call by Value** (default), **Call by Reference**, **Default argument values** and **Variable-length argument list**.

PHP Call By Value:

PHP allows you to call function by value and reference both. In case of PHP call by value, actual value is not modified if it is modified inside the function.Let's understand the concept of call by value by the help of examples

```
<?php
function sayHello($name){
echo "Hello $name<br/>";
}
sayHello("Sonoo");
sayHello("Vimal");
sayHello("John");
?>
```

```
<?php
function sayHello($name,$age){
echo "Hello $name, you are $age years old<br/>";
}
sayHello("Sonoo",27);
sayHello("Vimal",29);
sayHello("John",23);
?>
```

```
<?php
function adder($str2)
{
    $str2 .= 'Call By Value';
}
$str = 'Hello ';
adder($str);
echo $str;
?>
```

PHP Call By Reference:

Value passed to the function doesn't modify the actual value by default (call by value). But we can do so by passing value as a reference.By default, value passed to the function is call by value. To pass value as a reference, you need to use ampersand (&) symbol before the argument name.In case of PHP call by reference, actual value is modified if it is modified inside the function. In such case, you need to use & (ampersand) symbol with formal arguments. The & represents reference of the variable.

```
<?php
function adder(&$str2)
{
    $str2 .= 'Call By Reference';
}
$str = 'Hello ';
adder($str);
echo $str;
?>
```

```
<?php
function adder(&$str2)
{
    $str2 .= 'Call By Reference';
}
$str = 'This is ';
adder($str);
echo $str;
?>
```

PHP Default Argument Value

We can specify a default argument value in function. While calling PHP function if you don't specify any argument, it will take the default argument.

```php
<?php
function sayHello($name="Sonoo"){
echo "Hello $name<br/>";
}
sayHello("Rajesh");
sayHello();//passing no value
sayHello("John");
?>
```

```php
<?php
function sayHello($name="Ram"){
echo "Hello $name<br/>";
}
sayHello("Sonoo");
sayHello();//passing no value
sayHello("Vimal");
?>
```

PHP Returning Value: PHP function that returns value.

```php
<?php
function cube($n){
return $n*$n*$n;
}
echo "Cube of 3 is: ".cube(3);
?>
```

PHP Parameterized Function:

PHP Parameterized functions are the functions with parameters. You can pass any number of parameters inside a function. These passed parameters act as variables inside your function. They are specified inside the parentheses, after the function name. The output depends upon the dynamic values passed as the parameters into the function.

Example: Addition and Subtraction

In this example, we have passed two parameters **$x** and **$y** inside two functions **add()** and **sub()**.

```php
<!DOCTYPE html>
<html>
<head>
    <title>Parameter Addition and Subtraction Example</title>
</head>
<body>
<?php
        //Adding two numbers
        function add($x, $y) {
            $sum = $x + $y;
 echo "Sum of two numbers is = $sum <br><br>";
        }
add(467, 943);

        //Subtracting two numbers
        function sub($x, $y) {
            $diff = $x - $y;
            echo "Difference between two numbers is = $diff";
        }
        sub(943, 467);
    ?>
</body>
</html>
```

PHP Recursive Function:

PHP also supports recursive function call like C/C++. In such case, we call current function within function. It is also known as recursion.It is recommended to avoid recursive function call over 200 recursion level because it may smash the stack and may cause the termination of script.

```php
<?php
function display($number) {
    if($number<=5){
     echo "$number <br/>";
     display($number+1);
    }
}

display(1);
?>
```

```php
<?php
function factorial($n)
{
    if ($n < 0)
        return -1; /*Wrong value*/
    if ($n == 0)
        return 1; /*Terminating condition*/
    return ($n * factorial ($n -1));
}

echo factorial(5);
?>
```

11) PHP Arrays:

PHP array is an ordered map (contains value on the basis of key). It is used to hold multiple values of similar type in a single variable.

Advantage of PHP Array:

Less Code: We don't need to define multiple variables.

Easy to traverse: By the help of single loop, we can traverse all the elements of an array.

Sorting: We can sort the elements of array.

PHP Array Types:

There are **3 types of array** in PHP.

1. Indexed Array
2. Associative Array
3. Multidimensional Array

PHP Indexed Array:

PHP index is represented by number which starts from 0. We can store number, string and object in the PHP array. All PHP array elements are assigned to an index number by default.There are two ways to define indexed array:

1st way:

```
1. $season=array("summer","winter","spring","autumn");
2. $size=array("Big","Medium","Short");
```

2nd way:

```
1. $season[0]="summer";
2. $season[1]="winter";
3. $season[2]="spring";
4. $season[3]="autumn";
```

```
1. <?php
2. $season=array("summer","winter","spring","autumn");
3. echo "Season are: $season[0], $season[1], $season[2] and $season[3]";
4. ?>
```

Traversing PHP Indexed Array:

We can easily traverse array in PHP using foreach loop. Let's see a simple example to traverse all the elements of PHP array.

```
1. <?php
2. $size=array("Big","Medium","Short");
3. foreach( $size as $s )
4. {
5.   echo "Size is: $s<br />";
6. }
7. ?>
```

Count Length of PHP Indexed Array:

PHP provides count () function which returns length of an array.

```
1. <?php
2. $size=array("Big","Medium","Short");
3. echo count($size);
4. ?>
```

PHP Associative Array:

We can associate name with each array elements in PHP using => symbol.

There are two ways to define associative array:

1st way:

```
1. $salary=array("Sonoo"=>"350000","John"=>"450000","Kartik"=>"200000");
2. $salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
```

2nd way:

```
1. $salary["Sonoo"]="350000";
2. $salary["John"]="450000";
3. $salary["Kartik"]="200000";
```

```php
<?php
$salary=array("Sonoo"=>"350000","John"=>"450000","Kartik"=>"200000");
echo "Sonoo salary: ".$salary["Sonoo"]."<br/>";
echo "John salary: ".$salary["John"]."<br/>";
echo "Kartik salary: ".$salary["Kartik"]."<br/>";
?>
```

```php
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
echo "Sonoo salary: ".$salary["Sonoo"]."<br/>";
echo "Vimal salary: ".$salary["Vimal"]."<br/>";
echo "Ratan salary: ".$salary["Ratan"]."<br/>";
?>
```

Traversing PHP Associative Array:

By the help of PHP for each loop, we can easily traverse the elements of PHP associative array.

```php
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
foreach($salary as $k => $v) {
echo "Key: ".$k." Value: ".$v."<br/>";
}
?>
```

PHP Multidimensional Array:

PHP multidimensional array is also known as array of arrays. It allows you to store tabular data in an array. PHP multidimensional array can be represented in the form of matrix which is represented by row * column.

Definition:

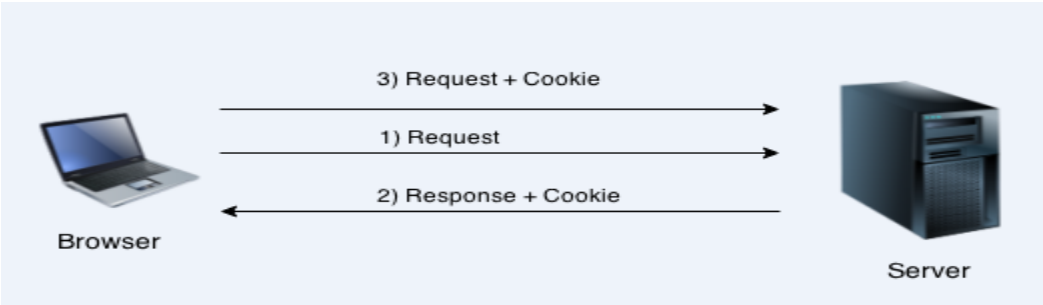
```php
$emp = array
  (
  array(1,"sonoo",400000),
  array(2,"john",500000),
  array(3,"rahul",300000)
  );
```

```php
<?php
$emp = array
  (
  array(1,"sonoo",400000),
  array(2,"john",500000),
  array(3,"rahul",300000)
  );
for ($row = 0; $row < 3; $row++) {
  for ($col = 0; $col < 3; $col++) {
    echo $emp[$row][$col]."  ";
  }
  echo "<br/>";
}
?>
```

PHP Advanced Concepts

12) 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.



Types of Cookie:

There are **2 types** of cookies in servlets.

1. Non-persistent cookie
2. Persistent cookie

Non-persistent cookie:
It is **valid for single session** only. It is removed each time when user closes the browser.
Persistent cookie:
It is **valid for multiple sessions**. It is not removed each time when user closes the browser. It is removed only if user logout or sign-out.
Advantage of Cookies:

1. Simplest technique of maintaining the state.
2. Cookies are maintained at client side.

Disadvantage of Cookies:

1. It will not work if cookie is disabled from the browser.
2. Only textual information can be set in Cookie object.

PHP setcookie() function:
PHP setcookie() function is used to set cookie with HTTP response. Once cookie is set, you can access it by $_COOKIE superglobal variable.
Note: The setcookie() function must appear BEFORE the <html> tag.
Syntax:
bool setcookie (string $name [, string $value [, int $expire = 0 [, string $path [, string $domain [, bool $secure = false [, bool $httponly = false]]]]]])
Example:

1. setcookie("CookieName", "CookieValue");/* defining name and value only*/
2. setcookie("CookieName", "CookieValue", time()+1*60*60);//using expiry in 1 hour(1*60*60 seconds or 3600 seconds);
3. setcookie("CookieName", "CookieValue", time()+1*60*60, "/mypath/", "mydomain.com", 1);

PHP $_COOKIE:
PHP $_COOKIE is superglobal variable then it is used to get cookie and are also use the isset()function.
Example

1. $value=$_COOKIE["CookieName"];//returns cookie value

PHP Cookie Example:

```
1.  <?php
2.  setcookie("user", "Sonoo");
3.  ?>
4.  <html>
5.  <body>
6.  <?php
7.  if(!isset($_COOKIE["user"])) {
8.      echo "Sorry, cookie is not found!";
9.  } else {
10.     echo "<br/>Cookie Value: " . $_COOKIE["user"];
11. }
12. ?>
13. </body>
14. </html>
```

Output:
Sorry, cookie is not found!
Firstly cookie is not set. But, if you *refresh* the page, you will see cookie is set now.
Output:
Cookie Value: Sonoo

PHP Delete Cookie:
To delete a cookie, use the setcookie() function with an expiration date in the past.

