Malla Reddy College Engineering (Autonomous)

Maisammaguda, Dhulapally (Post Via. Hakimpet), Secunderabad, Telangana-500100 www.mrec.ac.in

Department of Information Technology

III B. TECH I SEM (A.Y.2018-19)

Lecture Notes

On

80602 - Web Technologies

UNIT-I

PHP

Introduction

PHP:—

The PHP Hypertext preprocessor (PHP) is a programming language that allows web developers to create dynamic content that interacts with databases. PHP is basically used for developing web based software applications.

→ PHP is a recursive acronym for "PHP: Hypertext ~~processor~~ Preprocessor"

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

→ the syntax is based on perl, java & c.

History:- PHP is a development from a project called Personal Home Page tools which was started by Rasmus Lerdorf in 1994.

→ PHP is an interpreted language, i.e, there is no need for compilation.

→ PHP is faster than other scripting language eg. asp & jsp

Common uses of PHP:-

* PHP performs system functions, i.e from files on a system it can create, open, read, write & close them.
* PHP can handle forms i.e, gather data from files, save data to a file, through email you can send data, return data to the user.
* you can add, delete, modify elements within your database through PHP.
* Access cookies variables & set cookies.
* Using PHP, you can restrict users to access some pages of your website.
* It can encrypt data.

Web Technologies - Uttam K Roy
The Complete Reference PHP - Steven Holzner

Scanned by CamScanner

~~Declaring Variables :-~~

~~In PHP, a variable starts with $sign, followed by the name of the variable:~~

PHP Example:- Save: hello.php

```
<!DOCTYPE>
  <html>
    <body>
      <?php
        echo "<h2> Hello by PHP </h2>";
      ?>
    </body>
  </html>
```

PHP code

output: Hello by PHP

after run

* Declaring Variables :-

In PHP, a variable starts with the $sign, followed by the name of the variable:

Syntax:
$variablename = value;

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

After the execution of the statements above, the variable $txt will hold the value Hello world!, the variable $x will hold the value 5, and the variable $y will hold the value 10.5.

The complete Reference PHP - Steven Holzner
Web Technologies - Uttam K Roy

Scanned by CamScanner

Start PHP:

To get access to a web server with PHP support, you can:

- Install Apache (or IIS) on your own server, install PHP, and MySQL
- Or find a web hosting plan with PHP and MySQL support.

Softwares needed:

- If your server supports PHP you don't need to do anything. Just create some .php files in your web directory, and the server will parse them for you. Because it is free, most web hosts offer PHP support.
- However, if your server does not support PHP, you must install PHP.
  - Download PHP free here: http://www.php.net/downloads.php
  - Download MySQL Database free here: http://www.mysql.com/downloads/
  - Download Apache Server free here: http://httpd.apache.org/download.cgi

Creating PHP script:

- The PHP code began with <?php and ended with ?>.
- This is similar to all HTML tags because they all begin with a less than (<) symbol and end with a greater than (>) symbol. These symbols (<?php and ?>) are called ***PHP tags***.
- They tell the web server where the PHP code starts and finishes.
- Any text between the tags is interpreted as PHP.
- Any text outside these tags is treated as normal HTML.

  EX:
  ```
  <?php
  ?>
  ```
- A PHP file must have a .php extension.
- A PHP file normally contains HTML tags, and some PHP scripting code.
- Below, we have an example of a simple PHP script that sends the text "Hello World" back to the browser:

  message.php:
  ```
  <html>
    <body>
    <?php
      echo "Hello World";
    ?>
    </body>
  </html>
  ```
- Each code line in PHP must end with a semicolon (;).
- The semicolon is a separator and is used to distinguish one set of instructions from another.
- There are two basic statements to output text with PHP: **echo** and **print**.
- In the example above we have used the echo statement to output the text "Hello World".
- The code that you now have in this file consists of four types of text:
  - HTML
  - PHP tags
  - PHP statements

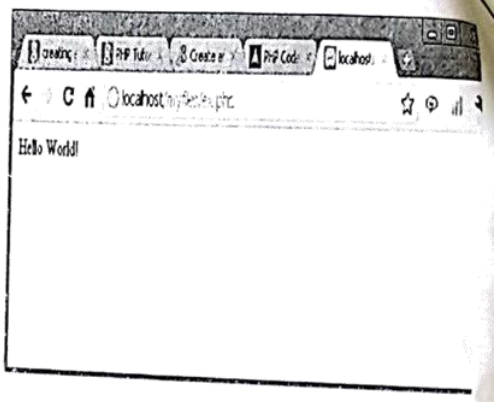
Scanned by CamScanner

- Whitespace

You can also add comments. Most of the lines in the example are just plain HTML.

Running PHP code:

- First start the xampp control panel, then start Apache server, and MySql server.
- In **C:/xampp**, you should save your PHP files in **htdocs** folder (which is in xampp directory, usually **C:\xampp\htdocs**).
- Easiest way to run them is open your web broswer (Internet explorer, Firefox etc.) type http://localhost/message.php (assuming you have started Apache server, and you stored message.php in htdocs folder.
- Then the message "Hello World" print on the browser.
- To run php files, just use browser, that's enough.



Comments in PHP:

In PHP, we use // to make a one-line comment or /* and */ to make a comment block:

```
<html>
  <body>
    <?php
      //This is a comment
      /*
        This is
        a comment block
      */
    ?>
  </body>
</html>
```

Scanned by CamScanner

PHP Variables :-

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

Rules for PHP Variables :-

A variable starts with the $ sign, followed by the name of the variable.

A variable name must start with a letter or the underscore character.

A variable name cannot start with a number

A variable name can only contain alpha-numeric characters & underscores (A-z, 0-9, and _).

Variable names are case-sensitive ($age and $AGE are two different variables).

PHP is a loosely Typed language

In the example above, notice that we did not have to tell PHP which data type the variable is. PHP automatically converts the variable to the correct data-type, depending on its value. In other languages such as C, C++, & Java, the programmer must declare the name & type of the variable before using it.

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

Scanned by CamScanner

Global and local scope:-

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

```
<?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:

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

Scanned by CamScanner

Data Types :-

PHP datatypes are used to hold different types of data or values. (or) Variables can store data of different types, and different datatypes can do different things.

PHP supports 8 primitive data types that can be categorized further in 3 types:

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

Scalar Types :

there are 4 Scalar data types in PHP.

1. boolean
2. Integer
3. float
4. String

Compound Types :

there are 2 Compound data types in PHP

1. array
2. object

Special Types :

there are 2 special data types in PHP

1. resource
2. Null

PHP String :

A String is a sequence of characters, like "Helloworld!".

A String can be any text inside quotes. you can use single or double quotes.

Scanned by CamScanner

```
<?php
 $x = "Helloworld!";
 $y = "Helloworld!";
 echo $x;
 echo "<br>";
 echo $y;
?>
```

PHP Integer :-

An Integer is a whole number (without decimals). It is a number between -2,147,483,648 and +2,147,483,647.

Rules for Integers :-

An integer must have atleast one digit (0-9).
An integer cannot contain comma or blanks
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 datatype & value:

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

Scanned by CamScanner

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 datatype and value:

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

PHP Boolean:-

A Boolean represents two possible states: TRUE or FALSE

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

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 datatype and value:

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

PHP Object:-

An object is a datatype which stores data & 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 & methods:

```
<?php
class car {
  function car()
```

Scanned by CamScanner

```
{
$this -> model ="vw";
}
}
//Create an object
$herbie =new Car();
// show object properties
echo $herbie ->model;
?>
```

PHP NULL Value:-

Null is a special datatype which can have only one value : NULL
A variable of datatype NULL is a variable that has no value assigned to it.

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

PHP Resource:-

the special resource type is not an actual datatype. It is the storing of a reference to functions and resources external to PHP.

* Array:-

Scanned by CamScanner

Array:-

It is an order map

Advantages: Less code → we don't need to define multiple variables
Easy to traverse, sorting → By the help of while loop we can traverse all the elements of an array
we can sort all the elements of array

An Array is a Special variable, which can hold more than one value at a time.

if you have a list of items (a list of car names, for example), Storing the Cars in Single variables could look like this:

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

However, what if you want to loop through the Cars & find a Specific one? And what if you had not 3Cars, but 300?

the solution is to Create an array!

An array can hold many values under a Single name, and you Can access the values by referring to an index number.

Create an Array in PHP

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

In PHP, there are three types of arrays:

1. Indexed arrays-Arrays with a numeric index
2. Associative Arrays - Arrays with named Keys
3. Multi dimensional Arrays - Arrays Containing one or more arrays

PHP Indexed Arrays:-

there are two ways to Create indexed arrays:

the index can be assigned automatically (index always starts at 0), like this: or we can store number in the array.

```
$Cars = array ("Volvo", "BMW", "Toyota"); or
$cars[0] = "volvo";
$cars[1] = "winter";
$cars[2] = "Toyota";
```

Scanned by CamScanner

the following example creates an indexed array named $cars, assigns three elements to it, and then prints a text containing the array values:

```
<?php
$cars = array ("volvo", "BMW", "Toyota");
echo "I like" . $cars[0] . "," . $cars[1] . "and" . $cars[2] . ".";
?>
```

"I like: $cars[0], $cars[1], $cars[2]";

Php Associative Arrays:-

We can associate name with each array element

Associative arrays are arrays that use named keys that you assign to them. using ⇒ symbol

there are two ways to create an associative array:

$age = array ("peter" ⇒ "35", "Ben" ⇒ "37", "Joe" ⇒ "43");

$age ["peter"] = "35";
$age ["Ben"] = "37";
$age ["Joe"] = "43";

```
<?php
$age = array ("peter" ⇒ "35", "Ben" ⇒ "37", "Joe" ⇒ "43");
echo "peter is" . $age['peter'] . "years old.";
?>
```

php Multidimensional Arrays:-

It is also known as array of arrays. In php, It is represented in the form of matrix which is represented by row * column.

Ex:-

```
<?php
$emp = array (array (1, "sonoo", 4000),
              array (2, "john", 5000),
              array (3, "rahul", 3000));
for ($row = 0; $row < 3; $row ++)
{
for ($col = 0; $col < 3; $col++) {
 echo $emp [$row] [$col] . " "; }
 echo "<br />";
}
?>
```

1	sonoo	4000
2	john	5000
3	rahul	3000

Scanned by CamScanner

Strings :-

A String is a sequence of characters, i.e, used to store & manipulate text like " Hello World!".

we can specify string by using ' ' or " "

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 "Helloworld!":

```
<?php
 echo strlen("Helloworld!"); // output:12
?>
```

Count the Number of words in a String :-

The PHP str_word_Count() function counts the noof words in a string:

```
<?php
echo str_word_Count ("Hello world!"); // output:2
?>
```

Reverse a String :-

the PHP strrev() function reverses a String:

```
<?php
echo strrev("Helloworld!"); //output : dlrowolleH
?>
```

String Concatenation Operator :-

To Concatenate two String variables together, use the dot(.) operator-

```
<?php
 $string1 = "Hello world";
 $string2 = "1234";
 echo $string1 . " " . $string2; //output:- Helloworld 1234
?>
```

Using the strpos() function :-

the strpos() function is used to search for a string or character within a String.

If a match is found in the String, this function will return the position of the first match. if no match is found, it will return FALSE.

Scanned by CamScanner

Let's see if we can find the String "world" in our String -

```
<?php
 echostrpos ("Helloworld!","world"); ||output: 6
?>
```

Note:- the first character position in a String is 0 (not 1)

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!"); ||output: Hello Dolly!
?>
```

* Operators:- An operator is a symbol that specifies a particular action in an expression. (or) Operators are classified into different types, An operator is a symbol i.e, used to perform operations on operands.

(a) Arithmetic Operators:-

the arithmetic operators are,

operators	label	Example
+	addition	$a+$b
-	subtraction	$a-$b
*	Multiplication	$a*$b
/	Division	$a/$b
%	Modulus	$a%$b

(b) Assignment Operators:-

Assignment operators mainly used to to assign a data value to a variable. the simplest form of assignment operator just assigns some value.

operator	Label	Example
=	Assignment	$a=5
+=	Addition-assignment	$a+=5
*=	Multiplication-assignment	$a*=5
/=	division - assignment	$a/=5
.=	Concatenation - assignment	$a.=5

Scanned by CamScanner

String Operators:- php's string operators provide two operators that two operators usefull for Concatenation the two Strings.

operator	label	example	output
.	Concatenation	$txt1. $txt2	concatenation of $txt1 and $txt2
.=	Concatenation-assignment	$txt1.=$txt2	Appends $txt2 to $txt1

(d) Increment and decrement Operators:-

Increment (++) and decrement(--) operators increment by 1 and decrement by 1 from the current value of a variable.

operator	Name	Description	result
++	increment	++$a, $a++	increment $a by 1
--	Decrement	--$a, $a--	decrement $a by 1

(e) logical operators:-

the PHP logical operators are used to combine Conditional Statements. logical operators make it possible to direct the flow of program and used frequently with Control structures such as the if conditional and while loops.

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

Scanned by CamScanner

(f) Equality Operators:-

equality operators are used to compare two values, testing for equivalence.

operator	label	Example
<	lessthan	$a< $b
>	Greater than	$a > $b
<=	Lessthan or equalto	$a<= $b
>=	greater than or equalto	$a >= $b

(g) bitwise operators:-

Bitwise operators are used for variations on some of the logical operators.

operator	Name	Example	Result
&	AND	$a& $b	and together each bit contained in $a and $b
|	OR	$a| $b	or together each bit contained in $a and $b
^	XOR	$a^ $b	exclusive or together each bit contained in $a and $b
~	NOT	~$b	negate each bit in $b
<<	shift left	$a<< $b	$a will receive the value of $b shifted left two values
>>	shift right	$a >>$b	$a will receive the value of $b shifted right two values.

Scanned by CamScanner

Expressions:-

An expression is a phrase representing a particular action in a program. all expressions consists of atleast one operand and one (or) more operations.

Ex:-

```
$a=5 ; // assign integer value 5 to the variable $a
$a="5"; // assign string value "5" to the variable $a
$a="abit"; // assign "abit" to the variable $a
```

→ Here, operands are the input expressions,

Ex:- `$a++; // $a is the operand`

`$sum= $val1 +$val2 ; // $sum, $val1, $val2 are operands`

Control Structures | statements:-

PHP Supports different types of statements like,

(1) if Statement:-

PHP if Statement is executed if Condition is true.

Syntax:-

```
if(condition) {
// Code to be executed
}
```

Example:-

```
<?php
$num=12 ;
if($num<100) {
echo "$num is less than100";
}
?>
```

output:-

12 is less than 100

Scanned by CamScanner

if-else Statement:-

PHP if-else statement is executed whether condition is true, false.

Syntax:-

```
if(condition)
{ ||code to be executed if true
} else {
||Code to be executed if false
}
```

Example:-

```
<?php
 $num=12;
 if($num %.2 ==0)
 {
 echo " $num is even number";
 }
 else
 {
 echo " $num is odd number";
 }
 ?>
```

output:-

12 is even number

Switch Statement:-

PHP Switch Statement is used to execute one Statement from multiple conditions. It works like PHP if-else-if Statement.

Syntax:-

```
Switch(expression) {
Case value1;
```

Scanned by CamScanner

```
// Code to be executed
break;
Case value2:
// Code to be executed
break;
------
default:
Code to be executed if all cases are not matched;
}
```

Example:-

```
<?php
 $num=20;
 Switch ($num) {
 case 10;
 echo( "number is equal to 10");
 break;
 Case 20:
 echo("number is equal to 20");
 break;
 Case 30:
 echo("number is equal to 30");
 break;
 default:
 echo("number is not equal to 10,20 or 30");
 }
 ?>
```

output:-

number is equal to 20

Scanned by CamScanner

Php forloop Statement:-

Php for loop can be used to traverse set of code for the specified number of times.