Example:

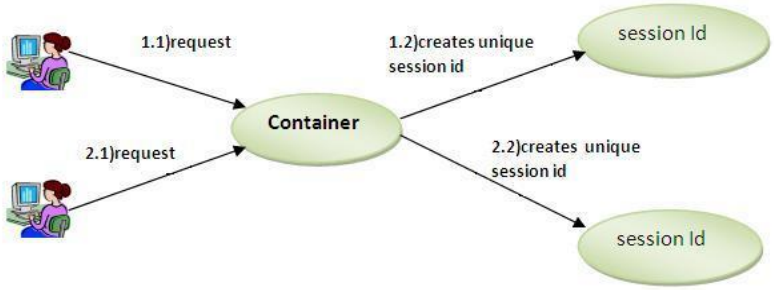
```
1.  <?php
2.  setcookie ("CookieName", "", time() -
        3600);// set the expiration date to one hour ago
3.  ?>
```

Sample Program:

```
<?php
// set the expiration date to one hour ago
setcookie("user", "", time() - 3600);
?>
<html>
<body>
<?php
echo "Cookie 'user' is deleted.";
?>
</body>
</html>
```

13) PHP Session:

PHP session is used to store and pass information from one page to another temporarily (until user close the website).PHP session technique is widely used in shopping websites where we need to store and pass cart information e.g. username, product code, product name, product price etc from one page to another.PHP session creates unique user id for each browser to recognize the user and avoid conflict between multiple browsers.



PHP session_start() function:

PHP session_start() function is used to start the session. It starts a new or resumes existing session. It returns existing session if session is created already. If session is not available, it creates and returns new session.

Note: The session_start() function must be the very first thing in your document. Before any HTML tags.

Syntax:

```
bool session_start ( void )
```

Example:

```
session_start();
```

PHP $_SESSION:

PHP $_SESSION is an associative array that contains all session variables. It is used to set and get session variable values.

Example: Store information

```
$_SESSION["user"] = "Sachin";
```

Example: Get information

```
echo $_SESSION["user"];
```

Sample program:

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>
</body>
</html>
```

PHP Destroying Session:

PHP session_destroy() function is used to destroy all session variables completely.

```
1. <?php
2. session_start();
3. session_destroy();
4. ?>
```

14) Working with Date and Time:

The PHP date () function is used to format a date and/or a time.

The PHP Date() Function:

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

Syntax: date (format, timestamp)

Parameter	Description
format	Required. Specifies the format of the timestamp
timestamp	Optional. Specifies a timestamp. Default is the current date and time

Get a Simple Date:

The required *format* parameter of the date() function specifies how to format the date (or time).

Here are some characters that are commonly used for dates:

- 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

Other characters, like"/", ".", or "-" can also be inserted between the characters to add additional formatting.

Get a Simple Time:

Here are some characters that are commonly used for times:

- h - 12-hour format of an hour with leading zeros (01 to 12)
- i - Minutes with leading zeros (00 to 59)
- s - Seconds with leading zeros (00 to 59)
- a - Lowercase Ante meridiem and Post meridiem (am or pm)

The example below formats today's date in three different ways:

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

The example below outputs the current time in the specified format:

```
<?php
echo "The time is " . date("h:i:sa");
?>
```

Get Your Time Zone:

If the time you got back from the code is not the right time, it's probably because your server is in another country or set up for a different timezone.So, if you need the time to be correct according to a specific location, you can set a timezone to use.

```
<?php
date_default_timezone_set("America/New_York");
echo "The time is " . date("h:i:sa");
?>
```

Create a Date With PHP mktime():

The optional *timestamp* parameter in the date() function specifies a timestamp. If you do not specify a timestamp, the current date and time will be used.The mktime() function returns the Unix timestamp for a date. The Unix timestamp contains the number of seconds between the Unix Epoch (January 1 1970 00:00:00 GMT) and the time specified.

Syntax:

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

Example:

```
<?php
$d=mktime(11, 14, 54, 8, 12, 2014);
echo "Created date is " . date("Y-m-d h:i:sa", $d);
?>
```

15) Using HTTP Headers:

HTTP headers are slightly finicky but rather powerful sets of functionality. The most important aspect to remember about headers is that they can be called only before any output has been written to the web page. If you attempt to call a header after output has been sent to the page, you will generate an error; hence, your script will fail on you.That being said, the functionality of headers is rather powerful. You can use them to control everything, including setting the current page location, finding out what file format is being displayed, and managing all aspects of the browser cache. In the following examples,you will learn how to use the header() function in a variety of ways. **The header() function's prototype is as follows:**

voidheader (string string [, bool replace [, inthttp_response_code]])

Redirecting to a Different Location

One of the more common uses for HTTP headers is redirecting a script. By using headers inside processing scripts, you can force the browser to return to any page you want. We prefer to use headers to control exception handling within process scripts. The following script makes sure that all input coming from a form is not blank.

Example:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<title>Sample 12.5</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
</head>
<body>
<form action="sample12_5.php" method="post">
Name: <input type="text" name="yourname" maxlength="150" /><br />
<input type="submit" value="Submit" style="margin-top: 10px;" />
</form>
</body>
</html>
```

The form in the previous block of code will then call the processing statement as follows:

```
<?php
if (trim ($_POST['yourname']) == ""){
header ("Location: sample12_5.html");
exit;
}
echo $_POST['yourname'];
?>
```

The header() function is rather nice in that it will redirect you automatically to the appropriate file (providing it exists) without a single hiccup in the processing. You will simply find yourself at the appropriate page. You can even use the header() function with the Locationparameter to send you to a page not currently on the server on which the script is located.

As such, this functionality can be rather effective even as a simple page redirection script.

Content Type Application

application/pdf Adobe Portable Document Format (PDF) types
application/msword Microsoft Word documents
application/excel Microsoft Excel documents
image/gifGIF images
image/png PNG images
application/octet-stream Zip files
text/plainPlain text (text files)

Forcing File "Save As"Downloads

Because web browsers can output many different file types directly onto the screen, the default when you use headers to output a wide variety of file types is to make them automatically appear on the screen. What if you would rather have the file appear as a download, though? You can use the header() function to force a Save As dialog box to appear for the user to accept a download.

The following example uses largely the same code as the previous example but instead forces the user to download the file.
Eg:

```
<?php
$path = "images/winter.jpg";
try {
if (is_file ($path)){
if ($file = fopen($path, 'rb')) {
while(!feof($file) and (connection_status()==0)) {
$f .=fread($file, 1024*8);
}
fclose($file);
}
$outputname = "myimage";
header ("Content-type: image/jpeg");
header("Content-disposition: attachment; filename=".$outputname.".jpg");
print $f;
} else {
throw new exception ("Sorry, file path is not valid.");
}
} catch (exception $e){
echo $e->getmessage();
}
?>
```

The key point in this code is showing content-disposition in the header. By making content-disposition an attachment value, the browser will force a download rather than display the file inline. By using this, you can force the download to appear with any particular filename you prefer and also with pretty much any file extension. By using content-type, you force the browser to output a file of the requested type.

16) Authenticating Your Users:

No matter what type of online application you are building, if you need to keep sections of it private, you will at some point need to create a way of authenticating your users so that you know you have a valid user accessing the site. You can handle authentication in a variety of ways, but the two most common methods for securing a file or set of files is through HTTPbasedauthentication and through cookie authentication. Neither is technically superior to the other, and they both have their own uses. Both can be set up dynamically, and both will stop users in their tracks should they not meet the authenticated demands.

Setting Up HTTP-Based Authentication:

HTTP-based authentication can be a true challenge from a scripting point of view. The interesting part about it is that most server interfaces (such as Cpanel or Ensim) can create HTTPbasedauthentication on the fly. In this case, we have written a class to do this for you.

We are not the biggest fans of HTTP-based authentication because the login mechanism is largely the same. You can set a few variables to customize it slightly, but in the end, it is the same pop-up window asking for your username and password. That being said, this class lets you handle the authentication on the fly.

For this code to work properly, you must first set up a file called .ht access and ensure that you set the proper path to it when calling the class.

Setting Up Cookie Authentication:

Managing user authentication through cookies or sessions is a little harder than using TTPbasedauthentication, but it can ultimately be more flexible and rewarding. Some of the nice features of cookie-based authentication are being able to set your own error messages, being able to control what happens upon login, and being allowed to make your login form blend seamlessly into your application (rather than being forced to use the pop-up boxes of the HTTP-based variety).

Two schools of thought exist on the whole cookie vs. sessions issue; the advantages of sessions being kept on the server side and working on any platform outweigh the cookie method's advantage of being slightly more flexible. By using sessions you will know that your script should work on pretty much any platform and will be a reliable, secure way of handling authentication.

17) Using Environment and Configuration Variables:

PHP provides a means to use and verify the configuration settings and environment variables relative to the server space the script is occupying. Having access to this feature set can come in handy on many occasions. By having access to environment variables, you can customize your scripts to work optimally on the platform that is available. By having access to the configuration variables of PHP, you can customize the PHP environment your script is working in for special occurrences.

A common use of the environment variables in PHP is for dynamic imaging. While Windows systems commonly store their fonts in one folder, Linux-based systems keep theirs in another. By using PHP's environment variables to determine the current operating system, you can make your code slightly more portable.Using configuration variables can also come in quite handy, particularly with file uploadscripts. The base PHP installation leaves only enough processing time to upload files that are generally 2MB or smaller in size. By manipulating the PHP configuration files temporarily, you can increase the limit enough to allow a script to process much larger files.

Reading Environment and Configuration Variables:

PHP 5 makes reading environment and configuration variables easy. The $_ENV superglobalis PHP's method for reading a system's environment variables and has an argument set that is based upon the current environment that is available to it. Because of its relative flexibility, there is no real set argument list, as it is generated based on the current server environment.

You can use the phpinfo() function to determine the current environment variables, and you can retrieve them using the getenv() function, which needs to be supplied a valid environment variable name.

Reading configuration variables, on the other hand, takes place through two functions, ini_get() and ini_get_all(). The function ini_get() will retrieve the value of a specified configuration variable, and the function ini_get_all() will retrieve an array filled with the entire selection of configuration variables that are available.

The following example shows how to retrieve both environment and configuration variables.

Eg:

```
<?php
echo $_ENV['ProgramFiles'] . "<br />"; //Outputs C:\Program Files.
echo $_ENV['COMPUTERNAME'] . "<br />"; //Outputs BABINZ-CODEZ.
echogetenv("COMPUTERNAME") . "<br />"; //Also Outputs BABINZ-CODEZ.
print_r (ini_get_all());
?>
```

There is really no problem when reading environment and configuration variables. You can get the job done in a bunch of ways, and predefined functions exist in allaspects of PHP to take care of any issue you may encounter.

Setting Environment and Configuration Variables:

Setting environment and configuration variables is just as easy as it is to get them. While working with environment variables, you merely need to assign a new value to the $_ENVsuperglobal to process a temporary change. The change will be in effect for the script's duration.

The same applies for configuration variables but with a different approach. To set a configuration variable, you have to use the PHP function ini_set(), which will allow you to set a configuration variable for the script's duration. Once the script finishes executing, the configuration variable will return to its original state.

The prototype for ini_set() is as follows:

```
stringini_set( string varname, string newvalue )
```

Example:

```
<?php
echo $_ENV['COMPUTERNAME'] . "<br />"; $_ENV['COMPUTERNAME'] = "Hello World!";
echo $_ENV['COMPUTERNAME'] . "<br />";
echoini_get ('post_max_size');
ini_set('post_max_size','200M');
?>
```

Setting environment and configuration variables are a rather simple task. It can be a handy task, and it can help you modify the current environment to work for you. Many times in your coding career you will have to code around a certain server's configuration. By combining a means to analyze your environment and a means to subsequently work with it, PHP ensures that your scripts will be able to operate to their fullest.

Unit-4

Creating and Using Forms & XML

Understanding Common Form Issues

When dealing with forms, the most important aspect to remember is that you are limited to a certain variety of fields that can be applied to a form. The fields that have been created are non-negotiable and work in only the way they were created to work. It is important, therefore, to fully understand what is available and how best to use the form features to your advantage.

HTML Form Elements

Element Description

TEXT INPUT	→ A simple text box
PASSWORD INPUT	→ A text box that hides the characters inputted
HIDDEN INPUT	→A field that does not show on the form but can contain data
SELECT	→ A drop-down box with options
LIST	→ A select box that can have multiple options selected
CHECKBOX	→A box that can be checked
RADIO	→ A radio button that can act as a choice
TEXTAREA	→ A larger box that can contain paragraph-style entries
FILE	→ An element that allows you to browse your computer for a file
SUBMIT	→ A button that will submit the form
RESET	→ A button that will reset the form to its original state

1. **GET vs. POST**

When dealing with forms, you must specify the way that the information entered into the form is transmitted to its destination (method=""). The two ways available to a web developer are GET and POST. When sending data using the GET method, all fields are appended to the

Uniform Resource Locator (URL) of the browser and sent along with the address as data. With the POST method, values are sent as standard input. Sending data using the GET method means that fields are generally capped at 150 characters, which is certainly not the most effective means of passing information. It is also not a secure means of passing data, because many people know how to send information to a script using an address bar.

Sending data using the POST method is quite a bit more secure (because the method cannot be altered by appending information to the address bar) and can contain as much information as you choose to send. Therefore, whenever possible, use the POST method for sending information and then adjust your script to handle it.

PHP 5's current methods for dealing with GET and POST variables are the $_GET and $_POST superglobals, respectively. By using these two superglobals, you can designate exactly where the information should be coming from and subsequently handle the data in the way you want. The following example shows the difference between using the GET and POST methods.

Example:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"➥

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<title>Sample 13.1</title>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />

</head>

<body>

<div style="width: 500px; text-align: left;">

<?php

if ($_GET['submitted'] == "yes"){

if (trim ($_GET['yourname']) != ""){

echo "Your Name (with GET): " . $_GET['yourname'];

} else {

echo "You must submit a value.";
```

```
}
?><br /><a href="sample13_1.php">Try Again</a><?php
}
if ($_POST['submitted'] == "yes"){
if (trim ($_POST['yourname']) != ""){
echo "Your Name (with POST): " . $_POST['yourname'];
} else {
echo "You must submit a value.";
}
?><br /><a href="sample13_1.php">Try Again</a><?php
}
?>
<?php
if ($_GET['submitted'] != "yes" && $_POST['submitted'] != "yes"){
?>
<form action="sample13_1.php" method="get">
<p>GET Example:</p>
<input type="hidden" name="submitted" value="yes" />
Your Name: <input type="text" name="yourname" maxlength="150" /><br />
<input type="submit" value="Submit with GET" style="margin-top: 10px;" />
</form>
<form action="sample13_1.php" method="post">
<p>POST Example:</p>
<input type="hidden" name="submitted" value="yes" />
Your Name: <input type="text" name="yourname" maxlength="150" /><br />
```

```
<input type="submit" value="Submit with POST" style="margin-top: 10px;" />

</form>

<?php

}

?>

</div>

</body>

</html>
```

This block of code demonstrates the difference between the GET and POST methods using the two different forms. You should remember a few things when using such code. Specifically, try hitting the Refresh button after submitting data using the POST form. You will note that the browser will ask you if you want to resubmit the data that was passed to it previously. If you want to resend the data, you must select Yes to this option. On the other hand, when using the

GET method, you will not be presented with this issue. (The browser will automatically send the data again.)

Other than a mild bit of validation, this script is pretty simple. It receives either a POST method or a GET method submission of a text field and then displays it if it is not an empty field. Note that because you are using the $_POST and $_GET superglobals, you can determine from where the information is coming. Although each form has a field called submitted, the script knows which value to display based upon the way the information was passed to it.

- **Superglobals vs. Globals**

Before the advent of superglobals, data was passed along from script to script with loose security.

In the php.ini file, you can change a value called register_globals to either on or off.

If you leave it on, then whenever you pass a value using the GET or POST method, you can access the variable simply by putting an ampersand (&) character in front of the name of the element you are passing. The problem with this method is that malicious users can insert values into your code to bypass the form entirely.

Therefore, if you want your code to be as secure as possible (and who doesn't?), you should definitely code your applications with register_globals turned off and ensure that you receive your values from where you expect them to come. Using superglobals allows you to do this. The following example shows how you can submit values using globals or superglobals.

Note that for this example to work properly, you must temporarily switch your register_globals value to on (don't forget to turn it off afterward!).

eg:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"➥
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<title>Sample 13.2</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
</head>
<body>
<div style="width: 500px; text-align: left;">
<?php
if ($submitted == "yes"){
if (trim ($yourname) != ""){
echo "Your Name: $yourname.";
} else {
echo "You must submit a value.";
}
?><br /><a href="sample13_2.php">Try Again</a><br /><?php
}
if ($_POST['submitted'] == "yes"){
if (trim ($_POST['yourname']) != ""){
echo "Your Name: " . $_POST['yourname'] . ".";
```

```
} else {

echo "You must submit a value.";

}

?><br /><a href="sample13_2.php">Try Again</a><br /><?php

}

?>

<?php

if ($_POST['submitted'] != "yes"){

?>

<form action="sample13_2.php" method="post">

<p>Example:</p>

<input type="hidden" name="submitted" value="yes" />

Your Name: <input type="text" name="yourname" maxlength="150" /><br />

<input type="submit" value="Submit" style="margin-top: 10px;" />

</form>

<?php

}

?>

</div>

</body>

</html>
```

The $_POST superglobal looks pretty much identical, minus the $_POST preceding the field name. The interesting thing to note is that if you run this code with register_globals turned on, both scripts will fire. If, however, you run this code with register_globals turned off, only the second script will fire. Now, consider how easily someone could inject some code into the first script and potentially change the received value. Since the script would not recognize where the value is coming from, it could be easily intercepted. Using the second script, the value passed has to be the one coming from the $_POST

superglobal. It should become common practice to code only with register_globals turned off to create as secure an application as possible.

2. **Validating Form Input**

In this day and age of constant attacks on websites, one of the biggest issues is attacking forms directly. To ensure a suitable submission of form data, validation is key. You have many ways to validate a form and many form elements to consider. Generally, you need to determine what qualities you want a piece of data to adhere to and then ensure that the submitted data comes in the correct form. If the data comes in a format that is not to your liking, you must be ready to take care of this. The following example shows a few examples of form validation using PHP.

Eg:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"➥

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<title>Sample 13.3</title>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />

</head>

<body>

<div style="width: 500px; text-align: left;">

<?php

function validemail($email){

return preg_match("/^([a-zA-Z0-9])+([.a-zA-Z0-9_-])*@([a-zA-Z0-9_-])➥

+(.[a-zA-Z0-9_-]+)+[a-zA-Z0-9_-]$/",$email);

}

if ($_POST['submitted'] == "yes"){

$goodtogo = true;

try {

if (trim ($_POST['yourname']) == ""){

$goodtogo = false;
```

```
throw new exception ("Sorry, you must enter your name.<br />");

}

} catch (exception $e) {

echo $e->getmessage();

}

try {

if ($_POST['myselection'] == "nogo"){

$goodtogo = false;

throw new exception ("Please make a selection.<br />");

}

} catch (exception $e) {

echo $e->getmessage();

}

try {

if (!validemail (trim ($_POST['youremail']))){

$goodtogo = false;

throw new exception ("Please enter a valid email address.<br />");

}

} catch (exception $e) {

echo $e->getmessage();

}

if ($goodtogo){

echo "Your Name: " . $_POST['yourname'] . "<br />";

echo "Your Selection: " . $_POST['myselection'] . "<br />";

echo "Your Email Address: " . $_POST['youremail'] . "<br />";
```

```
}
?><br /><a href="sample13_3.php">Try Again</a><br /><?php
}
?>
<?php
if ($_POST['submitted'] != "yes"){
?>
<form action="sample13_3.php" method="post">
<p>Example:</p>
<input type="hidden" name="submitted" value="yes" />
Your Name: <input type="text" name="yourname" maxlength="150" /><br /><br />
Selection:
<select name="myselection">
<option value="nogo">make a selection...</option>
<option value="1">Choice 1</option>
<option value="2">Choice 2</option>
<option value="3">Choice 3</option>
</select><br /><br />
Your Email: <input type="text" name="youremail" maxlength="150" /><br />
<input type="submit" value="Submit" style="margin-top: 10px;" />
</form>
<?php
}
?>
</div>
```