Syntax:-

```
for(initialization; Condition; increment/decrement)
{ //code to be executed
}
```

Example:-

```
<?php
for($n=1; $n<=10; $n++)
{ echo "$n<br/>";
}
?>
```

output:-

1
2
3
4
5
6
7
8
9
10

Nested forloop:-

We can use for loop inside for loop in PHP, is known as nested forloop

Example:-

```
<?php
for($i=1; $i<=3; $i++)
{
 for($j=1; $j<=3; i++)
 {
  Echo "$i $j <br/>";
 }
}
?>
```

output:-

1 1
1 2
1 3
2 1
2 2
2 3
2 1
3 2
3 3
3

Scanned by CamScanner

Syntax:-

```
while(Condition)
{
  //code to be executed
}
```

Example:-

```
<?php
$n=1;
while($n<=10)
{
echo "$n<br/>";
$n++;
}
?>
```

output:-

1
2
3
4
5
6
7
8
9
10

PHP dowhile loop:-

PHP do while loop can be used to traverse set of code like php while loop. the PHP do-while loop is guaranteed to run atleast once.

It executed the code at least onetime always because condition is checked after executing the code.

Syntax:-

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

Scanned by CamScanner

for Each loop Statement:-

PHP for Eachloop is used to traverse array elements.

Syntax:-

```
foreach ($array as $var)
{
  //code to be executed
}
```

Example:-

```
<?php
$season = array ("summer", "winter", "Spring", "autumn");
foreach($season as $var)
{
 echo "Season is: $arr<br/>";
}
?>
```

output:-

Season is: Summer
Season is: winter
Season is: Spring
Season is: autumn

PHP while loop:-

PHP while loop is used to traverse set of code like for loop. It should be used if no.of iteration is not known.

Scanned by CamScanner

Example:-

```
<?php
 $n=1;
 do
 {
 echo "$n<br/>";
 $n++;
 } while($n<=10);
 ?>
```

output:-

1
2
3
4
5
6
7
8
9
10

break and goto Statement:-

PHP break Statement break the execution of current for, while, do-while, Switch & for-each loop. if you use break inside inner loop, it breaks the execution of inner loop only.

Syntax:-

```
jump Statement;
 break;
```

Example:-

```
<?php
 for($i=1; $i<=10 ; $i++)
  {
  echo "$i<br/>";
   if($i==5) {
    break;
   }
  }
 ?>
```

output:-

1
2
3
4
5

Scanned by CamScanner

Goto:- this means we can suddenly jump to a specific location outside of a looping & conditional construct.

Continue:-

Continue statement execute the current loop iteration to the end.

Scanned by CamScanner

Function:

- Function is a set of statements for performing a task.
- A function will be executed by a call to the function.
- You may call a function from anywhere within a page.

Create a PHP Function:

- A function will be executed by a call to the function.
- In PHP the function will be create with a keyword "function".

Syntax:

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

PHP function guidelines:

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

Example:

A simple function that writes my name when it is called:

```
<html>
  <body>
    <?php
        function add()
        {
         $a=10;
         $b=20;
         echo "Addition=".($a+$b);
        }
        add();
    ?>
  </body>
</html>
```

Output:

Addition=30

PHP Functions - Adding parameters:

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

Example:

Scanned by CamScanner

```
<html>
  <body>
    <?php
        function add($a,$b)
        {
          echo "Addition=".($a+$b);
        }
        add(10,20);
    ?>
  </body>
</html>
```

PHP Functions - Return values:

To let a function return a value, use the return statement.

Example:

```
<html>
  <body>
    <?php
      function add($x,$y)
      {
       $total=$x+$y;
      return $total;
      }
      echo "1 + 16 = " . add(1,16);
    ?>
  </body>
</html>
```

Output:

1 + 16 = 17

PHP Form Handling:

- The most important thing to notice when dealing with HTML forms and PHP is that any form element in an HTML page will **automatically** be available to your PHP scripts.
- The PHP $_GET and $_POST variables are used to retrieve information from forms, like user input.

Example:

The example below contains an HTML form with two input fields and a submit button:

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

Scanned by CamScanner

```
  <form action="welcome.php" method="post">
    Name: <input type="text" name="fname" />
    Age: <input type="text" name="age" />
    <input type="submit" />
  </form>
 </body>
</html>
```

When a user fills out the form above and clicks on the submit button, the form data is sent to a PHP file, called **"welcome.php"**:

"welcome.php" looks like this:

```
<html>
  <body>
    Welcome <?php echo $_POST["fname"]; ?>!<br />
    You are <?php echo $_POST["age"]; ?> years old.
  </body>
</html>
```

Output:

Welcome John!
You are 28 years old.

Scanned by CamScanner

functions :-

Arrays :- Array is a collection of heterogeneous (different elements) datatypes in php. Because php is a loosely typed language.

Ex :- 1

```
<?php
$arr= array(10,20,30);
print_r($arr);
?>
```

output :- [0]=10, [1]=20, [2]=30

Ex :- 2

```
<?php
$arr = array (100 -> 10 , 101 -> 20, 102 -> 30);
print_r($arr);
?>
```

output :- [100]=10, [101]=20, [102]=30

Ex :- 3

```
<?php
$arr=array(100=>10,20,30, 106=>36);
print_r($arr);
?>
```

output :-

[100]=10, [101]=20, [102]=30, [106]=30

Ex :- 4

```
<?php
$arr =array(100=>10, 'city'=>'hyd',
            105=>30, 50=>40,70);
print_r($arr);
?>
```

output :-

[100]=10, [city] = hyd, [105]=30

[50]=[40], [51] = [70]

Array functions :-

Count :- it retrns total no. of elements

Ex :-

```
<?php
$arr =array (10,20,30);
Echo count($arr);
?>
```

output :- 3

Scanned by CamScanner

Sort:- it returns the elements of an array in assending order.

Ex:-

```
<?php
$arr = array (60,20,30);
sort($arr);
print_r($arr);
?>
```

output:-

20,30,60

rsort:- it returns the elements of an array in descending order.

Ex:-

```
<?php
$arr = array (101,104, 102);
rsort_r($arr);
print_r ($arr);
?>
```

output:-

104, 102, 101

asort:- it returns the original key values with descending order.

Ex:-

```
<?php
$arr = array (104=>40, 101=>20, 108=>50, 102=>80);
asort ($arr);
print_r($arr);
?>
```

output:-

[102]=80, [108]=50, [104]=40, [101]=20

Scanned by CamScanner

Ksort:- it retrns the array in assending order with based on the "Keys".

Ex:-

```
<?php
 $arr= array (104 => 40, 101 => 20, 108 => 50, 102 => 80);
ksort ($arr);
 print_r ($arr);
?>
```

output:-

[101] = 20 , [102] = 80, [104] = 40 , [108] = 50

Krsort:- it retrns the array in dessending order with based on the "Keys".

Ex:-

```
<?php
 $arr = array (104 => 40, 101 => 20 , 108 => 50, 102 => 80);
krsort( $arr);
print_r ($arr);
?>
```

output:-

[108] = 50 , [104] = 40, [102] = 80 , [101] = 20

array_push():-

this function adds an elements into the end of an array and retrns the total no.of elements in that array.

Ex:-

```
<?php
$arr =array (10, 20, 30);
Echo array_push ($arr, 40);
 print_r($arr);
?>
```

output:-

Scanned by CamScanner

array_pop():- remove the last element & retsin the value of that element.

Ex:-

```
<?php
$arr = array (10, 20, 30);
Echo array_pop($arr);
print_r($arr);
?>
```

output:-

array_shift():- it removes the first element of an array and retsins the value of that element.

Ex:-

```
<?php
$arr = array (10,20,30);
Echo array_shift($arr);
print_r($arr);
?>
```

output:-

array_unshift():- add an element at the begining of an array and return size of an array.

Ex:-

```
<?php
$arr = array (10, 20, 30);
Echo array_unshift ($arr);
print_r ($arr);
?>
```

output:-

Scanned by CamScanner

array_change_key_case():-

it Converts all Keys of an array into lower case

Ex:-

```
<?php
$arr = array ('ABC' => 10, 20, 30);
print_r (array_change_key_case ($arr));
?>
```

array_chunk():- Splits an array into chunk of an array

Ex:-

```
<?php
$arr = array (10, 20, 30, 40, 50, 60);
print_r (array_chunk ($arr, 2));
?>
```

array_combine():-

Creats an array by using one array for Keys and another for its value.

Ex:-

```
<?php
$arr = array ('abc' => 10, 20, 30, 40, 50);
$arr1 = array (100, 200, 300, 400, 500);
print_r (array_combine($arr, $arr1));
?>
```

Scanned by CamScanner

array_keys():- it retrns new array with keys as value of another array.

Ex:-

```
<?php
$arr = array ('abc' => 10, 20, 30, 40);
print_r (array_keys ($arr));
?>
```

array_count_values();-

retrns an array with no.of occurance for each value

Ex:-

```
<?php
$arr = array ('ABC' => 10, 20, 30, 40, 50, 10);
print_r (array_count_values ($arr));
?>
```

array_values():- retun array with the values of an array

Ex:-

```
<?php
$arr = array ('ABC' = 10, 20, 30, 40, 50, 60);
print_r (array_values ($arr));
?>
```

Scanned by CamScanner

array_flip():- exchanges all Keys with their associated values in array.

Ex:-

```
<?php
$arr = array ('ABC' -> 10, 20, 30, 40);
    // $arr = array (10, 200, 400);
 printf_r (array_flip ($arr));
?>
```

array_interest():- Compares array values & returns the matches.

```
<?php
 $arr = array (10, 20, 30, 40);
 $arr = array (100, 200, 300, 400, 10);
 printf_r (array_interest ($arr));
?>
```

array_merge():- merges one or more arrays into one array

```
<?php
$arr = array ('ABC' => 10, 20, 30, 40);
 $arr = array (100, 200, 300);
 print_r (array_merge ($arr, $arr1));

?>
```

array_sum():- returns the sum of all elements of an array

```
<?php
 $arr = array (10, 20, 30, 40);
 Echo print_r (array_sum ($arr));
?>
```

Scanned by CamScanner

array_reverse():- It reverse the elements of array

```
print_r(array_reverse($arr));
```

array_unique():- removes the duplicate values & returns the values of an array.

```
<?php
$arr= ("ABC" => 10,20,30,40);
print_r(array_unique($arr));
?>
```

shuffle():- shuffle the elements of array

```
<?php
$arr =array ('ABC'=> 10,20,30,40);
shuffle ($arr);
print_r ($arr);
?>
```

Extract():- divides the elements of an array as individual variables.

```
<?php
$arr = array ('ABC'=> 10,20,30,40);
Extract($arr);
Echo $ABC;
?>
```

list():-

```
<?php
List ($x, $y, $z) = array (10,20,30);
Echo $x;
Echo $y;
Echo $z;
```

Scanned by CamScanner

String functions:-

① PHP Strtolower() function:-

This function returns string in lowercase letter.

Syntax:-

string strtolower(String $string)

Ex:-

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

o/p:-

my name is sai

② strtoupper()

this function returns string in upper case letter.

Syn:-

string strtoupper(string $string)

Ex:-

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

o/p:-

MY NAME IS SAI

③ ucwords()

returns string converting first character of each word into uppercase.

Syn:-

string ucwords(String $str)

Ex:-

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

o/p:-

My Name Is Sai

④ ucfirst(): capitalizing the first letter of string.

Ex: My name is sai

Scanned by CamScanner

Math functions :-

abs() function :-

It returns absolute value of given number. It returns an integer value but if you pass floating point value it returns a float value.

Syntax :-

```
number abs (mixed $number)
```

Ex :-

```
<?php
echo abs(-7); //7
echo abs(-7.2); //-7.2
?>
```

ceil() function

Syn :- It returns rounds fractions up.

```
float ceil (float $value)
```

Ex :-

```
<?php
echo ceil(3.3); //4
echo ceil (-4.557); //-4
?>
```

floor()

It returns rounds x fractions down.

Syn :-

```
float floor (float $value)
```

Ex :-

```
<?php
echo floor(3.3); // 3
echo floor (-4.55); //-5
?>
```

sqrt()

It returns square root of given argument.

Syn :-

```
float sqrt (float $arg)
```

Ex :-

```
<?php
echo (sqrt (16)); //4
echo (sqrt(25)); //5
?>
```

decbin()

It converts decimal number into binary. It returns binary number as a string.

Syn :-

```
string decbin (int $number)
```

```
<?php
echo( decbin(2)); //10
echo (decbin (10)); //1010
?>
```

Scanned by CamScanner

Comparing two String:-

Strcmp():- Compare two Strings (Case-Sensitive)

Syn:- int strcmp (String str1, String str2)

```
<?php
 $pwd1="abced";
 $pwd2="abced2";
?>
```

Strspn(): calculating the Similarity b/w two Strings

Syn:- int strspn (String str1, String str2 [, int start [, int length]])

```
<?php
 $pwd="abc123";
 if(strspn ($pwd, "1234567890") == strlen ($pwd))
 echo "the pwd Cannot Consist of numbers";
?>
```

Scanned by CamScanner

Handling file uploads:-

php file upload:- php allows you to upload single & multiple files through few lines of code only.

php $FILES

the php global $_FILES contains all the information of file. By the help of $_FILES global, we can get filename, filetype, file size, temp file name & errors associated with file.

Here, we are assuming that filename is filename.

$_FILES['filename']['name']

returns filename

$_FILES['filename']['type']

returns MIME type of the file

$_FILES['filename']['size']

returns size of the file (in bytes)

$_FILES['filename']['tmp_name']

returns temporary filename of the file which was stored on the server.

$_FILES['filename']['error']

returns the error code associated with this file

move_uploaded_file() function:-

the move_uploaded_file() function moves the uploaded file to a new location. the move_uploaded_file() function checks internally if the file is uploaded through the

Scanned by CamScanner

post request. It moves the file if it is uploaded through the post request.

Syntax:-

bool move_uploaded_file(String $filename, String $destination)

Example:-

file: uploadform.html

```
<form action="uploader.php" method="post" enctype="multipart/form-data">
  Select file:
      <input type="file" name="fileToupload"/>
      <input type="submit" value="UploadImage" name="submit"/>
</form>
```

file: uploader.php

```
<?php
  $target_path="e:/";
  $target_path = $target_path.basename($_FILES['fileupload']['name']);
  if(move_uploaded_file($_FILES['fileToupload']['tmp_name'], $target_path))
  {
    echo "file uploaded successfully!";
  }else
  {
    echo "sorry, file not uploaded, please try again!";
  }
?>
```

Scanned by CamScanner

Connecting to database (MySQL)

PHP mysqli_connect() is used to connect with MySQL database. It returns if connection is established or null.

Syntax:-

resource mysqli_connect(Server, username, password)

PHP mysqli_close() is used to disconnect with MySQL database. It returns connection is closed.

Syntax:-

bool mysqli_close (resource $resource_link)

Ex:-

```
<?php
$host = 'localhost:3306';
$user = '';
$pass = '';
$con = mysqli_connect ($host, $user, $pass);
if (!$conn)
{ die('could not connect:' .mysqli_error());
}
echo 'connected successfully';
mysqli_close ($conn);
?>
```

output:-

Connected successfully

Create Database:-

mysqli_query is used to create db.

```
$sql = 'CREATE Database mydb';
if (mysqli_query($conn, $sql))
{ echo "Database mydb Created successfully";
}else
{ echo " sorry, database Creation failed" .mysqli_error($conn);
}
```

Scanned by CamScanner

Create Table

```
$sql = "Create table emp5 (id INT AUTO_INCREMENT, name varchar (20) NOT NULL,
emp_salary INT NOT NULL , Primary Key (id))";
if (mysqli_query ($conn, $sql)) {
    echo " Table emp5 created succcutully";
  } else {
    echo "could not Create Table : ".mysqli_error ($conn);
  }
```

insert

```
$sql = "INSERT into emp5 (name, salary) values ("soroo", 901)";
```

update

```
$sql = "update emp5 set name = \"$name\", Salary = $salary where id= $id";
```

delete

```
$id=2;
$sql = "delete from emp5 where id= $id";
```

Scanned by CamScanner

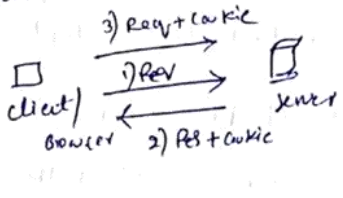
Using php Cookies :-

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 & saved to client. Each time when client sends request to the server, cookie is embedded with request. Such way, cookie can be recieved at serverside.