```
</body>

</html>
```

For this example, you have chosen three types of fields, it is important to take care of them in individual ways. For this example, you want to receive a name value that will not be blank, a selected value that must not be the default, and an e-mail address that must be in the proper format. To make sure you do not have a blank field, you can validate the name value by ensuring that it does not equal a blank string. In the case of the selection, if the user has not chosen a different value than the default, the value will be a nogo, against which you can then validate. For the last value, the e-mail address, you use a regular expression to ensure that the e-mail address is properly formatted. By using this type of validation, you ensure that all the submitted values are in the format you need.

Working with Multipage Forms

Sometimes you will need to collect values from more than one page. Most developers do this for the sake of clarity. By providing forms on more than one page, you can separate blocks of information and thus create an ergonomic experience for the user. The problem, therefore, is how to get values from each page onto the next page and finally to the processing script. Being the great developer that you are, you can solve this problem and use the hidden input form type. When each page loads, you merely load the values from the previous pages into hidden form elements and submit them.

Eg:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"➥

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<title>Sample 13.4 Page 1</title>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />

</head>

<body>

<div style="width: 500px; text-align: left;">

<form action="sample13_4_page2.php" method="post">

<p>Page 1 Data Collection:</p>
```

```
<input type="hidden" name="submitted" value="yes" />
Your Name: <input type="text" name="yourname" maxlength="150" /><br /><br />
<input type="submit" value="Submit" style="margin-top: 10px;" />
</form>
</div>
</body>
</html>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"➥
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<title>Sample 13.4 Page 2</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
</head>
<body>
<div style="width: 500px; text-align: left;">
<form action="sample13_4_page3.php" method="post">
<p>Page 2 Data Collection:</p>
Selection:
<select name="yourselection">
<option value="nogo">make a selection...</option>
<option value="1">Choice 1</option>
<option value="2">Choice 2</option>
<option value="3">Choice 3</option>
</select><br /><br />
<input type="hidden" name="yourname"  value="<?php echo $_POST['yourname']; ?>" />
```

```
<input type="submit" value="Submit" style="margin-top: 10px;" />
</form>
</div>
</body>
</html>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<title>Sample </title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
</head>
<body>
<div style="width: 500px; text-align: left;">
<form action="sample13_4_page4.php" method="post">
<p>Page 3 Data Collection:</p>
Your Email: <input type="text" name="youremail" maxlength="150" /><br />
<input type="hidden" name="yourname" value="<?php echo $_POST['yourname']; ?>" />
<input type="hidden" name="yourselection" value="<?php echo _POST['yourselection']; ?>" />
<input type="submit" value="Submit" style="margin-top: 10px;" />
</form>
</div>
</body>
</html>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"➥
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml">

<title>Sample 13.4 Page 4</title>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />

</head>

<body>

<div style="width: 500px; text-align: left;">

<?php

//Display the results.

echo "Your Name: " . $_POST['yourname'] . "<br />";

echo "Your Selection: " . $_POST['yourselection'] . "<br />";

echo "Your Email: " . $_POST['youremail'] . "<br />";

?>

<a href="sample13_4_page1.php">Try Again</a>

</div>

</body>

</html>
```

By passing the values in the hidden form fields, you can continue to collect information. In a real-world example, you most certainly want to perform validation to make sure you have all the information you need at every point in the script. In any case, if you follow the flow of the script, you will see that on each subsequent page the values from the previous pages are included and hence displayed once the final display page is reached.

3. Preventing Multiple Submissions of a Form

One possible occurrence that happens often is that users become impatient when waiting for your script to do what it is doing, and hence they click the submit button on a form repeatedly.

This can wreak havoc on your script because, while the user may not see anything happening, your script is probably going ahead with whatever it has been programmed to do.

Of particular danger are credit card number submittals. If a user continually hits the submit button on a credit card submittal form, their card may be charged multiple times if the developer has not taken the time to validate against such an eventuality.

Preventing Multiple Submissions on the Server Side

we can deal with multiple submittal validation in essentially two ways. The first occurs on the server. *Server side* refers to a script located on the server that is receiving the data; *client side* is more browser related (and explained in the next example). Because the server has no actual access to the browser, validating multiple submissions can be a bit trickier. While you can accomplish this goal in a number of ways from a server-side perspective, we prefer to use a session-based method. Basically, once the submit button has been clicked, the server logs the request from the individual user. If the user attempts to resubmit a request, the script notes a request is already in motion from this user and denies the subsequent request. Once the script has finished processing, the session is unset, and you have no more worries.

For the following example, you will need a test.txt text file that you can create and place relative to the script. (Or you can ensure that you have write privileges on the working directory, and the script will attempt to create it for you.) Keep in mind that the file must have the proper privileges set for writing (CHMOD to 777 to keep things simple).

Eg:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<title>Sample 13.6</title>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />

</head>

<body>

<div style="width: 500px; text-align: left;">

<form action="sample13_6_process.php" method="post">

<p>Example:</p>

<input type="hidden" name="submitted" value="yes" />
```

```
Your Name: <input type="text" name="yourname" maxlength="150” /><br />

<input type="submit" value="Submit" style="margin-top: 10px;" />

</form>

</div>

</body>

</html>

<?php

session_start ();

if (!isset ($_SESSION['processing'])){

$_SESSION['processing'] = false;

}

if ($_SESSION['processing'] == false){

$_SESSION['processing'] = true;

for ($i = 0; $i < 2000000; $i++){

}

if ($file = fopen ("test.txt","w+")){

fwrite ($file, "Processing");

} else {

echo "Error opening file.";

}

echo $_POST['yourname'];

unset ($_SESSION['processing']);

}

?>
```

Enter your name and continue to jam on the submit button. Rather than allow the script to continually run time and time again, the script verifies your existence via a session and determines if it is already

processing your server call. If the script sees you are already processing, then it will not allow you to try again no matter how many times you click the same button.

Once the script has finished performing its action, it merely unsets the session variable, and you could theoretically start again. By checking the session, the script ensures that it is the same user attempting to access the script and can therefore block multiple attempts from the same user.

Preventing Multiple Submissions on the Client Side:

Handling multiple submittals from a client-side perspective is actually much simpler than doing it on the server side. With well-placed JavaScript, you can ensure that the browser will not let the submittal go through more than once. The problem with this method, of course, is that JavaScript is not always foolproof because of the user's ability to turn it off. That being said, however, most users will have JavaScript enabled, so this script will likely work for 90 percent of web users. The following example uses JavaScript to cut off multiple submittals from a client-side (browser) level.

Don't forget to ensure that you have a valid test.txt file (CHMOD to 777), as specified in the previous recipe.

Eg:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<title>Sample 13.7</title>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />

<script language="javascript" type="text/javascript">

<!--

function checkandsubmit() {

document.test.submitbut.disabled = true;

document.test.submit();

}

</script>
```

```
</head>

<body>

<div style="width: 500px; text-align: left;">

<form action="sample13_6_process.php" method="post" name="test" onsubmit="return
checkandsubmit ()">

<p>Example:</p>

<input type="hidden" name="submitted" value="yes" />

Your Name: <input type="text" name="yourname" maxlength="150" /><br />

<input type="submit" value="Submit" style="margin-top: 10px;" id="submitbut" name"submitbut" />

</form>

</div>

</body>

</html>

<?php

for ($i = 0; $i < 2000000; $i++){

}

if ($file = fopen ("test.txt","w+")){

fwrite ($file, "Processing");

} else {

echo "Error opening file.";

}

echo $_POST['yourname'];

?>
```

XML

What Is XML?

XML stands for Extensible Markup Language. In the context of programming, extensible means that the developer can define the markup elements.XML is, on a basic level, extremely simple. In a parallel with HTML, XML information is enclosed in tags; however, you can name the tags anything you want, and these tags are basically labels for the enclosed information. For example, assume you want to send personal information about someone via XML. The markup might read as follows:

<person>

<name>Jason</name>

<age>24</age>

<gender>male</gender>

</person>

The benefit of XML is that it's easy to read.

Example 2

<?xml version="1.0" standalone="yes"?>

<library>

<book>

<title>Pride and Prejudice</title>

<author gender="female">Jane Austen</author>

<description>Jane Austen's most popular work.</description>

</book>

<book>

<title>The Conformist</title>

<author gender="male">Alberto Moravia</author>

<description>Alberto Moravia's classic psychological novel.</description>

</book>

</ library>

AJAX

AJAX stands for **A**synchronous **J**avaScript **a**nd **X**ML. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script. AJAX is not a programming language. AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

- Ajax uses XHTML for content, CSS for presentation, along with Document Object Model and JavaScript for dynamic content display.
- Conventional web applications transmit information to and from the sever using synchronous requests. It means you fill out a form, hit submit, and get directed to a new page with new information from the server.
- With AJAX, when you hit submit, JavaScript will make a request to the server, interpret the results, and update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server.
- XML is commonly used as the format for receiving server data, although any format, including plain text, can be used.
- AJAX is a web browser technology independent of web server software.
- Classic web pages, (which do not use AJAX) must reload the entire page if the content should change. AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

AJAX is based on open Standards: AJAX is based on the following open standards –

- Browser-based presentation using HTML and Cascading Style Sheets (CSS).
- Data is stored in XML format and fetched from the server.
- Behind-the-scenes data fetches using XMLHttpRequest objects in the browser.
- JavaScript to make everything happen.

AJAX cannot work independently. It is used in combination with other technologies to create interactive WebPages.

JavaScript:

- Loosely typed scripting language.
- JavaScript function is called when an event occurs in a page.
- Glue for the whole AJAX operation.

DOM:

- API for accessing and manipulating structured documents.
- Represents the structure of XML and HTML documents.

CSS:

- Allows for a clear separation of the presentation style from the content and may be changed programmatically by JavaScript

XMLHttpRequest:

- JavaScript object that performs asynchronous interaction with the server.

Here is a **list of some famous web applications that make use of AJAX.**

Google Maps:

A user can drag an entire map by using the mouse, rather than clicking on a button.

Google Suggest:

As you type, Google offers suggestions. Use the arrow keys to navigate the results.

Gmail:

Gmail is a webmail built on the idea that emails can be more intuitive, efficient, and useful.

Yahoo Maps (new):

Now it's even easier and more fun to get where you're going!

All the available browsers cannot support AJAX. Here is a **list of major browsers** that support AJAX.

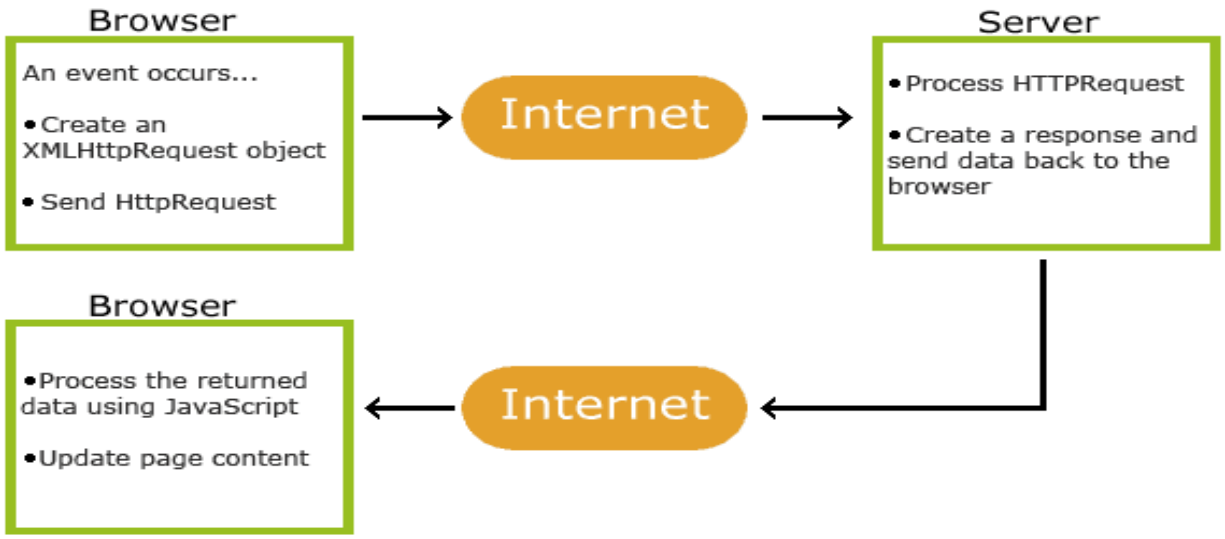
• Mozilla Firefox 1.0 and above. • Netscape version 7.1 and above.	• Apple Safari 1.2 and above. • Microsoft Internet Explorer 5 and above. • Opera 7.6 and above.