SetCookie() function

Note:- PHP cookie must be used before html tag




```
setCookie ("cookiename", "cookievalue");
        /* defining cookie name & value */
```

PHP $_COOKIE

It is a super global variable is used to get cookie.

```
$value = $_COOKIE ["CookieName"];  // retry cookie value
```

Ex:-

```
<?php
 setCookie ("user", "Sonoo");
 ?>
<html>
  <body>
    <?php
    if (!isset ($COOKIE["user"])) {
          echo "sorry, cookie not found!";
      } else {
        echo "<br/> cookie value :" . $_COOKIE ["user"];
      }
    ?>
  </body>
</html>
```

O/P

Sorry, cookie not found!

Firstly, cookie is not set. But, if you refresh page you will see cookie is set now

cookie value = sono o

Delete cookie

If you set the expiration date in past, cookie will be deleted.

cookie.php

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

Scanned by CamScanner

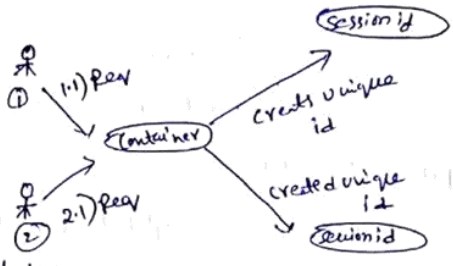
PHP Session:-

It is used to store & pass information from one page to another temporarily (until user close the website)

this technique widely used in shopping websites where we need to store, pass cart information eg: Username, product code, Pdtname, Productprice etc from one page to another.

PHP session creates unique userid for each browser to recognize the user & avoid conflict b/w multiple browser.

Session_start() function



His used to start the session.

```
session_start();
```

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

Store info

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

GET info

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

Ex:-

session1.php

```
<?php
Session_start();
?>
<html>
<body>
<?php
$_SESSION["user"]="sachin";
echo "session information are set successfully.<br/>";
?>
<a href="session2.php">visit nextpage</a>
</body>
</html>
```

session2.php

```
<?php
session_start();
?>
<html>
<body>
<?php
echo "User is:".$_Session["user"];
?>
</body>
</html>
```

Php Destroy session

php session_destroy() function is used to destroy all session variables completly.

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

Scanned by CamScanner

File Handling in PHP:-

PHP file System allows us to create file, read file line by line, read file character by character, write file, append file, delete file & close file.

PHP Open file - fopen():-

The PHP fopen() function is used to open a file

Syntax:-

resource fopen(String $filename, string $mode [, bool $use_include_path = false [, resource $context]])

(represents file to be opened; file mode for r, r+, w, w+ read only, read write, write only, read write)

Example:-

```
<?php
 $handle = fopen ("c:\\folder\\file.txt", "r");
?>
```

r = read-only
r+ = read-write
w = write-only
w+ = read-write

PHP Close file - fclose():-

The PHP fclose() function is used to close an open file pointer.

Syntax:-

bool fclose(resource $handle)

Example:-

```
<?php
 fclose($handle);
?>
```

Scanned by CamScanner

echo "

PHP Read file :-

PHP provides various functions to read data from file. there are different functions that allow you to read all file data, read data line by line and read data character by character.

the available PHP file read functions are given below,

1. fread()
2. fgets()
3. fgetc()

PHP Read file - fread() :-

If Php read() function is used to read data of the file. It requires two arguments: file resource & filesize

Syntax:-

String fread (resource $handle, int $length)

$handle represents file pointer that is created by fopen() function.

$length represents length of byte to be read.

Example:-

```
<?php
$filename ="c:\\ file1.txt";
$fp = fopen ( $filename ,"r"); // openfile in readmode
$contents = fread ( $fp, filesize($filename)); //read file
```

Scanned by CamScanner

```
echo "<pre>$contents</pre>"; //printing data of file
fclose($fp); //closefile
?>
```

output:-
this is first line
this is another line
this is third line

Php read file - fgets():-

the php fgets() function is used to read single line from the file.

Syntax:-

String fgets(resource $handle [, int $length])

Example:-

```
<?php
$fp = fopen("c:\file.txt"); //open file in read mode
echo fgets($fp);
fclose($fp);
?>
```

output:-
this is firstline

Php readfile - fgetc():-

It is used to read single character from the file. To get all data using fgetc() function, use !feof() function inside the while loop.

Syntax:-

String fgetc (resource $handle)

Scanned by CamScanner

Example:-

```
<?php
$fp= fopen ("c:\\file.txt", "r"); ||open file in readmode
while(!feof($fp)) {
    echo fgetc($fp);
  }
  fclose($fp);
  ?>
```

output:-

this is firstline-thisisanotherline-thisisthirdline

PHP write file:-

PHP fwrite() and fputs() functions are used to write data into file. You need to use w, r+, w+, x, x+, c & c+ mode

PHP write file- fwrite():-

the PHP fwrite() function is used to write content of the String into file.

Syntax:-

int fwrite(resource $handle, string $string [, int $length])

Example:-

```
<?php
$fp = fopen ('data.txt' , 'w'); || opens file in write-only mode
fwrite ($fp , 'welcome');
fwrite ($fp , 'to php file write');
fclose ($fp);
echo "file writen Successfully";
?>
```

output:- data.txt

welcome to phpfile write

Scanned by CamScanner

PHp overwriting file:-

If you run the Above Code agin, it will erase the Previous data of file & writes the new data. let's See the Code that writes only new data into data.txt file.

Syntax:- Example:-

```
<?php
$fp = fopen ('data.txt', 'w'); //opens file in write-only mode
fwrite ($fp, 'hello');
fclose ($fp);
echo "file written Successfully";
?>
```

output

data.txt

hello

php append to file:-

you Can ~~open~~ append data into file by using w or w+ mode in fopen() function. Let's See the example, that appends data into data.txt file.

let's See the data of file first.

data.txt

welcome to php file write

Example:-

```
<?php
$fp = fopen ('data.txt', 'a'); //opens file in append mode
fwrite ($fp, 'this is additional text');
```

Scanned by CamScanner

```
fwrite($fp, 'appending data');
fclose($fp);
echo "file appended successfully";
?>
```

output:-

data.txt

welcome to php file write this is additional text appending data

Php Delete file:-

In pHp, we can delete any file using unlink() function the unlink() accepts on argument only: filename. ~~It is similar to~~

PHP UNLINK() generates E_WARNING level error if file is not deleted. It returns TRUE if file is deleted successfully otherwise FALSE

Syntax:- bool unlink (string $filename [, resource $context])

$filename represents name of the file to be deleted.

Example:-

```
<?php
$status = unlink ('data.txt');
if($status) {
echo "file deleted successfully";
} else {
echo "sorry!";
}
?>
```

output:-

file deleted successfully

Scanned by CamScanner

UNIT-II
XML

Introduction to XML:-

XMS Stands for extensible Markup language. this is Similar to HTML, but the tags are user defined. HTml & xml are used to build webpages. All the Scripting languages are Concerned with representation of data on the web page. But, looking up the data, transfering (or) Storage is not possible. XML is designed to transport (or) to Store the data.

XML Standard was developed by WWWC (World wid web Consortium) in the late 1990's. It is a International Standard Organization. & This application (xml) is derived from SGL (Standarad Generalized Markup language) → how to specify a document markup language such Specification itself a DTD (Document Type Definition).

1. XML Stands for extensible Markup language
2. XML was designed to describe data
3. xml tags are not predefined in XML. you must define your own tags.
4. XML is self describing
5. XML uses a DTD to formally describe the data.

Difference between HTML & XML:-

XML is not a replacement for HTml.

XML & HTml were designed both different goals.

1. XML was designed to describe the data & focus on what data is.
   HTML was designed to display data & to focus on how data looks.
2. HTML is about displaying information, xml is about describing information.

Scanned by CamScanner

Uses of XML :-

→ XML is used to display the meta contents i.e, xml describes the contents of the document.

→ XML is useful in exchanging data between the applications.

→ The data can be ~~extracted~~ extracted from database and can be used in more than one application. Different application can perform different tasks on this data.

Features & Advantages of XML :-

① XML Separates data from HTML

If you need to display dynamic data in your HTML, it will take a lot of work to edit the HTML each time the data changes.

With XML, data can be stored in seperate XML files.

② XML Simplifies data sharing

In real world, computer systems & database contain data in incompatible formats.

XML data is stored in plaintext format. this provides a S/W & H/W independent way of storing data.

③ XML Simplified data transport

One of the most time-consuming challenges for developers is to exchange data between incompatible systems over the internet - Exchanging data as XML greatly reduces this complexity, since the data can be read by different incompatible applications.

④ XML Simplifies platform change

Upgrading to new systems is always time consuming.

XML data is stored in text format. this makes it easier to expand or upgrade to new operating systems, new applications or browsers without losing data.

Scanned by CamScanner

5. XML increases data availability
Different applications can access your data, not only in HTML pages, but also from XML data sources.
With XML, your data can be available to all kinds of reading machines.

6. XML can be used to create new internet languages
A lot of new internet languages are created with XML.
1. XHTML
2. WSDL for describing available web services.
etc.

XML Example:-

XML documents create a hierarchical structure looks like a tree so it is known as XML Tree that starts at the "root" and branches "to the leaves"

Example of XML document, message.xml

```
<?xml version="1.0"?>
<note>
    <to> Tove </to>
    <from> Jani </from>
    <heading> Remainder </heading>
    <body> Don't forget me this weekend! </body>
</note>
```

The first line is the XML declaration. It defines the XML version (1.0).
The next line describes the root element of the document (note)
The next 4 lines describe 4 child elements of the root (to, from, heading, and body).
And finally the last line defines the end of the root element (/note)

Scanned by CamScanner

→ All elements can have sub elements (child)

```
<root>
    <child>
        <subchild> --- </subchild>
    </child>
</root>
```

~~All~~ the terms parents, child, subchild are used to describe the relationships b/w elements.

Another Example :-

Books.xml

```
<bookstore>
    <book category="Cooking">
        <title lang="en">Everyday Italian</title>
        <author>Giada De Laurentiis</author>
        <year>2005</year>
        <price>30.00</price>
    </book>
    <book Category="children">
        <title lang="en">Harry potter</title>
        <author>J.K. Rowling</author>
        <year>2005</year>
        <price>29.99</price>
    </book>
    <book Category="web">
        <title lang="en">Learning XML</title>
        <author>Erik T. Ray</author>
        <year>2003</year>
```


Scanned by CamScanner

<price> 39.95 </price>

</book>

</bookstore>

The root element in the example is <bookstore>. All elements in the document are contained within <bookstore>

The <book> element has 4 children: <title>, <author>, <year>, <price>.

Example:-

email.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<emails>
    <email>
        <to> vimal </to>
        <from> Sonoo </from>
        <heading> Hello </heading>
        <body> Hello brother, how are you! </body>
    </email>
    <email>
        <to> peter </to>
        <from> Jack </from>
        <heading> Birthday wish </heading>
        <body> Happy birthday Tom! </body>
    </email>
```


Scanned by CamScanner

```
<email>
    <to> James </to>
    <from> Jaclin </from>
    <heading> Morning walk </heading>
    <body> please start morning walk to stay fit!! </body>
</email>
</emails>
```

Defining XML Tags:-

An xml document consists of the following parts:

1. prolog
2. body

Prolog:- this part of xml document may contain the following parts:

→ XML Declaration
→ Optional processing instruction
→ Comments
→ Document Type Definition

→ XML Declaration:-

Every xml Document should start with a one-line xml declaration.

```
<? xml version="1.0" ?>
```

the declaration may use two optional attributes:

encoding

```
<? xml version="1.0" encoding="UTF-8" ?>
```

the UTF-8 (Unicode Transformation format) is used which has the same character set as ASCII.

Standalone

this optional attribute indicates whether the document can be processed as a standalone document. if "Yes" is

Scanned by CamScanner

specified document must contain external DTD. it value "no" nothing is specified.

<?xml version="1.0" encoding="UTF-8" standalone="no"?>

→ **Processing instruction:-**

it starts with <? and end with ?>. they allow XML documents to contain special instructions that are used to pass parameters to the application.

<?xml-stylesheet href="simple.xsl" type="text/xsl"?>

this processing instruction states that the xml document should be transformed using the style sheet simple.xsl.

→ **Comments:-**

Like Html, Comments may appear any where in the xml documents. An xml Comment starts with <!-- and ends with -->. Everything within these character sequences will be ignored by browser.

<!-- Comment text -->

→ **Document Type Declaration:-**

It is used to specify the logical structure of the xml document. the structure is specified by constraint on what tags can be used and where.

Body:- the xml document contains textual data marked up by tags. It must have one element called root element.

<greeting> Helloworld! </greeting>

the root element contains other elements.

Scanned by CamScanner

```
<?xml version="1.0" encoding="UTF-8"?>
<Contact>
    <person>
        <name> B.S.Roy </name>
        <number> 9674141 54 </number>
    </person>
    <person>
        <name> G. Mahapatra </name>
        <number> 944 180 4070 </number>
    </person>
</Contact>
```

this xml document has the root element <contact> which has two <person> elements. Each person has two elements <name> & <number>.

Elements:-

An xml element consists of a starting tag, an ending tag, and its contents & attributes. the content may be a simple text, or other elements or both. Each element contain different types of data that are stored in the XML document.

A tag begins with the less-than character ('<') and ends with greater than ('>') character. It takes the form <tag-name>. every tag must have a corresponding ending tag. </tag-name>

Ex:- <greeting> Helloworld! </greeting>

Here, <greeting> is the starting tag and </greeting> is the ending tag. everything between these two form an element. this element has only text content "Hello world!".

Scanned by CamScanner

Anatomy of an element:-

An element consists of an opening tag, a closing tag & Contents.

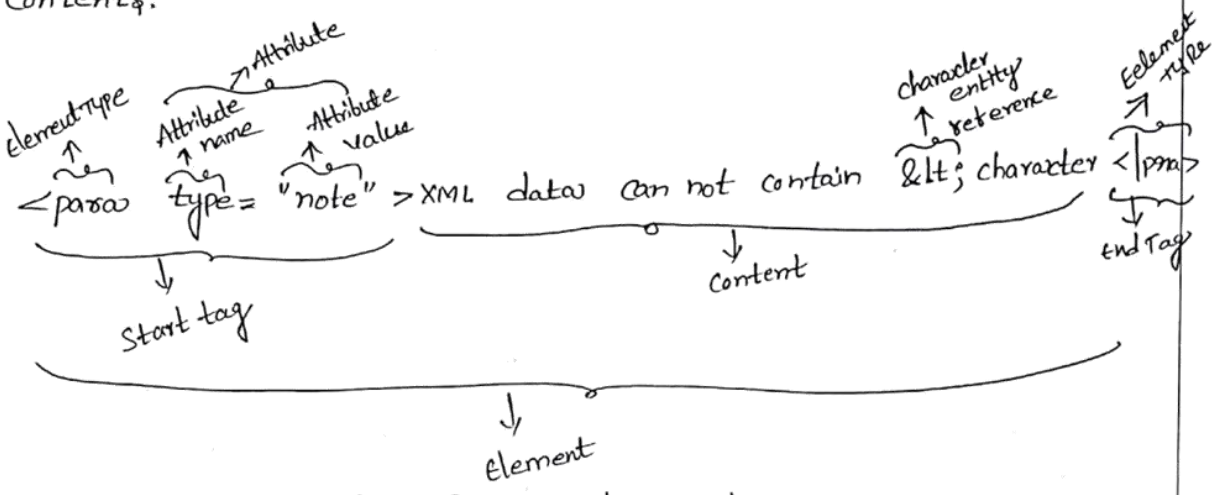


fig:- Structure of an element

The above figure shows the structure of an element. The name element is para. The opening tag is written within angular brackets. The closing tag is written exactly the same way, except that a slash (/) is added before the element name.

The attributes are declared with in the opening tag. For example, the para element has only one attribute type whose value is note. The attribute value must be quoted.

For, the para element, the Content is text. The text may Contain entities that refer to Some piece of text to be ~~submitted~~ Substituted while processing. An entity Starts with '&' and ends with ';' characters.

Scanned by CamScanner

Naming Rules:-

the following rules must be obeyed while selecting element name:

→ Names can only contain letters, digits and some other special characters.

→ Names can not start with a number.

→ Names must not contain the string "xml", "XML", or "xml".

→ Names cannot contain white space(s).

→ XML is a case sensitive. (you must follow whatever combination of upper or lower case. So, you can't say <Body> ...</body> both upper & lower case must match).

Empty Elements:-

Empty elements are those do not have any content.

```
<line width="100"></line>
(or)
<line width="100"/>
```

Attributes & Values:-

Attributes are used to describe elements or to provide more information about elements. they appear in the starting tag of the element.

Syntax:-

```
<element-name attribute-name="attribute-value">------
</element-name>
```

Ex:-

```
<employee gender="male">....</employee>
```

An element can also have multiple attributes:

```
<employee gender="male" id="12345">...</employee>
```

Note:- XML attributes value must always be quoted. we can use single or double quotes.

Scanned by CamScanner

well-formed xml:-

An xml document is said to be well-formed if it contains text & tags.

The following rules must be followed by the XML documents to be well-formed.

→ An XML document must have one & exactly one root element.

Consider, the following well-formed XML document. It has one root element <contact>, which contains one <person> element, which contains two elements, <name> & ~~contact~~ <number>.

```
<contact>
    <person>
        <name> Sai </name>
        <number> 0001 </number>
    </person>
</contact>
```

the following document is not well-formed as it has two top-level elements, <contact> & <phonebook>

```
<contact>
    <person>
        <name> Sai </name>
        <number> 00012356 </number>
    </person>
    </contact>
    <phonebook>
        <name> Geetha </name>
        <number> 12345678 </number>
    </phonebook>
```

→ All tags must be closed.

Every element must have a closing tag corresponding to its opening tag,

not well formed:

```
<!-- Incorrect -->
<name> Sai 123
```

Scanned by CamScanner

well formed: <!-- Correct -->

<name> Sai </name>

A tag must be closed even if it doesnot have any ~~elements~~ content (empty element).

In Html, Some tags such as <img>, <br>, <hr> donot require any closing tag. However, in xml closing tags are always needed.

incorrect in xml :

correct in xml :
</br> 或

→ All tags must be properly nested,

Elements must not overlap. An ending tag must have the same name as the most recent unmatched tag.

incorrect: thes is incorrect nesting

correct: This is correct nesting

→ XML tags are Case-Sensitive,

Unlike Html, xml element names are Case sensitive.

So, Message & message refer to two different names.

not well formed: <message> Hi </Message>

Well-formed: <message> Hi </message>

→ Attributes must always be quoted,

the value of an attribute must be quoted by double inverted commas,

correct: < speed unit = "rpm" > 7200 </speed>

incorrect: < speed unit = rpm > 7200 </speed>

→ Certain characters are reserved for processing,

Certain characters cannot be used in the xml document directly. For example, '<', '>' and " " can not be used.

Scanned by CamScanner

<!-- Incorrect -->

<Condition> if Salary < 1000 then </condition>

<!-- Correct -->

<Condition> if Salary < 1000 then </condition>

Predefined Entities:-

Entityname	Entitynumber	Description	Character
<	<	lessthan	<
>	>	greaterthan	>
&	&	ampersand	&
"	"	quotationmark	"
'	'	apostrophe	'

Valid xml:-

well-formed xml documents obey only basic well-formedness Constraints. So, valid xml documents are those that,

- Are well-formed
- Comply with rules specified in the DTD or schema.

Validation:-

It is a method of checking whether an xml document is well-formed & Conforms to the rules specified by a DTD or schema. Many tools are available to validate xml against DTD or schema. Unix/Linux provide one application called xmllint for this purpose,

xmllint --valid Sample.xml

(command line xml tool)

Rules:-

1. It must begin with the XML declaration
2. It must have one unique root element

Scanned by CamScanner

3. All start tags of xml documents must mach end tags
4. XML tags are Case Sensitive
5. All elements must be closed
6. All elements must be properly nested
7. All Attributes values must be quoted
8. xml entities must be used for special characters.

Displaying XML :-

XML documents do not carry information about how to display the data. new tags can be added in the xml document. web browsers donot have any idea about the tags used in the Xml file. So, if you open an xml file in a browser, the entire content (data & tags) is displayed in a tree like structure.

there are many ways to display data stored in an xml document. the two common methods are,

- CSS (Cascading Style sheets)
- XSL [extensible stylesheet language)

RSS already we discussed. XSL was specially designed for xml

Save books.xsl

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="books.xsl"?>
<bookstore>
    <book category="literature">
        <title lang="beng">Sanchoitao</title>
        <author>Ravindranath Tagore</author>
        <year>2009</year>
        <price>200.00</price>
    </book>
```

Scanned by CamScanner

```
<book  Category ="literature">
      <title  lang ="en"> Gitanjali </title>
      <author> Ravindranath Tagore </author>
      <year> 2008 </year>
      <price> 29.00 </price>
  </book>
  <book  Category =" web">
      <title  lang =" en"> Essential xml </title>
      <author> Don Box </author>
      <year> 2000 </year>
      <price> 150 </price>
  </book>
</bookstore>
```

save
books.xml

```
<?xml  version ="1.0"  encoding ="ISO - 8859-1" ?>
<xsl : stylesheet  version =" 1.0"
                   xmlns:xsl ="http:// www.w3.org /1999/xsl / Transform">
<xsl :output  method =' html'  version='1.0'  encoding ='UTF -8'
                                indent ='yes' />
  <xsl: template  match =" /">
  <html>
    <body>
        MyBook  Collection:
      <table  border = " 1" >
```

Scanned by CamScanner

```
<tr bgcolor="yellow">
    <th> title </th> <th> Author </th> <th> year </th>
    <th> price </th>
</tr>
<xsl:for-each select="bookstore/book">
<tr>
    <xsl:for-each select="./*">
        <td> <xsl:value-of select="."/> </td>
    </xsl:for-each>
</tr>

</xsl:for-each>
</table>
</body>
</html>
</xsl:template>

</xsl:stylesheet>
```

Scanned by CamScanner

Document Type Definition:-

A DTD defines the legal building blocks of an XML document. It defines the document structure with list of legal elements & attributes. A DTD can be declared inside the XML document or as an external reference.

DTD
- Internal DTD
- External DTD

Internal DTD declaration:-

If the DTD is declared inside the XML file that can be called as internal DTD. It should be wrapped in a DOCTYPE definition with the following syntax,

```
<!DOCTYPE root-element [element-declarations]>
```

Ex:-

```
<?xml version="1.0"?>
<!DOCTYPE note[
<!ELEMENT note (to, from, heading, body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
```

rootelement

of type 'PCDATA'

```
<note>
  <to>Tove</to>
  <from> jani </from>
  <heading> Remainder
    </heading>
  <body>Don't forget me this
    weekend</body>
</note>
```

Limitations of DTD

- → DTD is quite different from basic XML doc
- → we can't use multiple DTD to validate xm
- → DTD are not object oriented.

Scanned by CamScanner

External DTD:-

If the DTD is declared in an external file, that can be called as external DTD. this file can be saved with an extension of ".DTD".

To import the external DTD file into xml file we have to use a DOCTYPE definition with following syntax:

<!DOCTYPE root-element SYSTEM "filename">

↓ after xml prolog & before the root element

↓ it must occur

↓ location of DTD file is local (or) PUBLIC

Ex:-

note.DTD

note.xml

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
    <to> Tove </to>
    <from> Jani </from>
    <heading> Remainder </heading>
    <body> Don't forget me this weekend! </body>
</note>
```

→ with a DTD, each of your xml files can carry a description of its own format.

Building blocks of Xml Document:-

Elements

these are the main building blocks of both xml & Html documents.

html Ex:- body, table

xml Ex: message, note

DTD

<! ELEMENT element-name (element-content)>

category

Empty elements

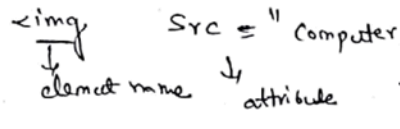
<!ELEMENT element-name Empty> xml

continue next page

Scanned by CamScanner

Attributes:- these are provide extra information about elements. these are placed always inside the opening tag of element.

Attributes always come in name / value pairs.



Declaring Attributes:- in DTD

<!ATTLIST element-name attribute-name attribute-type default-value>

Ex:- DTD : <!ATTLIST Payment type CDATA "check">

xml : <Payment type = "check" />

Entities:-

Entities are variables used to define shortcuts to standard text or special characters.

Syn:- <!ENTITY entity-name "entity-value">

Ex:- DTD : <!ENTITY WT "Web Technology">

xml : <subject> &WT; </subject>

An entity has three parts : ampersand (&), an entity name, & semi colon (;).

  → (no-breaking space) to insert the extra space in a document.

PCDATA:-

It means parsed character data. In PCDATA all text will be parsed & special characters will be replaced with their corresponding characters.

the character data should not contain any &, <, or > if these are contain in document errors will be occured. represented by example: <, > entities respectively

Scanned by CamScanner

Elements with any content:-

Category keyword ANY, can contain any combination of parsable data.

<!ELEMENT element-name ANY>

Ex:- <!ELEMENT note ANY>

elements with children

<!ELEMENT element-name (child1,child2,...)>

Ex:- <!ELEMENT note (to, from, heading,body)>

Keyword Ex:-

* → Declaring zero or more occurances of an elements

<!ELEMENT element-name (child-name*)>

Ex:-

<!ELEMENT note (message*)>

can occur zero or more times inside note element

? → zero or more time occurs

1 → indicates an option

+ → one or more occurs

? → specifies the exact order of occurrence

() → a group of expressions to be matched to pattern

<![

<email> jai@mrce.in </email>

<name> vimal </name>

<lastname> sai </lastname>

<![CDATA [

CDATA:-

Character Data. CDATA is text that will NOT be parsed by a parser.

* PCDATA is text that will be parsed by a parser. Tags inside the text will be treated as markup & entities will be expanded.

* CDATA is text that will not be parsed by a parser. Tags inside the text will not be treated as markup & entities will not be expanded.

Scanned by CamScanner

Similarly, we can write Server.xml,

```
<s:table>
   <s:row>
         <s:col> Jsp </s:col>
         <s:col> Asp </s:col>
   </s:row>
</s:table>
```

Who guarantees that the prefixes used by different developer will be unique? When using prefixes in XML, a namespace for the prefix must be defined.

The namespace can be defined by an xmlns attribute in the start tag of an element.

The namespace declaration has the following syntax:

```
xmlns:prefix="URI"
```

site address

```
<root>
   <c:table   xmlns:c="namespaceURI">
   <c:tr> <c:td> JavaScript </c:td>
          <c:td> VBScript </c:td>
   </c:tr>
   </c:table>
   <s:table   xmlns:s="namespaceURI">
   <s:row> <s:col> Jsp </s:col>
           <s:col> Asp </s:col>
   </s:row>
   </s:table>
</root>
```

Scanned by CamScanner

The purpose of using an URI is to give the name space a unique name.

URI (Uniform Resource Identifier) is a string of characters which identifies an internet Resource.

The most common URI is Uniform Resource locator (URL) which identifies an Internet domain address.

Scanned by CamScanner

DTD VS XSD

DTD	XSD
1) DTD Stands for document type definition.	1) XSD stands for XML Schema definition.
2) DTD are derived from SGML Syntax.	2) XSD's are written in XML
3) DTD doesn't support data types	3) XSD supports datatypes for elements & attributes
4) DTD doesn't support namespace	4) XSD supports namespace
5) DTD doesn't define order for child elements.	5) XSD defines order for childelments
6) DTD is not extensible	6) XSD is extensible
7) DTD is not simple to learn	7) XSD is simple to learn because you don't need to learn new language.
8) DTD provides less control on XML Structure.	8) XSD provides more control on XML Structure.

Scanned by CamScanner

XML Schema :-

- XML Schema is an XML based alternative to DTD.
- An XML Schema is used to define the structure of an XML document. It is like DTD but provides more control on XML Structure.
- An XML Schema language is also referred to as XML schema definition (XSD).

An XML schema :-

- Defines elements that can appear in a document.
- Defines attributes that can appear in a document.
- Defines which elements are child elements.
- Defines the order of child elements
- Defines the number of child elements.
- Defines whether an element is empty or can include text.
- Defines datatypes for elements & attributes.
- Defines default & fixed values for elements & attributes

Advantages

XML Schemas are extensible to future additions.

XML schemas are richer & more powerful than DTD's

XML schemas support data types.

XML schemas support namespaces

Example:- let's create ~~xsd~~ a schema file, employee.xsd

```
<?xml version = "1.0"?>
<xs: schema xmlns:xs = " http://www.w3.org/2001/XMLSchema"
        targetNamespace = " http://www.javatpoint.com"
        xmlns = " http://www.javatpoint.com"
        elementFormDefault = "qualified" >
```

Scanned by CamScanner

NameSpace :-

These are used for providing uniquely named elements & attributes in xml document. XML was developed to be used by many applications. If many want to Communicate using xml documents, a potential problem may occur.

The problem :-

In xml document, element names & attribute names are selected by developers. According to the XML 1.0 Specification, element & attribute names are unstructured flat Strings. So, name Conflicts may occur when merging xml document from different developers to get a final document.

Consider, the following XML document (client.xml) which represents an Html table of client side technologies.

```
<table>
    <tr> <td> JavaScript </td> <td> VBScript </td>
    </tr>
</table>
```

this XML document has the root element table. the following XML document (Server.xml) Carries information about a table of server-side technologies.

```
<table>
    <row><col> Jsp </col> <col> Asp </col> </row>
</table>
```

Let us, now merge these two XML documents to obtain a Single XML document as follows.

```
<technology>
    <table>
        <tr> <td> JavaScript </td> <td> VBScript </td> </tr>
    </table>
    <table>
        <row> <col> Jsp </col> <col> Asp </col> </row>
    </table>
</technology>
```

Scanned by CamScanner

this new XML document has two <table> elements which have different Content & meaning. So, if we query a parser to find the table Containing Server-side technologies, it fails. one Simple but not too- flexible way to resolve this problem is to embed them in two different elemets as follows:

```
<technology>
    <client>
        <table>
            <tr> <td> Javascript </td> <td> VBscript </td></tr>
        </table>
    </client>
    <server>
        <table>
            <row> <col> Jsp </col> <col> Asp </col> </row>
        </table>
    </server>
</technology>
```

Now, the first & second table elements can be uniquely referred by refferring to their parent element names, which are unique. Needless to say, this mechanism needs not only extra elements to be inserted, but also knowledge about the XML documents to be merged.

Solution:-

XML namespace provides a Simple, Straightforward, but elegant way to distinguish between element names in the xml document, no matter where they come from. xml namespace suggests that we use a prefix with every element. So, client.xml can now be written like this:

```
<c:table>
    <c:tr> <c:td> Javascript </c:td>
           <c:td> VBscript </c:td>
    </c:tr>
```

Scanned by CamScanner

```
<xs:element name="employee">
<xs:complexType>
<xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
    <xs:element name="email" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

defines the element name employee

the complex type is a sequence of elements

set the element, is employee complex type

the element firstname is a string/text type

Let's see the XML file using XML schema (or) XSD file

employee.xml

```
<?xml version="1.0"?>
<employee xmlns="http://www.javatpoint.com"
          xmlns:xsi="http://www.w3.org/2001/xmlschema-instance"
          xsi:schemaLocation="http://www.javatpoint.com employee.xsd">
  <firstname> vimal </firstname>
  <lastname> jaiswal </lastname>
  <email> vimal@javatpoint.com </email>
</employee>
```

Scanned by CamScanner

XML schema datatypes :-

there are two types of datatypes in XML schema,

1) Simple Type 2) Complex Type

Simple Type :- It contains only Text. They can not have sub elements ~~attributes~~. The text can be various datatypes such as, Strings, Date, Time, Numeric [Decimal, Integer], Boolean DataTypes.

Syntax:- <xs: element name = "xxx" type = "yxy" />

Ex:- Here, Some XML elements

<last name> Refsnes </lastname>
< age > 36 < /age>
<data born> 1970-09-27 < /dateborn>

And here are the corresponding simple element definitions:

<xs:element name = "lastname" type = "xs:string" />
<xs:element name = "age" type = "xs:integer" />
<xs:element name = "dateborn" type = "xs:date" />

Complex Type:-

It contains sub elements, attributes etc many times they are made up of one (or) more simple elements.