NOTE − When we say that a browser does not support AJAX, it simply means that the browser does not support the creation of Javascript object – XMLHttpRequest object.

How AJAX Works:

Steps of AJAX Operation**:**

- A client event occurs.
- An XMLHttpRequest object is created.
- The XMLHttpRequest object is configured.
- The XMLHttpRequest object makes an asynchronous request to the Webserver.
- The Webserver returns the result containing XML document.
- The XMLHttpRequest object calls the callback () function and processes the result.
- The HTML DOM is updated.
- The response is read by JavaScript
- Proper action (like page update) is performed by JavaScript



AJAX Security: Server Side

- AJAX-based Web applications use the same server-side security schemes of regular Web applications.
- You specify authentication, authorization, and data protection requirements in your web.xml file (declarative) or in your program (programmatic).
- AJAX-based Web applications are subject to the same security threats as regular Web applications.

AJAX Security: Client Side

- JavaScript code is visible to a user/hacker. Hacker can use JavaScript code for inferring server-side weaknesses.
- JavaScript code is downloaded from the server and executed ("eval") at the client and can compromise the client by mal-intended code.
- Downloaded JavaScript code is constrained by the sand-box security model and can be relaxed for signed JavaScript.

AJAX Benefits

- Before AJAX, the client side of a web application could not communicate directly with the server without refreshing the page.
- AJAX makes this possible. AJAX allows the client and server of a web application to freely communicate with each other.
- The primary purpose of AJAX is to modify the part of the web page already displayed in the browser's window without reloading the entire page.

Asynchronous and synchronous Requests:

The ***asynchronous flag*** is typically set to true. It makes the request asynchronous. In such a case, the browser can continue with normal execution. The user can interact with browser even if the response is not completed. When the response comes back, a Java Script function is executed, where we can process the data sent by the server.

The ***asynchronous flag*** is set to false (***synchronous request)*** leaves the browser in a waiting state until the response comes from the server and the service routine is executed. During this time, the user cannot interact with browser.

Ajax Client Server Architecture:

Classic web application reloading the page for every user interaction:

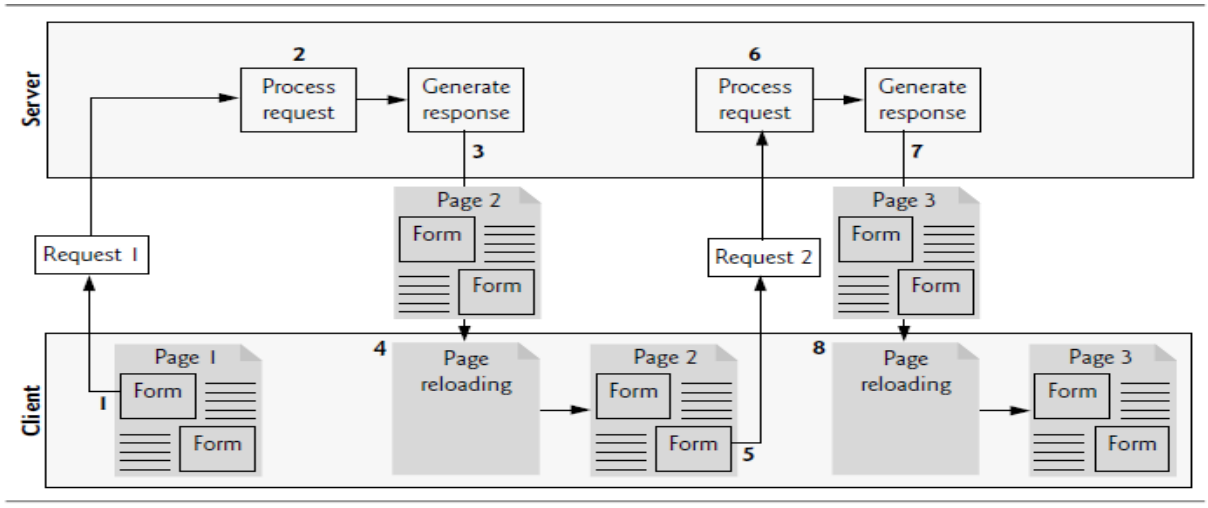


Fig. 16.1 | Classic web application reloading the page for every user interaction.

Ajax-enabled web application interacting with the server asynchronously:

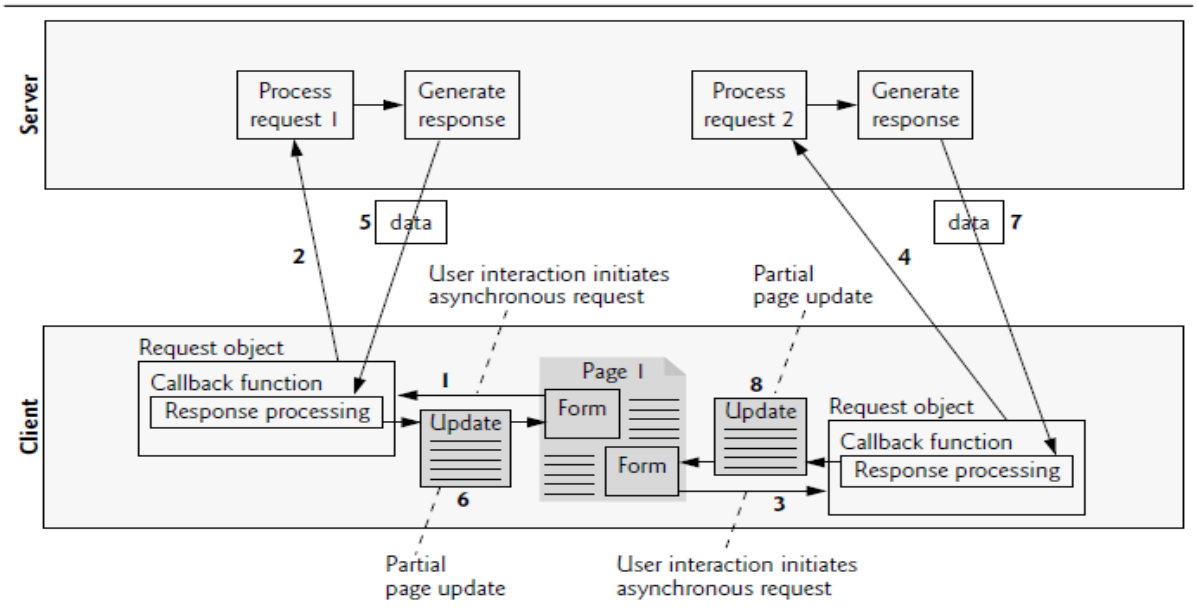
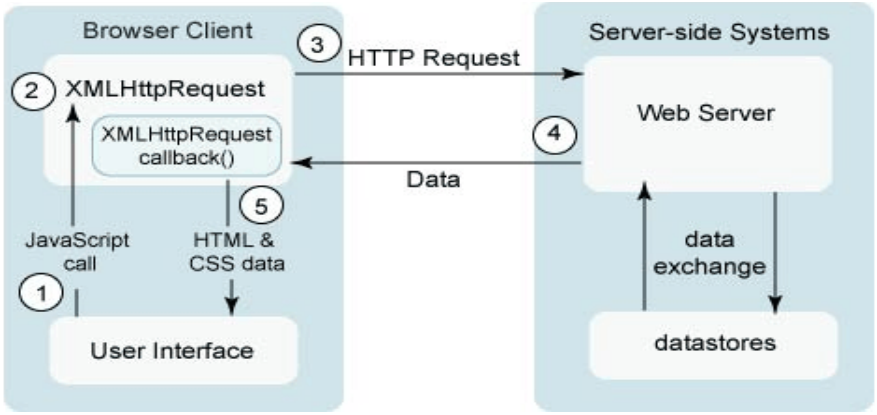


Fig. 16.2 | Ajax-enabled web application interacting with the server asynchronously.



The XMLHttpRequest Object/call back methods:

All modern browsers support the XMLHttpRequest object.The XMLHttpRequest object can be used to exchange data with a server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

Create an XMLHttpRequest Object:

The heart of the AJAX technology is the XMLHttpRequest object. It provides a set of useful properties and methods that are used to send HTTP request to and retrieve data from the web server. All modern browsers (Chrome, Firefox, IE7+, Edge, Safari, and Opera) have a built-in XMLHttpRequest object.Simply use the XMLHttpRequest() constructor as follows:

Syntax for creating an XMLHttpRequest object:	**Example:**
variable = new XMLHttpRequest();	var xhttp = new XMLHttpRequest();

Old Versions of Internet Explorer (IE5 and IE6):

Old versions of Internet Explorer (IE5 and IE6) use an ActiveX object instead of the XMLHttpRequest object: variable = new ActiveXObject("Microsoft.XMLHTTP");
variable= new ActiveXObject("Msxml2.XMLHTTP");

The simplest way to make your source code compatible with a browser is to use *try...catch* blocks in your JavaScript.

```
<html>
  <body>
    <script language="javascript" type="text/javascript">
      <!--
      //Browser Support Code
      function ajaxFunction() {
        var ajaxRequest;  // The variable that makes Ajax possible!
        try {
          // Opera 8.0+, Firefox, Safari
          ajaxRequest=newXMLHttpRequest();
        } catch (e) {
          // Internet Explorer Browsers
          try {

ajaxRequest=newActiveXObject("Msxml2.XMLHTTP");
          } catch (e) {
```

```
try {
ajaxRequest=newActiveXObject("Microsoft.XMLHTTP");
            } catch (e) {
              // Something went wrong
              alert("Your browser broke!");
              return false;
            }
          }
        }
      }
    //-->
    </script>
      <form name = 'myForm'>
   Name: <input type = 'text' name = 'username' /> <br />
      Time: <input type = 'text' name = 'time' />
    </form>
      </body>
</html>
```

Specify a Handler:

When we use an asynchronous request, we cannot be sure when the response will come back.so,before we even think about sending the HTTP request to the web server, we must first specify a function to be called when the HTTP response from the server comes back. We call this

function"response handler". This function will receive information returned by the server,extract the desired data, and perform a specific task. The response handler is specified using the onreadystatechange property of the xmlHttp object**.**

Example:

```
// Create a function that will receive data sent by the server
function handler(){
//code to extract and use data sent by the server
}
//specify the function handler () as the response handler
 xmlHttp.onreadystatechange=handler;
```

The property onreadystatechange stores a function.As the name implies,every time the "ready state"changes this function gets executed.