XSD indicators:-

There are five indicators,

Order indicators
- → All
- → choice
- → sequence

Occurrence indicators
- → max Occurs
- → Min Occurs

Order Indicators :-

1) All :-

the <all> indicator specifies that the child elements can appear in any order, & that each child element must occur only once:

```
<xs: element name = " person">
<xs : complexType >
< xs : all>
    <xs: element name = "firstname" type = " xs:string" />
    <xs: element name = "Lastname" type = " xs: string" />
< /xs: all>
</xs:element>
```

Scanned by CamScanner

② <choice> :-

It specifies that either one child (or) another element can occur:

```
<xs:element name="person">
<xs:complexType>
  <xs:choice>
      <xs:element name="employee" type="xs:employee"/>
      <xs:element name="member" type="xs:member"/>
  </xs:choice>
</xs:complexType>
</xs:element>
```

③ sequence :-

It specifies that the child elements must appear in a specific order.

```
<xs:element name="person">
  <xs:complexType>
  <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
</xs:element>
```

④ Occurence:-

maxOccurs:- It specifies the maximum no. of times an element can occur:

```
<xs:element name="person">
  <xs:complexType>
  <xs:sequence>
      <xs:element name="full_name" type="xs:string"/>
      <xs:element name="child_name" type="xs:string"
                                  maxOccurs="10"/>
  </xs:sequence>
  </xs:complexType> </xs:element>
```

Scanned by CamScanner

⑤ minOccurs:- It specifies the minimum no.of times an element can occur,

```
<xs:element name="person">
  <xs: complexType>
    <xs: sequence>
      <xs: element name="full_name" type="xs:string"/>
      <xs: element name="child_name" type="xs:string"
                 maxOccurs="10" minOccurs="0"/>
    </xs: sequence>
  </xs: complexType>
</xs:element>
```

to occur unlimited no.of times maxOccurs="unbounded"

Scanned by CamScanner

Unit - II

XML Parser:

An XML Parser converts an XML document into another format; these formats are XHTML and DHTML. This process is also called as XML Processor.

DOM Parser:

> DOM is a tree representation of an XML document in memory.

> We can access information of an XML document by interacting with the tree nodes.

> Useful for smaller applications.

> We can insert or delete a node.

> Traversing is done in any direction in DOM approach.

> In this method the entire XML document is stored in the memory before actual processing. Hence it requires more memory.

> Checking well-formedness of XML document using DOM API.

Scanned by CamScanner

XML DOM Parser:

The XML DOM contains methods (functions) to traverse XML trees, access, insert and delete nodes. However, before an XML document can be accessed and manipulated it must be loaded into an XML DOM object. An XML parser reads XML, and converts it into an XML DOM object that can be accessed with Java Script.

Loading an XML File:

```
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>
<script>
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function()
{ if (this.readyState == 4 && this.status == 200)
  { myfunction(this);
  }
};
xhttp.open("GET", "books.xml", true);
xhttp.send();
```

Scanned by CamScanner

```
function myFunction (xml)
{
    var xmlDoc = xml.responseXML;
    document.getElementById ("demo").innerHTML =
    xmlDoc.getElementsByTagName ("title")[0].childNodes[0].
      nodeValue;
}
</script>
</body>
</html>
```

Loading an XML String

```
<html>
<body>
<p id="demo"></p>
<script>
var text, parser, xmlDoc;
text = "<bookstore><book>" + "<title>Everyday Italian</title>'+
"<author>Giada De Laurentiis</author>" + "<year>2005
  </year>" + "</book></bookstore>";

parser = new DOMParser();
xmlDoc = parser.parseFromString (text, "text/xml");
document.getElementById ("demo").innerHTML =
xmlDoc.getElementsByTagName ("title")[0].childNodes[0].node
Value;
```


Scanned by CamScanner

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

XML DOM Properties.

These are some typical DOM properties:

nodeName - finding the name of the node.

nodeValue - Obtaining the value of the node.

parentNode - Getting the parent node.

childNodes - Getting the child nodes.

attributes - Getting the attributes.

XML DOM methods

getElementsByTagName (name) - get all elements with a specified tag name.

appendChild (node) - insert a child node.

removeChild (node) - remove a child node.

Scanned by CamScanner

XSL:-

XSL Stands for Extensible style sheet Language. It describes how the XML document should be displayed.

XSLT:-

XSLT is a language for transforming XML documents into XHTML documents (or) to other XML documents.

XSLT is the most important part of XSL

with XSLT you can add | remove elements & attributes to or from the output file. you can also rearrange and sort elements, perform tests and make decisions about which elements to hide and display a lot more.

XSLT Elements

<xsl:stylesheet> and <xsl:transform> are completly synonymous & either can be used.

xslt stylesheet, xslt transform, xslt template, xslt value-of

XSLT for-each, xslt-if, xslt when, xslt choose.

Syntax:-

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/2001/XSL/Transform">
```

Create an xsl stylesheet:-

"cdcatalog.xsl"

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/2001/XSL/Transform">

<xsl:template match="/">
<html>
<body>
<h2> My CD Collection </h2>
```

one or more set of rules. It Contains rules to apply when a specified node is matched

Scanned by CamScanner

```
<table border="1">
<tr background-color="Red">
  <th> Title </th>
  <th> Artist </th>
</tr>
<xsl:for-each select="catalog/cd">
  <tr>
    <td> <xsl: value-of select="title"/> </td>
    <td> <xsl: value-of select="artist"/> </td>
  </tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl: stylesheet>
```

It is used to select every xml element of a specified node-set.

It is used to extract the value of xml element & add it to the output stream of the transformation

It uses select attribute to identify elements in source document to which template should be applied.

Link the xsl style sheet to the XML document,

```
cd catalog.xml,
<?xml version="1.0" encoding="ISO-8859-1"?
<?xml-stylesheet type="text/xsl"
                 href="cdcatalog.xsl"?>

<catalog>
  <cd>
    <title> Empire Burlesque </title>
    <artist> Body Dylan </artist>
    <country> USA </country>
    <company> Columbia </company>
    <price> 10.90 </price>
    <year> 1985 </year>
  </cd>
</catalog>
```

Scanned by CamScanner

SAX parser:- [Simple API to XML]

- SAX is an event based parsing method used to parse the given XML document.

In this the parsing is done by generating the sequence of events (or) it calls handler functions.

Useful to parsing the large xml document because this approach is event based.

XML gets parsed node by node does not require large amount of memory.

We can insert (or) delete a node.

Top to bottom traversing is done in this approach.

Packages:-

javax.xml.parsers
org.xml.sax
org.xml.sax.ext
org.xml.sax.helpers

Steps to implement SAX:-

Step:1

specifying parser

Step:2

First create an instance of a parser factory & then use that to create a SAX parser object.

Ex:- SaxParserFactory fac = SaxParserFactory.newInstance();
SaxParser parser = fac.newSAXparser();

Scanned by CamScanner

Step:-3 Create a Content. Handler,

Responds to parsing events, primary event methods. (termed as callbacks)

Ex:- startDocument()
endDocument()
startElement()
endElement()
Characters()
ignorableWhitespace()

· Displaying the contents of xml document using SAX api.

Ex:- Parsing_SAXDemo1.java

Ex:-

```
import java.io.*;
import org.xml.sax.*;
import org.xml.sax.helpers.*;
public class Parsing_SAXDemo
{
public static void main(String[] args) throws IOException
{ try
{
System.out.println("enter the name of xml doc");
BufferedReader input = new BufferedReader(new InputStreamReader(System.in));
String file_name = input.readLine();
File fp = new File(file_name);
if (fp.exists())
{
try
{
XMLReader reader = XMLReaderFactory.create XMLReader();
reader.parse(file_name);
System.out.println(file_name + " is wellformed");
```

class is used to read the text from a character based input stream

Scanned by CamScanner

```
catch (Exception e)
{
System.out.println (file_name + " is not well formed");
System.exit(1);
}
}
else
{
System.out.println ("file is not present:" + file_name);
}
}
Catch (IOException ex)
{
ex.printStackTrace();
}
}
}
}
```

In the above example enter the xml file name as input. If the file is well formed that gives the output as the given file is well formed.

Scanned by CamScanner

Date
5/2/18

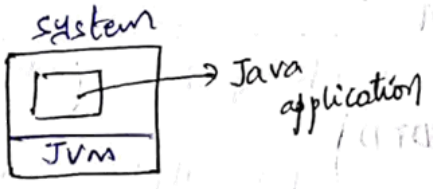
UNIT - III

Servlets

Why we need to go for a web apps in Java?

⇒ Using JDK s/w we can develop 2 standalone - 2-Tier client & server apps.

⇒ A standalone app will execute on local JVM of system & provides services to single ~~di~~ client. A stand alone app can't be accessed from another place.



* 2 - Tier - app contains server and client.

* The service of a server app can be accessed from client app across n/w.

Scanned by CamScanner

Draw backs of 2-client server App's:-

→ To know exact location of server

→ The client app need to be create at client system.

→ we need s/w both client & server side.

→ fixed no. of client can only communicate with server Apps

→ To overcome above programs we get web app. in Java.

→ A application is a server side app it was on a server and provides service to multiple client across n/w or Internet

Types of web Application:-

(1) Static web App

(2) Dynamic web App

Static web App:- It only concentrate on Display of the content.

e.g:- online tutorial

Scanned by CamScanner

Dynamic web app:- It will concentrate on the dynamic service to the client :- eg:- Aponline which provides dynamic payment of electricity bill, IT e.t.c,

Passive & Active Resources.

Which doesn't require any processing on server side	Whose processing is done at serverside.

Different Resources Req develop the and run the web application

(1) Client side — HTML, AJAX

(2) Server side — Tomcat, IIS, weblogic, webspheir, glass fish

(3) Browser —

(4) Database — oracle,

Scanned by CamScanner

What is webserver?

A server is an app which accepts request from client across network and send the request to appropiate web app collects response from the web app & finally provides that response on to a client browser.

Web container:- It is a Java application along with a server and provides run time support for dynamic resource for web application.

Adv:-

→ Communication support
→ life cycle management
→ Multi Threading.
→ security

Common Gateway Interface:- (CGI)

It is used for creating dynamic web application. CGI was introduced by opensource community called NCSA [National Centre for Suppor Commuting Application].

Scanned by CamScanner

CGI is a programm

CGI App are created using P.L. Called perl (pratice extraction reporting language). There are two major problems in CGI.

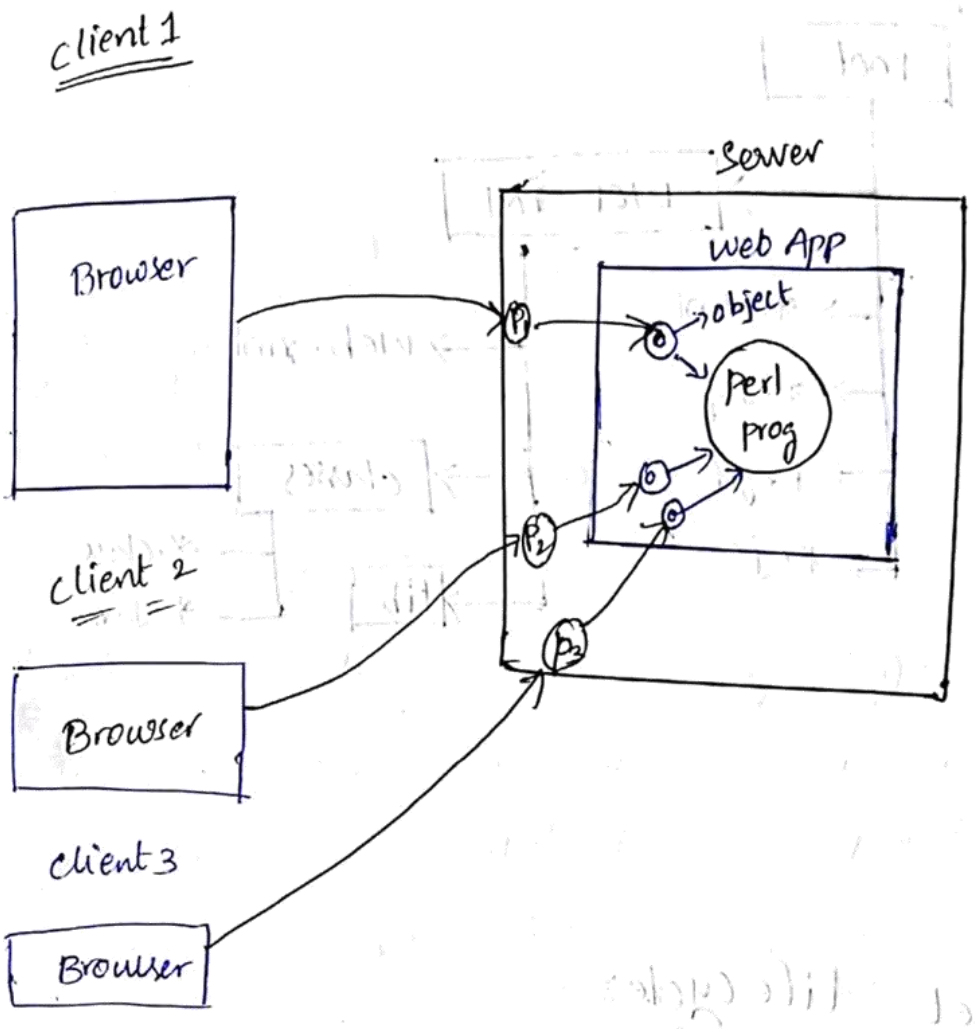
(1) CGI creats an os level process and They object of perl program.

for each request to provide the response. if the clients are increased Then The os level processes and CGI program objects also increased in a server. It means that increasing the burden on server. So, that performance of server is decreased.

(2) The pearl is a non-secure programming lang So, CGI is failed to provide security for data & logic.

In order to overcome the above two draw back The "Sun micro System" Introduced the next Technology for creating Dynamic app as Servlet Technology.

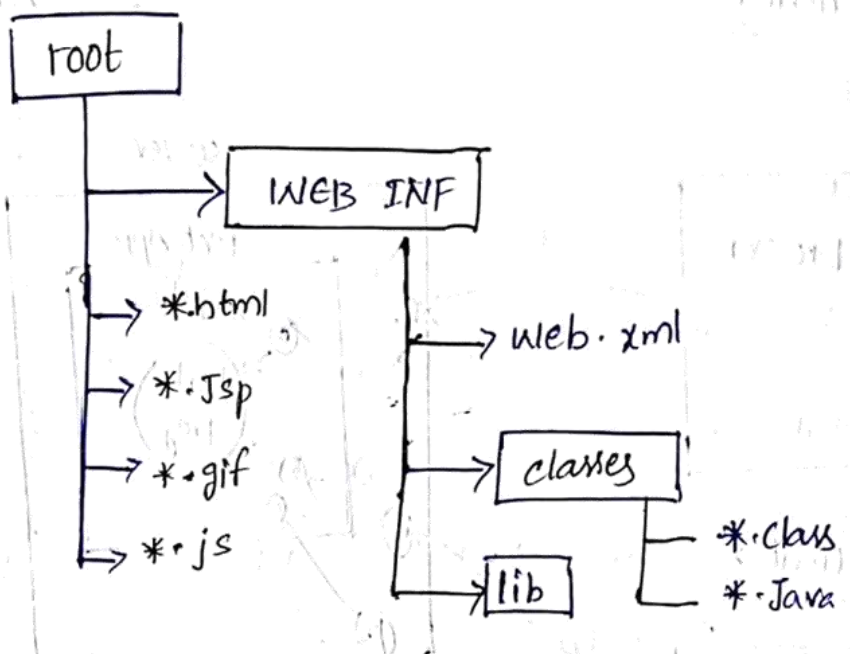
Scanned by CamScanner



Web application directry structure:- To create dynamic web app used Technologies servlet & JSP we need to follow some standard web app directory structure given by Sun micro systems

⇒ By following web app directry structure a web app gets principle called WODAs (Write once & Deploy Any server).

Scanned by CamScanner

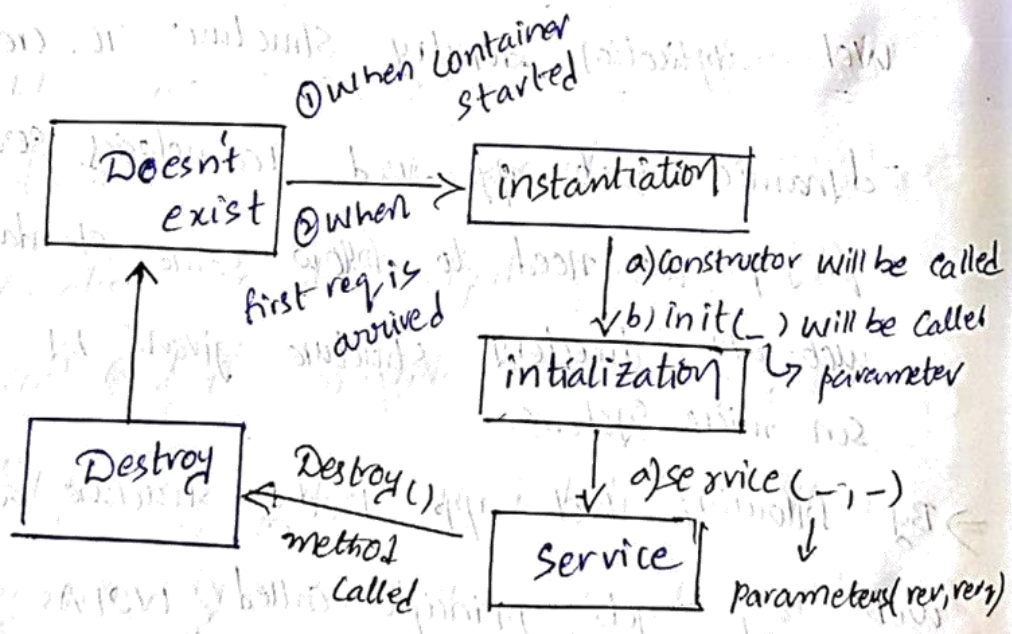


Date

6/2/18

***

Servlet life cycle:



Scanned by CamScanner

A servlet programmer create a servlet class and implement some bussiness logic required init and Then generate a class file for The Servlet.

After generating a class file Then a programer Deploy the Servlet class into server side There is web container which manage The life and death of the a servlet object in server. It means a servlet life cycle is managed by servlet container.

A servlet container manages a servlet life cycle by using set of well define steps

(1) Doesn't exist

(2) instantiation

(3) initialization

(4) service

(5) Destroy

By default a servlet object is unaviable in a server. It means a servlet object

Scanned by CamScanner

doesn't exist in a server.

→ When a servlet container is started Then servlet container loads a servlet class into memory (JVM) and Then The Container Instanciates a server. by This step a servlet object is moved from doesn't exist to instantiation.

→ By default a servlet container delays instantiation a first request is called arived for a servlet.

→ we can inform a servlet Container Through deployment so That a servlet Container creates a servlet object when ever a container is executed.

→ After a servlet is instatiated Then immediately constructor will be executed

→ Once a init method execution is completed Then The servlet will be ready to provide service to client request.

Scanned by CamScanner

→ once init method execution is completed then the servlet will be ready to provide service to the client request.

→ Now, reads a client Request executes Business logic & generates response to client. This stage is called service.

→ The service method is called for each Request given by client to the servlet.

→ It means container invokes service method for multiple times

→ After service is done ie, when servlet no longer required in server then the container will destroy a servlet.

→ Once a servlet object is distroyed then the servlet object becomes doesn't exist in the server.

Scanned by CamScanner

Life cycle method

(i) init():-

```
public void init(servlet Config config)
                    throws servlet expression
{
    =
}
```

(ii) Service:-

```
public void service(Servlet Request req,
                    Servlet Response res)
        Throwy servlet Exception, Io Exception
{
    =
}
```

(iii) Destroy:-

```
public void destroy()
{
    =
}
```

Scanned by CamScanner

Deployment Descriptor:

⇒ A servlet programer creates a servlet with the required Bussiness logic but in the server a servlet Container will manage a servlet to inform about a servlet to the Container about a programmer writes a servlet config file.

⇒ while writing a servlet Config A programmer will config the following three names of servlet

(i) fully qualified name of servlet

(ii) logical name of servlet.

(iii) URL pattern name of servlet

Web.xml

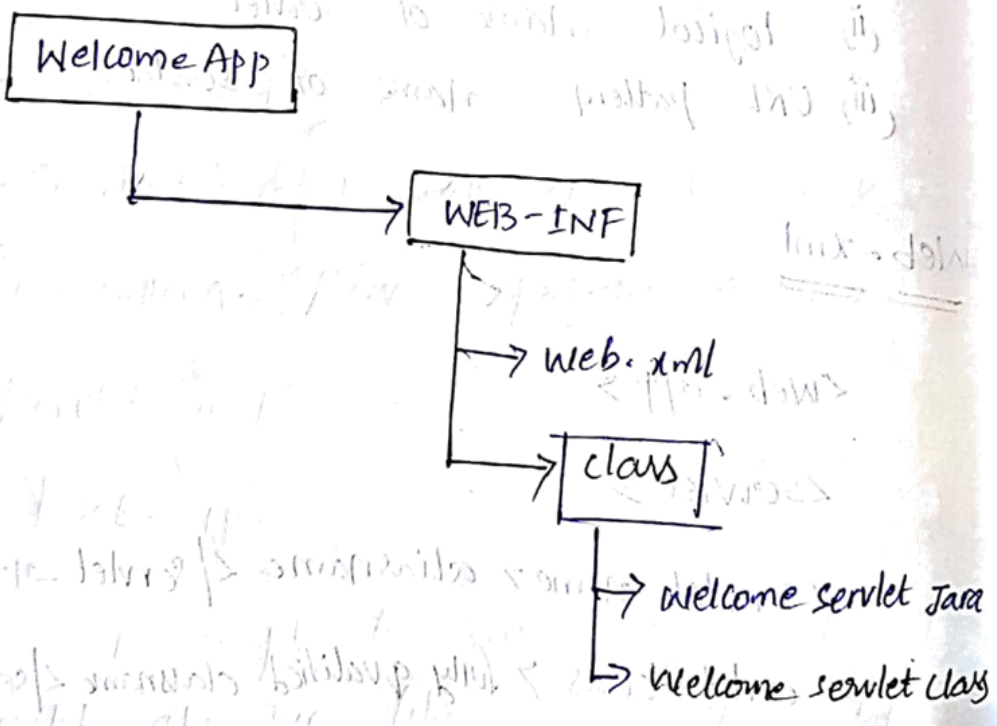
```
<web-app>
 <servlet>
 <servlet-name> aliasname </servlet-name>.
 <servlet-class> fully qualified classname </servlet-class>
```

Scanned by CamScanner

```
<servlet-mapping>
    <servlet-name> alias name </servlet-name>
    <servlet-class>
        url pattern name </servlet-class>
</servlet-mapping>
</web-app>
```

⇒ Step-by-step process to create a web application with a servlet to get some ~~wett~~-come welcome response.

Step 1:-
⇒ Create the following directory structure



Scanned by CamScanner

step2:- Create the welcome servlet .Java like following

```
//welcome Servlet.java

import javax.servlet.*;

import java.io.*;

public class welcomeServlet extends GenericServlet

{

public void service(ServletRequest req, ServletResponse res)
                throws ServletException, IOException
{
  PrintWriter pw= response.getWriter();
  pw.println("<h1> welcome to servlets </h1>");
  pw.close();
}

}
```

In the above code the extended our class from an abstract class Generic servlets. In these generic servlet class there is a one method called "Service method".

Scanned by CamScanner

Step 2:-

NOTE:- If we write the sop statement Then the o/p will displayed on Server Console but not on a web Browser.

Step 3:-

⇒ Configure welcome servlet in web.xml file

```
<!-- web.xml -->
<web-app>
<servlet>
<servlet-name> welcome </servlet-name>
<servlet-class> welcome servlet </servlet-class>
</servlet>
<servlet-mapping>
<servlet-name> welcome </servlet-name>
<url-pattern> /serv, </url-pattern>
</servlet-mapping>
</web-app>
```

Step 4:- To compile to servlet we need to set servlet-api-Jar file class path by

Scanned by CamScanner

Tom-cat Server.

D:/welcome App/ WEB-INF/classes> s a Thread for each

set classpath = "C:/programfiles/

Apache software foundation/ Tomcat 7.0/lib/ servlet.api.jar;

Javac welcomeServlet.Java ↵

Step5:- Deploy the web app's in tom-cat Server,

to Deploy our web app's ~~web publication~~ in Tom-cat

Server, copy root directory of app into

C:/ programfiles/ Apache software foundation/ Tomcat 7.0/web

a web app's can be deployed in 3 ways

1) hard deployment
2) console deployment
3) Tool based deployment

Step6:- Start the Tom-cat Server.

To start ⇒ Goto program files /Apache s/w foundation/

Tomcat 7.0/ Bin folder and

double click - Tomcat 7.0.

Scanned by CamScanner

Step 7:- Open the browser and type the following URL. To send a request to servlet.

http://localhost:8080/WelcomeApp/srv ↵

Defination

Servlet:-

It is a simple java program That run on the server. It is an API. That provides many interfaces and classes including documentation.

Servlet is a web-component that is deployed in server to create dyamic web page.

Scanned by CamScanner

Adv:- ↳ Better performance

∴ Because it creates a Thread for each sequences process.

↳ protability

↳ Robaust

∴ Servlet are managed by JVM so, we don't need worry about memory, Garbage collection e.t.c

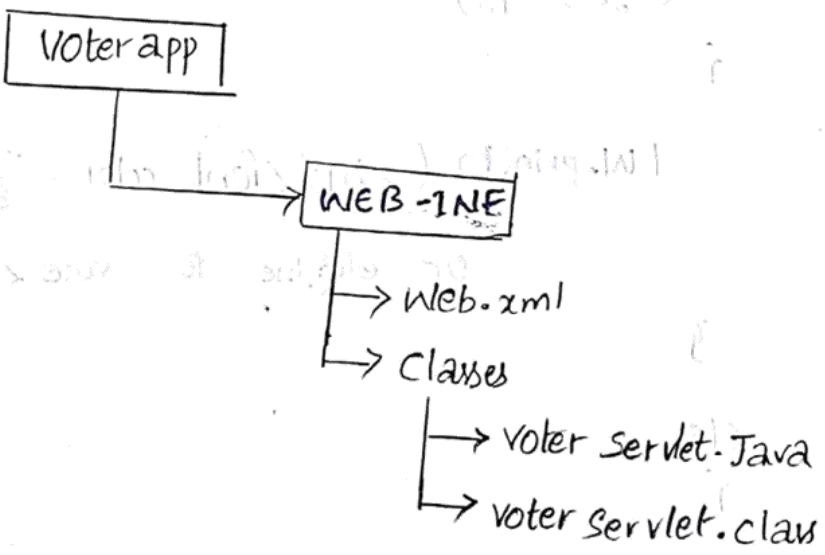
↳ security

Scanned by CamScanner

Date
19/2/18

The following web application contains a servlet which accepts age as a parameter and provides the response as Eligible (or) not Eligible to vote based on the age value?

Directory structure



Program

```
// voter Servelt.java
import java.io.*;
public import javax.servlet.*;
Public class voterservlet extends Generic Servlet
{
```

Scanned by CamScanner

```
public void Service (ServletRequest req, ServletResponse res)
        Throws servlet Exception, IOException
{
    PrintWriter pw = res.getwriter();
    String str = req.getparameter("age");
    int age = Integer.parseInt(str);
    if(age >=18)
    {
        pw.println("<h1><font color = "green">
                Ur eligible to vote </font></h1>")
    }
    else
    {
        pw.println("<h1> <font Color="red">
                Ur not eligible to vote </font></h1>")
    }
    pw.close();
}
}
```

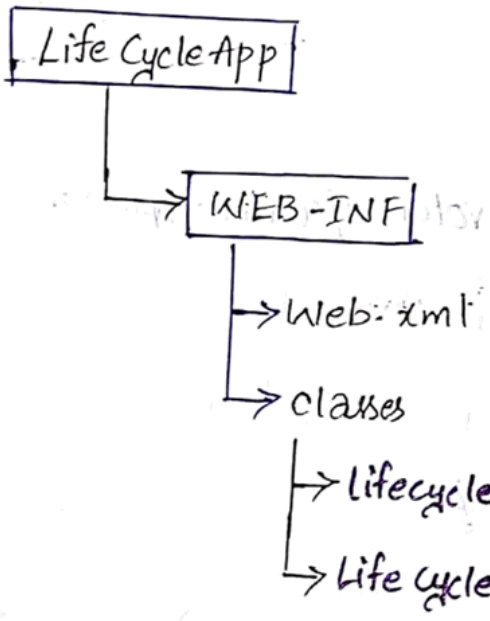
Scanned by CamScanner

```
<!-- web.xml -->

<web-app>
<servlet>

<servlet-name> voter </servlet-name>
<servlet-class> VoterServlet </servlet-class>
</servlet>
<servlet-mapping>
<servlet-name> voter </servlet-name>
<url-pattern> /vote
</url-pattern>
</servlet-mapping>
</web-app>
```

Scanned by CamScanner

The following web application contain servlet with all lifecycle method in it. This Example is To know The number of times a life cycle method is called by web container.

Directory structure:-



PROGRAM

```
// LifeCycleServlet.java
import java.io.*;
import javax.servlet.*;
Public class LifecycleServlet extends GenericServlet
{
```

Scanned by CamScanner

```
public LifecycleServlet()
{
  System.out.println("I am constructor");
}
public void init(ServletConfig Config)
                    Throws servletException
{
    System.out.println("Iam lifecycle init");
}
public void service(ServletRequest req, Servlet
        Response res)
            Throws servletException, IOException
{
  System.out.println("I am lifecycle service");
}
  public void destroy()
  {
    System.out.println("I am life cycle destroy");
  }
}
```

Scanned by CamScanner

```
<! -- web.xml -->

<web-app>

<servlet>

<servlet-name> life cycle </servlet-name>

<servlet-class> lifecycle servlet </servlet-class>
</servlet>

<servlet-mapping>
<servlet-name> lifecycle </servlet-name>
<url-pattern> /serv1 - </url-pattern>

</servlet-mapping>

</web-app>.
```

o/P ~~Prtout~~ ~~to~~

```
I am a constructor

I am a life cycle init

I am a lifecycle service

I am a lifecycle service
```

Scanned by CamScanner

Servlet Api :-

The javax.servlet & javax.servlet.http packages represent interfaces & classes for Servlet Api.

The javax.servlet package contains many interfaces & classes that are used by the Servlet or web Container. these are not specific to any protocol.

The javax.servlet.http package contains interfaces & classes that are responsible for http request only.

let's see the what are the interfaces of javax.servlet package.

javax.servlet package :-

this contains no.of interfaces & classes that establish the framework in which servlets operate. they are as follows,

interface	Description
Servlet	Declares life cycle methods for a servlet
ServletConfig	Allows servlets to obtaining get initialization parameters
ServletRequest	Used to read data from a client Request
ServletResponse	Used to write data to a client Response

class	Description
GenericServlet	implements the Servlet & ServletConfig interface
ServletInputStream	provides an inputStream for reading request from a client
ServletOutputStream	provides an outputstream for writing responses to a client
ServletException	indicates servlet error occurred.

Scanned by CamScanner

lets examine these interfaces & classes in more detail,

the Servlet interface

Void init() — called when the servlet is initialized

Void destroy() — called when the servlet is unloaded

void Service() — Called to process a request from a client

The ServletConfig Interface

getInitparameter() – Returns the value of the named servlet initialization parameter

getInitParameterNames() – Returns the names of all the servlet's initialization Parameter.

the ServletRequestInterface

getAttribute() — Returns the value, of the named attribute as an object

getAttributeNames() – Returns the names of the attributes

getParameter() – Returns value of the request parameter as a String

getParameterNames() – Returns array of the String objects containing the names.

the ServletResponse Interface

SetContentType() – Sets the Content type of the response being sent to client

SetContentLength() – Sets the length of the content body in the response

getWriter() — Returns a PrintWriter object that can send character text to the client

getOutputStream()

the GenericServlet class

It provides the implementation of all the methods of these interfaces except the Service method & provide implementation of Service.