AJAX readyState property:

The process of communicating with the server,from sending an HTTP request to getting a response and involves several states. The state can be one of the following:

Uninitialized,connection established,request sent,processing,completed,and response is ready.

The xmlHttp object has a propery called readystate,where the state of the HTTP request to be sent is stored.Each state is identified by appositive integer.

Every time the state changes, the function indicated by the onreadystatechange property gets executed.

The following table provides a **list of the possible values for the readyState property** :

Values	State	Description
0	The request is not initialized/ un-initialized	After creating the XMLHttpRequest object
1	The request has been set up/ connection established	After calling the open() method
2	The request has been sent/request sent	After calling the send() method
3	The request is in process/processing	After calling the send() method but before getting the reponse
4	The request is completed/ completed and response is ready	After the request has been completed and response have been completely received from the server

Example:

```
 Function handler() {
if(xmlHttp.readystate==4){
//get the data from the server response
}
else{
//request is not yet completed
   }}
```

XMLHttpRequest Properties:

- **Onreadystatechange:** An event handler for an event that fires at every state change**.**
- **readyState:** The readyState property defines the current state of the XMLHttpRequest object.
- **responseText:**Returns the response as a string.
- **responseXML:**Returns the response as XML. This property returns an XML document object, which can be examined & parsed using W3C DOM node tree methods & properties.
- **Status:**Returns the status as a number (e.g., 404 for "Not Found" and 200 for "OK").
- **statusText:**Returns the status as a string (e.g., "Not Found" or "OK").

Sending Information:

Now that the response handler is ready,we can send the HTTP request to the server. Sending a request consists of two steps:

- Opening a connection
- sending an HTTP request

Opening a connection:

Before sending an HTTP connection, we need to establish a TCP connection to the webserver.The object xmlHttp has several methods to open a TCP connection to the server as follows:

open(method, URL)
open(method, URL, async)
open(method, URL, async, userName)
open(method, URL, async, userName, password)

The most commonly used version is the second one, which takes three arguments, Specifies the method, URL, and other optional attributes of a request. The method parameter can have a value of "GET", "POST", or "HEAD". Other HTTP methods such as "PUT" and "DELETE" (primarily used in REST applications) may be possible.The "async" parameter specifies whether the request should be handled asynchronously or not. "true" means that the script processing carries on after the send() method without waiting for a response, and "false" means that the script waits for a response before continuing script processing.

Example: xmlHttp.open("GET","getResult.jsp",true);

Sending an HTTP request: Once the TCP connection is established,we can send the HTTP request to the server through this connection.The request is actually sent using the send()method on the xmlHttp object. It is possible to send data in other formats such as images,xml documents using the POST method.

Example: xmlHttp.send(null);

Method Type: Several methods are specified in the HTTP specification, the most commonly supported and used methods are HEAD, GET and POST.

Example: xmlHttp.open("HEAD","AJAXDemo.jsp",true);

If the **HEAD** method is specified in the HTTP request, only the meta information about the URL is returned by the web server.so, if you want to get header information but not the resource body and use this method.

The HTTP **GET** method allows us to send parameters to the HTTP server as a part of the URL. The parameters are sent as name=value pairs separated by &.

Example: xmlHttp.open("GET","getResult.jsp?roll=1001&year=2009",true);

In the **POST** method the parameters are not passed as a part of the URL and rather they are passed as a separate entity.

Example: xmlHttp.open("POST","update.jsp",true);

XMLHttpRequest Object Methods:

- **open(method, URL)**
- **open(method, URL, async)**
- **open(method, URL, async, userName)**
- **open(method, URL, async, userName, password)**
- **abort():**Cancels the current request.
- **getAllResponseHeaders():**Returns the complete set of HTTP headers as a string.
- **getResponseHeader (headerName):**Returns the value of the specified HTTP header.
- **send(content):**Sends the request.
- **setRequestHeader (label, value):**Adds a label/value pair to the HTTP header to be sent.
- **Void send ():** sends get request.
- **Void send (string):** send post request.

Web Services

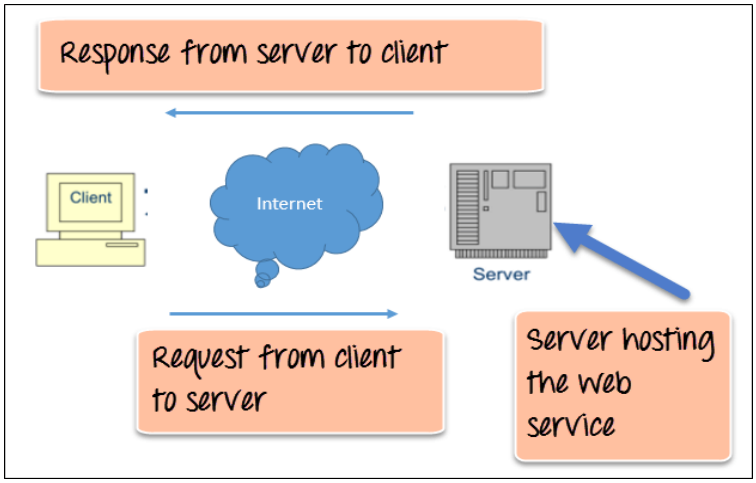
What is Web Service?

Web service is a standardized medium to propagate communication between the client and server applications on the World Wide Web.

A web service is a software module which is designed to perform a certain set of tasks.

- The web services can be searched for over the network and can also be invoked accordingly.

- When invoked the web service would be able to provide functionality to the client which invokes that web service.



The above diagram shows a very simplistic view of how a web service would actually work. The client would invoke a series of web service calls via requests to a server which would host the actual web service.
These requests are made through what is known as remote procedure calls. Remote Procedure Calls (RPC) is calls made to methods which are hosted by the relevant web service.

Type of Web Service:
Web service to be fully functional, there are certain components that need to be in place. These components need to be present irrespective of whatever development language is used for programming the web service.There are **mainly four types** of web service components:
SOAP :(Simple Object Access Protocol)
WSDL :(Web Services Description Language)
UDDI :(Universal Description, Discovery and Integration)
SOA :(Service-Oriented Architecture)

SOAP:

- SOAP is an acronym for Simple Object Access Protocol.
- SOAP is a XML-based protocol for accessing web services.
- SOAP is a W3C recommendation for communication between two applications.
- SOAP is XML based, so it is platform independent and language independent. In other words, it can be used with Java, .Net or PHP language on any platform.

Advantages of Soap Web Services:
WS Security: SOAP defines its own security known as WS Security.
Language and Platform independent: SOAP web services can be written in any programming language and executed in any platform.
Disadvantages of Soap Web Services:
Slow: SOAP uses XML format that must be parsed to be read. It defines many standards that must be followed while developing the SOAP applications. So it is slow and consumes more bandwidth and resource.
WSDL dependent: SOAP uses WSDL and doesn't have any other mechanism to discover the service.
WSDL:

- WSDL is an acronym for Web Services Description Language.
- WSDL is a xml document containing information about web services such as method name, method parameter and how to access it.
- WSDL is a part of UDDI. It acts as a interface between web service applications.
- WSDL is pronounced as wiz-dull.

UDDI:

- UDDI is an acronym for Universal Description, Discovery and Integration.
- UDDI is a XML based framework for describing, discovering and integrating web services.
- UDDI is a directory of web service interfaces described by WSDL, containing information about web services.

Service Oriented Architecture (SOA):

Service Oriented Architecture or SOA is a design pattern. It is designed to provide services to other applications through protocol. It is a concept only and not tied to any programming language or platform.Web services is a technology of SOA most likely.

Service

A service is well-defined, self-contained function that represents unit of functionality. A service can exchange information from another service. It is not dependent on the state of another service.

Service Connections

The figure given below illustrates the service oriented architecture. Service consumer sends service request to the service provider and service provider sends the service response to the service consumer. The service connection is understandable to both service consumer and service provider.

RESTful Web Services:

REST stands for REpresentational State Transfer.

REST is an architectural style not a protocol.

Advantages of RESTful Web Services:

Fast: RESTful Web Services are fast because there is no strict specification like SOAP. It consumes less bandwidth and resource.

Language and Platform independent: RESTful web services can be written in any programming language and executed in any platform.

Can use SOAP: RESTful web services can use SOAP web services as the implementation.

Permits different data format: RESTful web service permits different data format such as Plain Text, HTML, XML and JSON.

SOAP vs REST Web Services:

There are many differences between SOAP and REST web services. The important 10 differences between SOAP and REST are given below:

No.	SOAP	REST
1)	SOAP is a **protocol**.	REST is an **architectural style**.
2)	SOAP stands for **Simple Object Access Protocol**.	REST stands for **REpresentational State Transfer**.
3)	SOAP **can't use REST** because it is a protocol.	REST **can use SOAP** web services because it is a concept and can use any protocol like HTTP, SOAP.
4)	SOAP **uses services interfaces to expose the business logic**.	REST **uses URI to expose business logic**.

5)	**JAX-WS** is the java API for SOAP web services.	**JAX-RS** is the java API for RESTful web services.
6)	SOAP **defines standards** to be strictly followed.	REST does not define too much standards like SOAP.
7)	SOAP **requires more bandwidth** and resource than REST.	REST **requires less bandwidth** and resource than SOAP.
8)	SOAP **defines its own security**.	RESTful web services **inherits security measures** from the underlying transport.
9)	SOAP **permits XML** data format only.	REST **permits different** data format such as Plain text, HTML, XML, JSON etc.
10)	SOAP is **less preferred** than REST.	REST **more preferred** than SOAP.

Java web services Basics:

Java web services provide concepts and examples of two main java web services api: JAX-WS and JAX-RS. The java web service application can be accessed by other programming languages such as .Net and PHP.

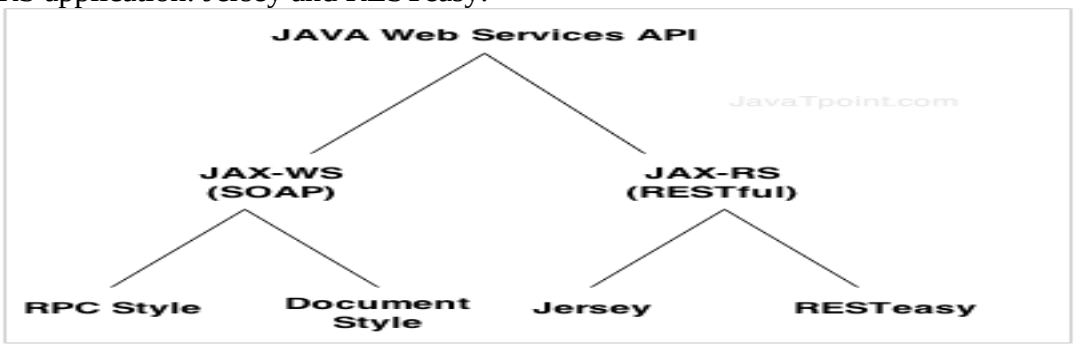
Java web service application performs communication through WSDL (Web Services Description Language). There are two ways to write java web service application code: SOAP and RESTful.

Java Web Services API

There are two API's defined by Java for developing web service applications since JavaEE 6.

1) **JAX-WS**: JAX-WS Stands for Java API for xml web services and it is based on the SOAP web services. It is based on xml based protocol that allows web services and client can communicate with different languages.There **are two ways to write JAX-WS** application code: by RPC style and Document style.

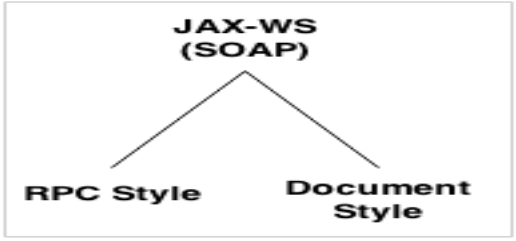
2) **JAX-RS**: JAX-RS stands Java API for xml REST web services. It uses Representational state transfer webservices. There are mainly **two implementation** currently in use for creating JAX-RS application: Jersey and RESTeasy.



JAX-WS:

JAX-WS is provides concepts and examples of JAX-WS API. This JAX-WS tutorial is designed for beginners and professionals.There are two ways to develop JAX-WS example.

1. RPC style
2. Document style



Difference between RPC and Document web services:
There are many differences between RPC and Document web services. The important differences between RPC and Document are given below:

RPC Style:

1) RPC style web services use method name and parameters to generate XML structure.
2) The generated WSDL is **difficult to be validated** against schema.
3) In RPC style, SOAP **message is sent as many elements**.
4) RPC style message is **tightly coupled**.
5) In RPC style, SOAP message **keeps the operation name**.
6) In RPC style, parameters are sent as **discrete values**.

The **RPC style generated WSDL** file.In WSDL file, it doesn't specify the types details.

```
<types/>
```

For message part, it defines name and type attributes.

```
<message name="getHelloWorldAsString">
<part name="arg0" type="xsd:string"/>
</message>
<message name="getHelloWorldAsStringResponse">
<part name="return" type="xsd:string"/>
</message>
```

Document Style:

1) Document style web services **can be validated against predefined schema**.
2) In document style, SOAP message is **sent as a single document**.
3) Document style message is **loosely coupled**.
4) In Document style, SOAP message **loses the operation name**.
5) In Document style, parameters are sent in **XML format**.

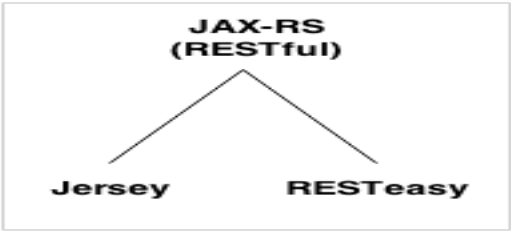
Let's see the **Document style generated WSDL file**.In WSDL file, it specifies types details having namespace and schemaLocation.

```
<types>
<xsd:schema>
<xsd:import namespace="http://javatpoint.com/" schemaLocation="http://localhost:7779/ws/hello?xsd=1"/>
</xsd:schema>
</types>
```

JAX-RS:

JAX-RS is provides concepts and examples of JAX-RS API. This JAX-RS tutorial is designed for beginners and professionals.There are two main implementation of JAX-RS API.

1. Jersey
2. RESTEasy



Web service Architecture:

Every framework needs some sort of architecture to make sure the entire framework works as desired. Similarly, in web services, there is an architecture which consists of three distinct roles as given below

1. Provider - The provider creates the web service and makes it available to client application who wants to use it.

2. Requestor - A requestor is nothing but the client application that needs to contact a web service. The client application can be a .Net, Java, or any other language based application which looks for some sort of functionality via a web service.
3. Broker - The broker is nothing but the application which provides access to the UDDI. The UDDI, as discussed in the earlier topic enables the client application to locate the web service.

Web services architecture: the service provider sends a WSDL file to UDDI. The service requester contacts UDDI to find out who is the provider for the data it needs, and then it contacts the service provider using the SOAP protocol. The service provider validates the service request and sends structured data in an XML file, using the SOAP protocol. This XML file would be validated again by the service requester using an XSD file.

Web Services Advantages:

We already understand why web services came about in the first place, which was to provide a platform which could allow different applications to talk to each other.

But let's look at some other advantages of why it is important to use web services.

1. Exposing Business Functionality on the network - A web service is a unit of managed code that provides some sort of functionality to client applications or end users. This functionality can be invoked over the HTTP protocol which means that it can also be invoked over the internet. Nowadays all applications are on the internet which makes the purpose of Web services more useful. That means the web service can be anywhere on the internet and provide the necessary functionality as required.

2. **Interoperability amongst applications** - Web services allow various applications to talk to each other and share data and services among themselves. All types of applications can talk to each other. So instead of writing specific code which can only be understood by specific applications, you can now write generic code that can be understood by all applications

3. **A Standardized Protocol which everybody understands** - Web services use standardized industry protocol for the communication. All the four layers (Service Transport, XML Messaging, Service Description, and Service Discovery layers) uses well-defined protocols in the web services protocol stack.

4. **Reduction in cost of communication** - Web services use SOAP over HTTP protocol, so you can use your existing low-cost internet for implementing web services.

Web service Characteristics

Web services have the following special behavioral characteristics:

1. They are XML-Based - Web Services uses XML to represent the data at the representation and data transportation layers. Using XML eliminates any networking, operating system, or platform sort of dependency since XML is the common language understood by all.

2. **Loosely Coupled** – Loosely coupled means that the client and the web service are not bound to each other, which means that even if the web service changes over time, it should not change the way the client calls the web service. Adopting a loosely coupled architecture tends to make software systems more manageable and allows simpler integration between different systems.

3. Synchronous or Asynchronous functionality- Synchronicity refers to the binding of the client to the execution of the service. In synchronous operations, the client will actually wait for the web service to complete an operation. An example of this is probably scenarios wherein a database read and write operation are being performed. If data is read from one database and subsequently written to another, then the operations have to be done in a sequential manner. Asynchronous operations allow a client to invoke a service and then execute other functions in parallel. This is one of the common and probably the most preferred techniques for ensuring that other services are not stopped when a particular operation is being carried out.

4. Ability to support Remote Procedure Calls (RPCs) - Web services enable clients to invoke procedures, functions, and methods on remote objects using an XML-based protocol. Remote procedures expose input and output parameters that a web service must support.

5. Supports Document Exchange - One of the key benefits of XML is its generic way of representing not only data but also complex documents. These documents can be as simple as representing a current address, or they can be as complex as representing an entire book.

Creating, Publishing, Testing and Describing a Web services (WSDL)

5.2.3 Creating, Publishing, Testing and Describing a Web Services (WSDL)

Q30. How to create, publish and text a web service? Explain with suitable examples using WSDL.

Answer :

Model Paper-III, Q10(b)

Web Services

Web services are software components which can be accessed remotely using method calls. They facilitate reusability and portability of softwares in applications. They use XML and HTTP for interaction between web services. Many Java APIs support the concept of web services. They can be published using some tools that implement them and invoke their methods on client applications. One such tool is "Netbeans" developed by "Sun".

The process of providing a web service whenever there are client requests is known as publishing a web service.

Example

Consider a huge integer web service that performs computations for positive numbers to an extent of 100 digits. It takes two string integers to input and computes the operations that is specified. These operations include sum, difference, greater than, less then and equality of numbers.

Creating a Web Service

In Netbeans, the initial step to create a web service is to create a "web application project". The steps to create a web application are as follows,

1. Initially select "New Project" from the "File" option. This opens "New project" dialog box.
2. Select "Web" from the dialog box.
3. Select "Web Application" from the list of "Projects" and click "Next".
4. Name the project and specify the location to store the project in "Project Name" and "Project Location" fields respectively. The location can be specified using the "Browse Button".
5. Then select Sun Java System Application Server 9 and Java EE5 from server and J2EE version drop-down list.
6. Finally, click on "Finish" button.

Now a web service class must be included to this web application project. The steps to include a web service class are as follows,

1. Initially go to "Projects" tab and right-click on the HugeIntegers node.
2. Then select web service from the New option. This opens "New Web Service" dialog box.
3. Specify the name in web service name and package fields with suitable names. For example, HugeIntegers as web services name and web service. Introduction part 1. Chapter 4 as package.

Finally, click on "Finish" button.

This creates a web service class name as HugeInteger. It resides in the "Projects" tab in the web services node.