① public void init(ServletConfig config) → initialize the servlet

② public abstract void service(ServletRequest request, ServletResponse response) → provides service for req, res

Scanned by CamScanner

③ public ServletConfig getServletConfig()

④ public void destroy()

⑤ public void init()

The ServletInputStream class:-

ServletInputStream class extends InputStream. It is implemented by Server & provides an inputStream to read the data from client req-

int readline(byte[] buffer, int offset, int size) throws IOException

↓

Here, buffer is the array into which size bytes are placed starting at offset. It returns the actual no. of bytes read.

The ServletOutputStream class

ServletOutputStream class extends outputStream. It is implemented by Server & provides outputStream that a Servlet developer can use to write data to a client respons.

print() → methods
println()

The ServletException class:-

Javax.Servlet defines two exceptions. The first is ServletException, which indicates that a Servlet problem.

Second is unavailableException which extends ServletException. It indicates that a Servlet is unavailable.

The javax.servlet.http package:-

It Contains a no. of interfaces & classes that are commonly used by Servlet developers. It is easy to build servlets that work with HTTP requests & responses.

Scanned by CamScanner

Interface	Description
HttpServletRequest	Enables Servlets to read data from an HTTP request
HttpServletResponse	Enables Servlet to write data to an HTTP response
HttpSession	Allows session data to be read & written

class	Description
Cookie	Allows state information to be stored on client machine
HttpServlet	provides methods to handle HTTP requests & responses

The HttpServletRequest Interface It is used to obtain the information from clients HTTP request

getCookies() - returns the information in the cookies in the request mode

getHeaders() - returns the value of the header fields

~~getRequestURI()~~

getpathInfo() - returns the path information about the Servletpath

getmethod() - returns the HTTP method for the client req

The HttpServlet Response Interface:-

It is used to formulate an HTTP response to the client

addCookie() - it is used to add cookie in the response

setHeader() - Set a response header with given name & value

encodeURL() - It is used to encode the specified URL

HttpSession Interface.

The Servlet can read (or) write this information using HttpSession information. It's implemented by Server

getID() - returns the session ID

getAttribute() - returns the value of attribute

getAttributenames() - returns the attribute names

Scanned by CamScanner

Cookie class:-

A cookie is a small piece of information that is stored in the client's machine.

~~class~~

getValue() — returns a value of the cookie

SetValue() — sets the value to the cookie

getName() — returns the cookie name

HttpServlet class

The HttpServlet class extends GenericServlet. It is used when developing servlets that receive & process HTTP requests.

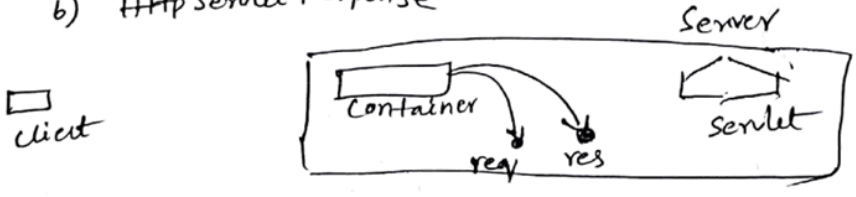
void doGet(HttpServletRequest req, HttpServletResponse res) — this method performs HTTP get request

void doPost(HttpServletRequest req, HttpServletResponse res) — this method performs HTTP post request

void Service(HttpServletRequest req, HttpServletResponse res) — this method is invoked for processing HTTP req & res

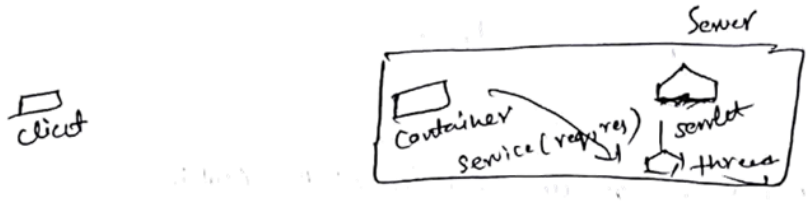
How Servlet Application works,

1. user sends request for a servlet by clicking URL.
2. the Container finds the servlet using deployment descriptor & creates two objects.
   a) HttpServletRequest
   b) HttpServletResponse



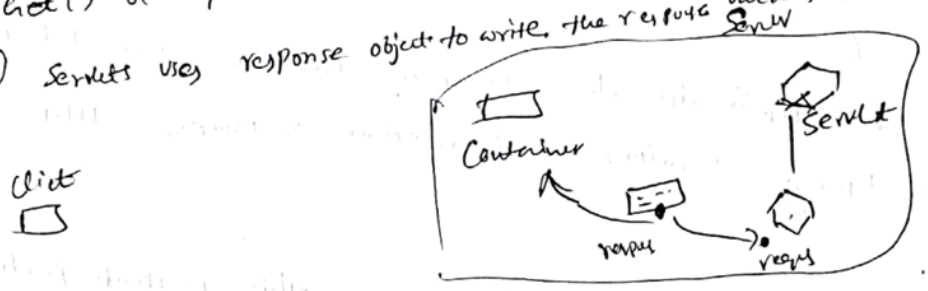
Scanned by CamScanner

③ Then Container creates a thread for that req & calls Service() method & passes the req, res objects as arguments

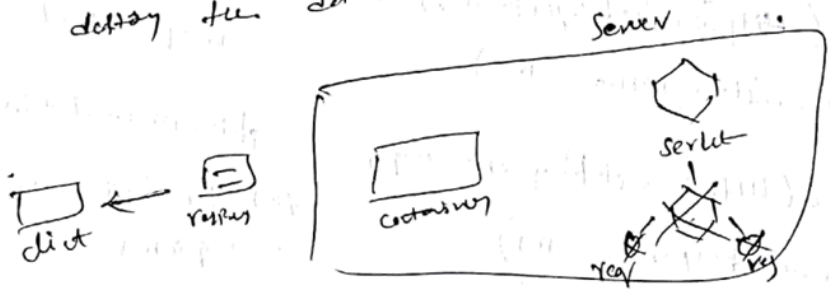


④ The Service() method, then decides which servlet method, doGet() or doPost() to call based on HTTP Request Method (Get, Post, etc)

⑤ Servlets uses response object to write the response back to the client



⑥ After the Service() method is complete the thread dies destroy the data



Scanned by CamScanner

Unit III

Reading Initialization parameters

A Servlet may have some values needed for it's execution or a Programmer may require some values to be fed to Servlet needed in the coding for execution. These values are specific to a particular Servlet and not required for all Servlets. For this, the deployment descriptor **web.xml**, comes with **<init-param>** tag and an example is given to read all <init-param> values and also a single one.

To read the <init-param> values, known as **initialization parameters**, we use **ServletConfig**interface from **javax.servlet** package.

```
<init-param>

  <param-name>portnumber</param-name>

  <param-value>8888</param-value>

</init-param>
```

The above code is an example entry in **web.xml** file. To read **8888** of **<param-value>**, it is used **portnumber** of **<param-name>**. **portnumber** and **8888** are used as **key/value** pairs in Servlet coding.

ServletConfig Example using getInitParameter() and getInitParameterNames() methods:

Client file to invoke Servlet: InitParam.html

```
<body>

Would you like read <a href="http://localhost:8888/india/bharat">initialization parameters </a>from web.xml file sir?

</body>
```

The above HTML code includes just an hyper link to invoke **ReadInitParamValues** Servlet with alias name **bharat**.

Scanned by CamScanner

Following is the web.xml entry for the Servlet:

```
<servlet>
    <servlet-name>hello</servlet-name>
    <servlet-class>ReadInitParamValues</servlet-class>
 <init-param>
    <param-name>pearlcity</param-name>
   <param-value>Hyderabad</param-value>
 </init-param>
 <init-param>
     <param-name>monument</param-name>
     <param-value>Charminar</param-value>
 </init-param>
 <init-param>
    <param-name>mangoCost</param-name>
    <param-value>250.5</param-value>
 </init-param>
 <init-param>
    <param-name>numberOfMangoes</param-name>
    <param-value>30</param-value>
 </init-param>
</servlet>
<servlet-mapping>
   <servlet-name>hello</servlet-name>
   <url-pattern>/bharat</url-pattern>
</servlet-mapping>
```

Scanned by CamScanner

web.xml file can have any number of <init-param> tags.

Servlet file: ReadInitParamValues.java (of alias name bharat):

```
import javax.servlet.ServletException;

import javax.servlet.ServletConfig;

import javax.servlet.http.*;

import java.io.*;

import java.util.Enumeration;

public class ReadInitParamValues extends HttpServlet

{

  public void service( HttpServletRequest req, HttpServletResponse res ) throws ServletException,
IOException

 {

 res.setContentType("text/html");

 PrintWriter pw = res.getWriter();

        // obtain an object of ServletConfig interface

 ServletConfig config = getServletConfig();

        // to read all values

 pw.println("Reading all values:" + "<br>");

 Enumeration e = config.getInitParameterNames();

 while(e.hasMoreElements())

 {

  String name = (String) e.nextElement();  // returns the <param-name>

  String value = config.getInitParameter(name);  // returns <param-value>

  pw.println("<br>" + name + " : " + value );

 }
```

the action of meaning a next thing, one by one

Scanned by CamScanner

```
        // to read one value (not all) and use in code
  String str1 = config.getInitParameter("mangoCost");
  String str2 = config.getInitParameter("numberOfMangoes");
        // parsing code
  double mc = Double.parseDouble(str1);
  int nom = Integer.parseInt(str2);
  pw.println("<hr>Reading one value:" + "<br><br>");
  pw.println("One mango cost: " + mc/nom);
        // to use getServletConfig() method
  pw.println("<hr>Servlet Name: " + config.getServletName());
        // prints <servlet-name> of web.xml file. In this case hello
  pw.close();
 }
}
```

Explanation for the above code:

Following is the method of **GenericServlet** class used in the servlet code.

```
public javax.servlet.ServletConfig getServletConfig();
```

Following are the methods of **ServletConfig** interface used in the servlet code.

```
public abstract java.lang.String getServletName();
public abstract java.lang.String getInitParameter(java.lang.String);
public abstract java.util.Enumeration getInitParameterNames();
```

Scanned by CamScanner

ServletConfig config = getServletConfig();

ServletConfig interface from **javax.servlet** package is used to read the initialization parameters of web.xml. Its object is returned by the **getServletConfig()** method of **HttpServlet**(inherited from **GenericServlet**). **config** is an object of **ServletConfig**.

Enumeration e = config.getInitParameterNames();

getInitParameterNames() method of **ServletConfig** interface returns an object of Enumeration. The object **e** of **Enumeration** contains the values of all <param-name> tags but not of <param-value> tags. The values of <param-value> tags are be obtained from <param-name> tags. This is what is done in the following loop.

```
while(e.hasMoreElements())
{
  String name = (String) e.nextElement();  // returns the <param-name>
  String value = config.getInitParameter(name);  // returns <param-value>
  pw.println("<br>" + name + " : " + value );
}
```

It is the same Enumeration interface we used to iterate collection classes. The **nextElement()**method of Enumeration returns an object of **Object** class and is type casted to String. The name variable contains the <param-name> value. The name variable is passed to **getInitParameter(name)** to get the value of its corresponding <param-value> value.

getInitParameterNames() returns all values and **getInitParameter("mangoCost")** returns only one value of corresponding <param-value> value. Observe the web.xml entry given earlier.

String str1 = config.getInitParameter("mangoCost");
String str2 = config.getInitParameter("numberOfMangoes");

getInitParameter(String) returns String object and is required to parse to use in arithmetic operations.

double mc = Double.parseDouble(str1);

Scanned by CamScanner

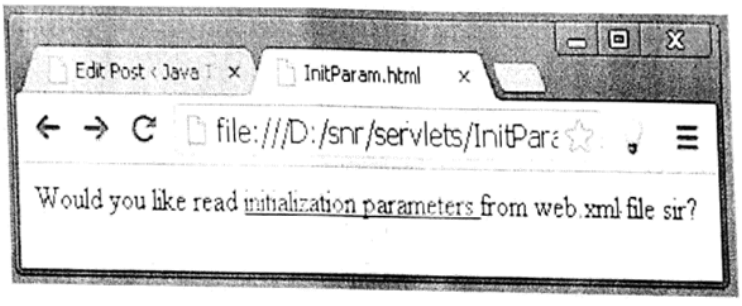
```
int nom = Integer.parseInt(str2);
```

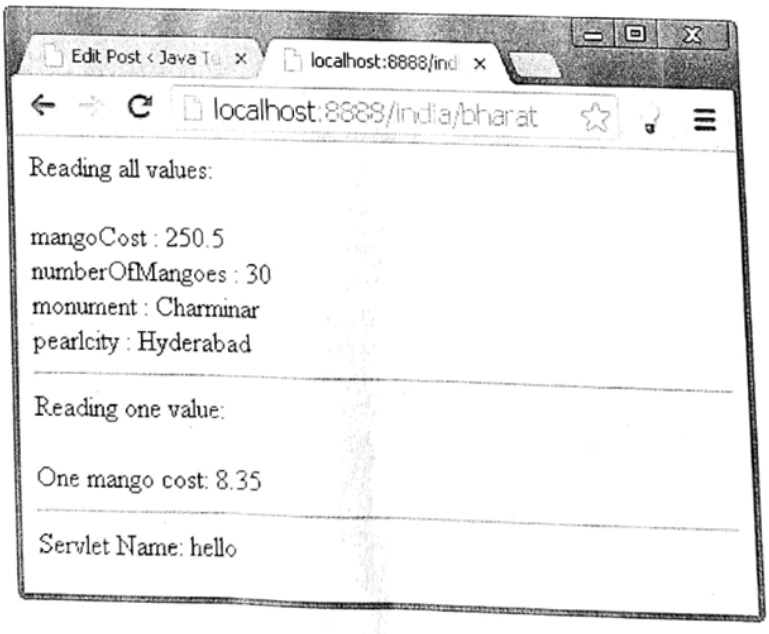
str1 representing **mangoCost** is parsed to double and
str2 representing **numberOfMangoes** is parsed to int.

```
pw.println("<hr>Servlet Name: " + config.getServletName( ));
```

The <servlet-name> value of web.xml can be obtained
with **getServletName()** of **ServletConfig**.

The output screen :





Scanned by CamScanner

Reading Servlet Parameters

The ServletRequest class includes methods that allow you to read the names and values of parameters that are included in a client request. We will develop a servlet that illustrates their use.

The example contains two files.

A Web page is defined in **sum.html** and a servlet is defined in **Add.java**

sum.html:

```
<html>
<body>
<center>
<form name="Form1"  method="post"  action="Add">
<table>
<tr>
<td><B>Enter First Number</td>
<td><input type=textbox name="Enter First Number" size="25" value=""></td>
</tr>
<tr>
<td><B>Enter Second Number</td>
<td><input type=textbox name="Enter Second Number" size="25" value=""></td>
</tr>
</table>
<input type=submit value="Submit">
</body>
</html>
```

Scanned by CamScanner

The HTML source code for sum.html defines a table that contains two labels and two text fields. One of the labels is Enter First Number,and the other is Enter Second Number. There is also a submit button. Notice that the action parameter of the form tag specifies a URL. The URL identifies the servlet to process the HTTP POST request.