Then the web service code for HugeInteger is developed and implemented in HugeInteger.java file. The code for implementing the web services is as follows,

```
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;
@WebService( name = "HugeInteger", serviceName = "HugeIntegerService")
public class HugeInteger
{
private final static int MAXIMUM = 100;
public int[ ] num = new int[MAXIMUM];
public String toString( )
{
String value = " ";
for (int digit : num)
value = digit + value;
int size = value.size( );
int location = -1;
for (int x = 0; x < size; x++)
{
if (value.charAt(i) ! = '0')
{
location = x;
break;
}}
return (location ! = -1 ? value.substring (location) : "0");
}
public static HugeInteger parseHugeInteger (String str)
{
HugeInteger temp = new HugeInteger( );
int stringsize = str.size( );
for (int x = 0; x< stringsize; xi++)
temp.num [x] = str.charAt(stringsize -x -1) - '0';
return temp;
}
@WebMethod (operationName = "add")
public String add (@WebParam(name = "string1") String string1, @WebParam(name = "string2") String string2)
{
int carry = 0;
HugeInteger operand1 = HugeInteger.parseHugeInteger(string1);
HugeInteger operand2 = HugeInteger.parseHugeInteger(string2);
HugeInteger result = new HugeInteger( );
for (int x = 0; x<MAXIMUM; x++)
{
result.num [x] = (operand1.num [x] + operand2.num [x] + carry) %10;
}
```



```
@WebMethod(operationName = "subtract")
public String subtract (@WebParam(name = "string1") String string1, @WebParam(name = "string2") String string2)
{
HugeInteger operand1 = HugeInteger.parseHugeInteger(string1);
HugeInteger operand2 = HugeInteger.parseHugeInteger(string2);
HugeInteger result = new HugeInteger( );
for (int x = 0; x<MAXIMUM; x++)
{
if (operand1.num [x] < operand2.num [x])
operand1.borrow (x);
result.num [x] = operand1.num [x] - operand2.num [x];
}
return result.toString ( );
}
private void borrow(int position)
{
if (position >= MAXIMUM)
throw new IndexOutOfBoundsException ( );
else if (num[position + 1] == 0)
borrow (position + 1);
num [position] += 10;
--num [position + 1];
}
@WebMethod(operationName = "bigger")
public boolean bigger (@WebParam(name = "string1") String string1,
@WebParam(name = "string2") String string2)
{
try
{
String difference = subtract(string1,string2);
return !difference.matches("^[0]+$");
}
catch (IndexOutOfBoundsException exp) {
return false;
}}
@WebMethod(operationName = "smaller")
public boolean smaller(@Webparam(name = "string1") String string1,
@WebParam(name = "string2") String string2)
{
return bigger(string2,string1);
}
@WebMethod(operationName = "equals")
public boolean equals (@Webparam(name ="string1")String string1,
@WebParam(name = "string2") String string2)
{
return !(bigger(string1,string2) || smaller(string1,string2));
}}}
```

Every new web service class is a plain old Java object and hence there is no need of extending a class or implementing an interface. A class that is compiled using JAX-WS 2.0 annotations will allow the compilers to generate the compile code framework. This framework enables the web services to wait for the client request and send a response to them. The various annotations that are used in creating and publishing the HugeInteger web services are as follows,

1. **@WebService**

This annotation is used at the beginning of every new web service class. It specifies the name and serviceName properties of the web service. It represents the implementation of web service.

Example

WebService(name="HugeInteger", ServiceName="HugeIntegerService")

The name indicates proxy class name for the client.

The serviceName indicates the class name that is used to retrieve the proxy class object.

2. **@WebMethod**

This annotation specifies the name of operation/method to be given to the web service client. It specifies @webparam annotation inside it. This annotation specifies the parameters for the method.

Example

@WebMethod(operationName = "add")

public string add(@webparam(name="first") string first, @webparam(name=second) string second)

Here, the operationName can be add, subtract, bigger, smaller and equal. They can be called remotely.

Publishing the Web Service

Publishing is the next step that is to be followed soon after creating the web service. Netbeans performs the publishing by managing the building and deploying process of the web service. It uses the "Projects" tab for this purpose. Whenever this tab is right-clicked, a pop-up menu will appear. This menu shows options necessary for building and developing the project. Some of those options are as follows,

1. **Build Project**

This option builds the project and returns the compilation errors.

2. **Deploy Project**

This option deploy the project on the server.

3. **Run Project**

This option performs the execution of the web application. It even builds or deploys the project if it was not build or deployed earlier.

Testing the Web Service

Testing is the next step that is to be followed after publishing the web service for a sun Java System Application. Server Testing can be performed by creating a web page. This server creates a web page dynamically for this purpose and also tests the methods.

The steps for testing the methods of web service are as follows,

1. Initially select the project name from the "Projects" tab.
2. Then select the "Properties" option. This will display the "Project Properties" dialog box.
3. Click on the "Run" option from the categories.
4. Type the URL in the "Relative URL" field.
5. Finally click on "OK" button.

After specifying the relative URL, the project must be run. Then the tester web page is build and loaded on the web browser from where the methods are to be tested.

The web service can be accesses when the application server is in running state. If the applications server is build by metbeans then it gets closed automatically when the developer closes the metbeans.

Describing the Web Service

Describing the web service is the next step after testing it. Then description includes methods, parameters and return values. JAX-WS uses WSDL for this purpose. The application server software (SJSAS) dynamically creates a WSDL for the webservice.

Q31. Describe the structure of a WSDL document, it elements and their purposes with appropriate exampl

OR

Explain the anatomy of WSDL.

Answer : Model Paper-I, Q11

Structure of WSDL Definition Document

Following are the elements that are contained within the structure of a WSDL definition document.

<definition>

The <definition> element is the root element of the WSDL document. It is used in the document for declaring the namespac and defining the name of the web service.

<type>

The <type> element is used for defining the type of message that can be transmitted between the web service provide and the web service requestor. The default message type is XML schema.

<message>

The <message> element is used for defining the data logically that can be transmitted between the service requestor an the service provider. Zero or more numbers of <part> elements can be included within it for describing the request parameters or response return values.

<portType>

The <portType> element is used for defining the operations of the web service abstractly. It does this by integrating request and response messages.

<binding>

The <binding> element is used for defining the operations of a web service in detail by using the concrete protocol and data format.

<port>

The <port> element is used for representing the address with which one can bound to the web service.

<service>

The <service> element has one or more <port> elements. Each <port> element indicates the binding information of a web service. The <service> element indicates the potentiality of the web service which can be called over multiple bindings.

Example

Consider the following WSDL document definition for the service providing the weather information.

```
<? xml version = "1.0"?>
<definition name = "Weather detail"
target Namespace = "http: //aboutweather.com/weather detail.wsdl"
xmlns: tns = "http://aboutweather.com/weatherdetail.wsdl"
xmlns: xsdl = "http://aboutweather.com/weatherdetail.xsd"
xmlns: soap = "http://schemas.xmlsoap.org/wsdl/soap/"
xmlns = "http: //schemas.xmlsoap.org/wsdl/">
<types>
    <schema targetNamespace =
"http: //aboutweather.com/weatherdetail.xsd"
xmlns = "http: //www.w3.org/2000/10/XMLschema">
<element name = "Request for WeatherDetail">
    <complex Type>
        <all>
        <element name = "City" type = "string"/>
        <element name = "zp" type = "string"/>
        <element name = "Instnt" type = "string"/>
        </all>
</complex Type>
```

```
</element>
<element name = "Weatherdetail">
<complex Type>
<all>
<element name = "Tmp" type = "float"/>
<element name = "Hmdty" type = "float"/>
</all>
</complex Type>
</element>
</schema>
</types>
<message name = "Obtain Weather DetailsI">
<part name = "Weather Detail RequestS"
element = "xsdl : Request for Weather Detail"/>
</message>
<message name = "Obtain Weather DetailsO">
<part name = "Weather detail"
element = "xsdl : Weather detail"/>
</message>
<port Type name = "Weather DetPort Type">
<operation name = "Obtain Weather Detail">
<input message = "tns : Obtain WeatherDetailI">
<output message = "tns : Obtain WeatherDetailO"/>
</operation>
</port Type>
<binding name = "Weather Detail Soap Binding"
type = "tns : Weather Det Port Type">
<soap : binding style = "document"
transport = "http : //schemas.xml soap. org/soap/http"/>
<operation name = "Obtain Weather Detail">
<soap : operation soap Action =
"http : //about weather.com/ObtainWeatherDetail"/>
<input>
<soap: body use = "literal"/>
</input>
<output>
<soap : body use = "literal">
</output>
</operation>
</binding>
<service name ="Weather Detail Service">
<documentation>
Provides Weather Details
</documentation>
<port name = "WeatherPort"
binding = "tns : WeatherDetail Soap Binding">
<soap : address location =
"http: //aboutweather.com/transmitsweather info"/>
</port>
</service>
</definitions>
```

Explanation

<definition>

- Represents name of the document i.e., Weatherdetail
- Represents instance specific namespace by using the attribute targetNamespace which represents the WSDL document as an XML schema namespace.
- Represents default namespace by specifying
  xmlns = "http : //schemas.xmlsoap.org/wsdl/" which indicates that the <type>, <message> and <port Type> elements are the part of it.

<message>

(i) Two message elements are defind,
  (a) ObtainWeatherDetail I (defines input message operation)
  (b) ObtainWeatherDetail O (defines output message operation).

(ii) The message element 'WeatherDetailsI' consists of an <input> element which includes the name as 'Weather Detail Requests' and the type as 'Request for Weatherdetail'

(iii) Similarly the other message element 'Obtain Weather DetailsO' consists for <output> element which includes the name as 'Weatherdetail' and the type as 'Weatherdetail'.

<portType> Element

1. Defines name of the operation as ObtainWeatherDetail
2. It includes two <input> messages ObtainWeather Detail I and ObtainWeatherDetailO defined by the <message> element
3. The different types of operations supported by WSDL are,

(a) One-way Operation

In this type of operation the service can only obtain the message. It is defined by onlyone input element.

(b) Request-response Operation

It this type of operation the service can obtain a request message defined by the <input> element and sends the response defined by the <output> element to the client.

(c) Solicit-response Operation

In this type of operation the service sends the request message defined by the <output> element and receives the response from the client defined by the <input> element.

(d) Notification Operation

In this type of operation the service sends the notification message defined by the <output> element to the client.

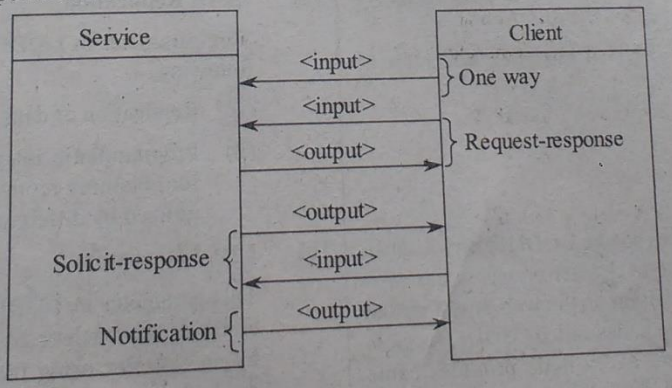


Figure: Operation Types of WSDL

<binding> Element

- Specifies message format
- Specifies details of protocols
- Includes <Type> attribute which refers
  <portType> element i.e., Weather Det Port Type.

<service> Element

- represents service location
- represents http : //aboutweatherdetail.com/transmitweather info/ as the service URL.