Add.java :

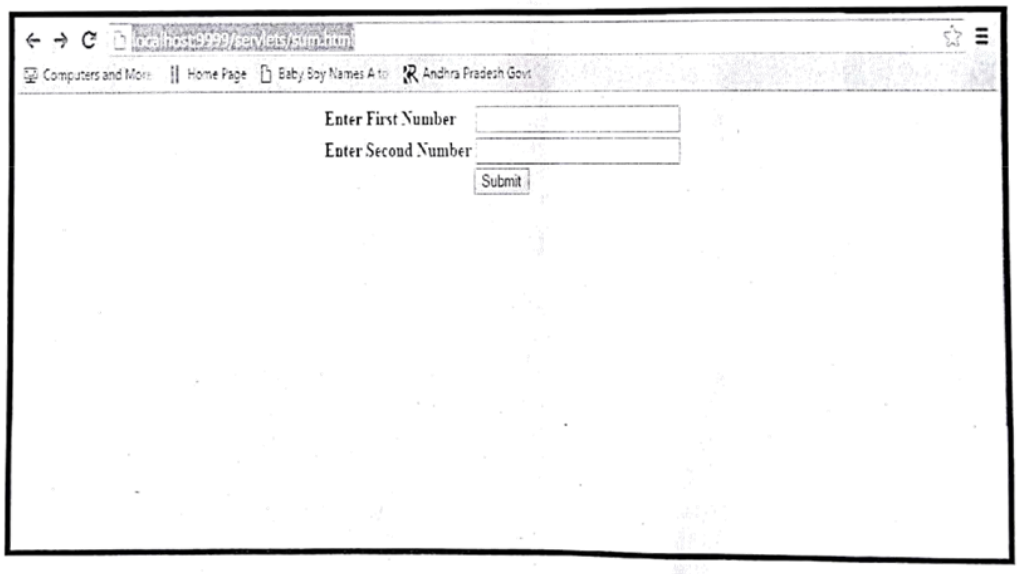
```
import java.io.*;

import java.util.*;

import javax.servlet.*;

import javax.servlet.http.*;

public class Add extends HttpServlet

{

public void doPost(HttpServletRequest request,HttpServletResponse response) throws
ServletException, IOException

{

// Get print writer.

response.getContentType("text/html");

PrintWriter pw = response.getWriter();

// Get enumeration of parameter names.

Enumeration e = request.getParameterNames();

// Display parameter names and values.

int sum=0;

while(e.hasMoreElements())

{

String pname = (String)e.nextElement();

pw.print(pname + " = ");

String pvalue = request.getParameter(pname);
```

Scanned by CamScanner

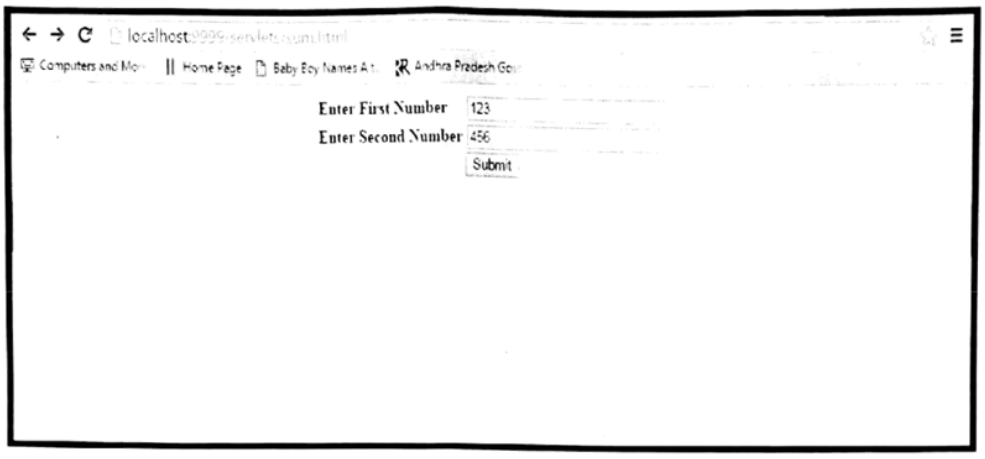
```
sum+=Integer.parseInt(pvalue);
pw.println(pvalue);
}
pw.println("Sum = "+sum);
pw.close();
}
}
```

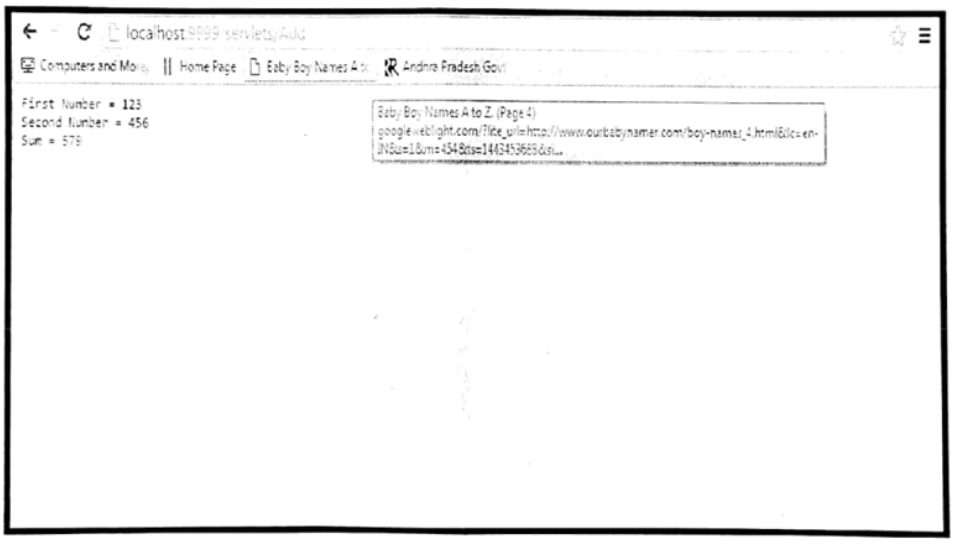
The source code for Add.java contains doPost( ) method is overridden to process client requests. The getParameterNames( ) method returns an enumeration of the parameter names. These are processed in a loop.we can see that the parameter name and value are output to the client. The parameter value is obtained via the getParameter( ) method.

after typing URL : http://localhost:9999/servlets/sum.html



Scanned by CamScanner





Scanned by CamScanner

Handling Http Request & Responses

The HttpServlet class provides specialized methods that handle the various types of HTTP requests. A servlet developer typically overrides one of these methods. These methods are doDelete( ), doGet( ), doHead( ), doOptions( ), doPost( ), doPut( ), and doTrace( ). The GET and POST requests are commonly used when handling form input.

Handling HTTP GET Requests:

The following programs are for servlets that handles HTTP GET request. The servlet is invoked when a form on a Web page is submitted.

The example contains two files. A Web page is defined in Get.html and a servlet is defined in GetServlet.java. The HTML source code for Get.html is shown in the following listing. It defines a form that contains a select element and a submit button. Notice that the action parameter of the form tag specifies a URL. The URL identifies a servlet to process the HTTP GET request.

Get.html:

```
<html>
<body>
<center>
<form name="Form1" action="http://localhost:9999/servlet/GS">
<B>Color:</B>
<select name="color" size="1">
<option value="Red">Red</option>
<option value="Green">Green</option>
<option value="Blue">Blue</option>
```

GetServlet

Scanned by CamScanner

```
</select>
<br><br>
<input type=submit value="Submit">
</form>
</body>
</html>
```

The source code for GetServlet.java is shown in the following listing. The doGet( ) method is overridden to process any HTTP GET requests that are sent to this servlet. It uses the getParameter( ) method of HttpServletRequest to obtain the selection that was made by the user. A response is then formulated.

GetServlet.java:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class GetServlet extends HttpServlet
{
public void doGet(HttpServletRequest request,HttpServletResponse response)
throws ServletException, IOException
{
String color = request.getParameter("color");
response.setContentType("text/html");
PrintWriter pw = response.getWriter();
pw.println("<B>The selected color is: ");
pw.println(color);
pw.close();
```

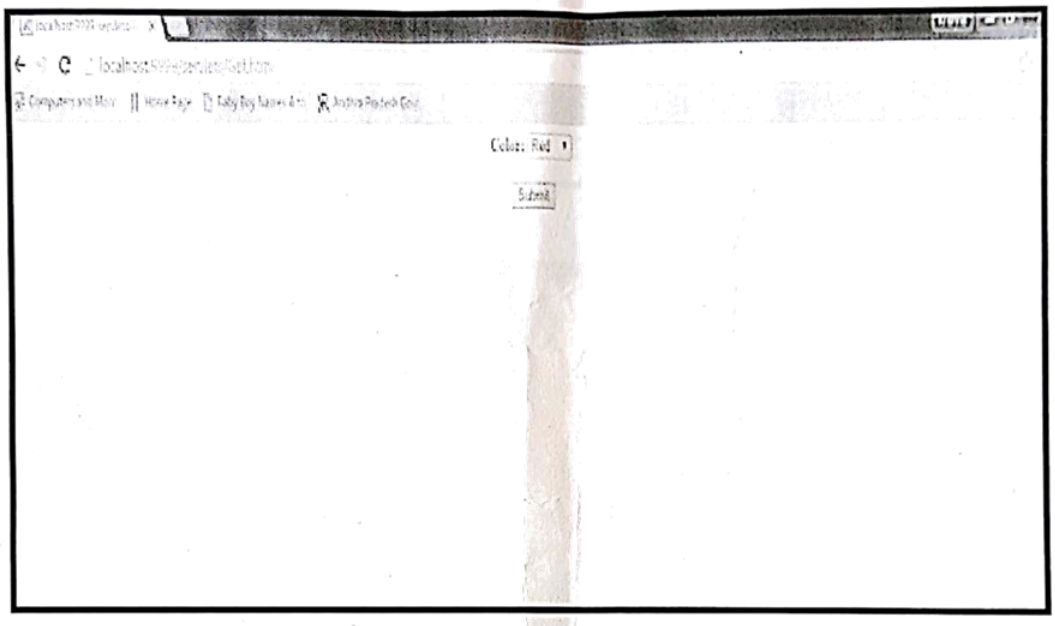
Scanned by CamScanner

```
}
}
```

Compile the servlet and perform these steps to test this example:

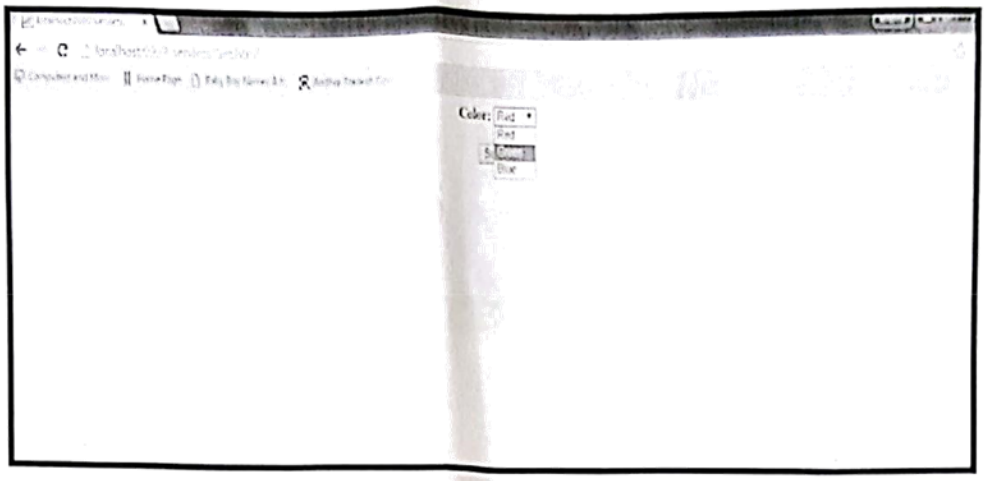
1. Start Tomcat, if it is not already running.

2. Display the Web page in a browser.

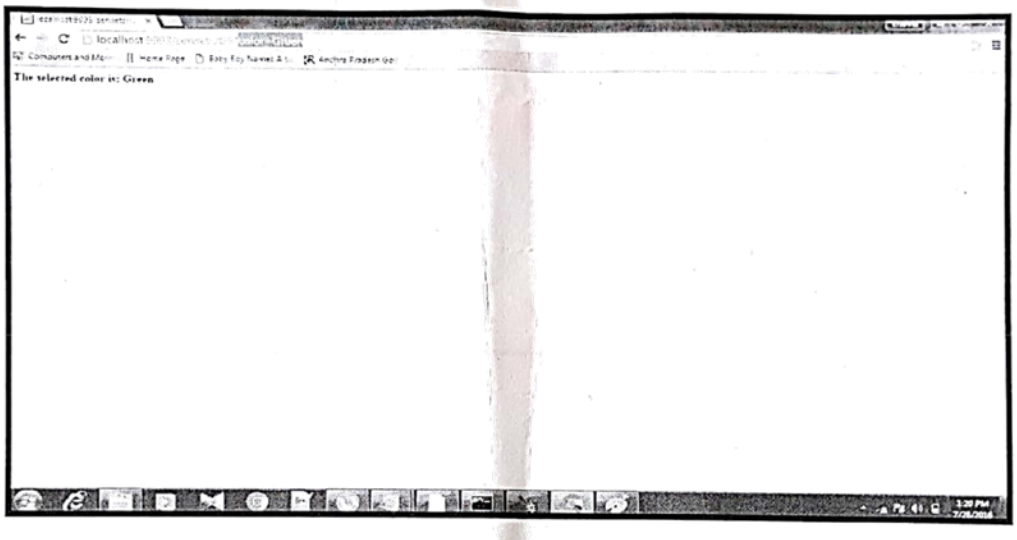


3. Select a color.

Scanned by CamScanner



4. Submit the Web page.



After completing these steps, the browser will display the response that is dynamically generated by the servlet.One other point: Parameters for an HTTP GET request are included as part of the URL that is sent to the Web server. Assume that the user selects the green option and submits the form. The URL sent from the browser to the server is
http://localhost:9999/servlet/GS?color=Green

The characters to the right of the question mark are known as the query string.

Scanned by CamScanner

Handling HTTP POST Requests:

The servlet is invoked when a form on a Web page is submitted. The example contains two files. A Web page is defined in Post.html and a servlet is defined in PostServlet.java.The HTML source code for Post.html is shown in the following listing. It isidentical toPost.html except that the method parameter for the form tag explicitly specifies that the POST method should be used, and the action parameter for the form tag specifies a different servlet.

Post.html:

```
<html>
<body>
<center>
<form name="Form1"  method="post"   action="http://localhost:9999/servlet/PS">
<B>Color:</B>
<select name="color" size="1">
<option value="Red">Red</option>
<option value="Green">Green</option>
<option value="Blue">Blue</option>
</select>
<br><br>
<input type=submit value="Submit">
</form>
</body>
</html>
```

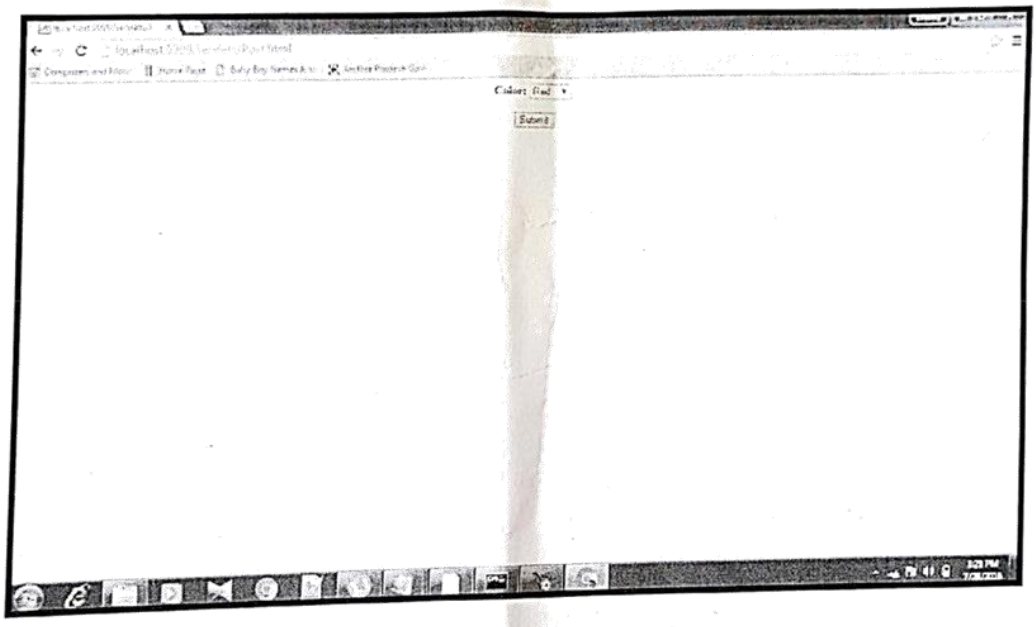
PostServlet.java:

```
import java.io.*;

import javax.servlet.*;

import javax.servlet.http.*;

public class ColorPostServlet extends HttpServlet

{

public void doPost(IHttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException

{
```

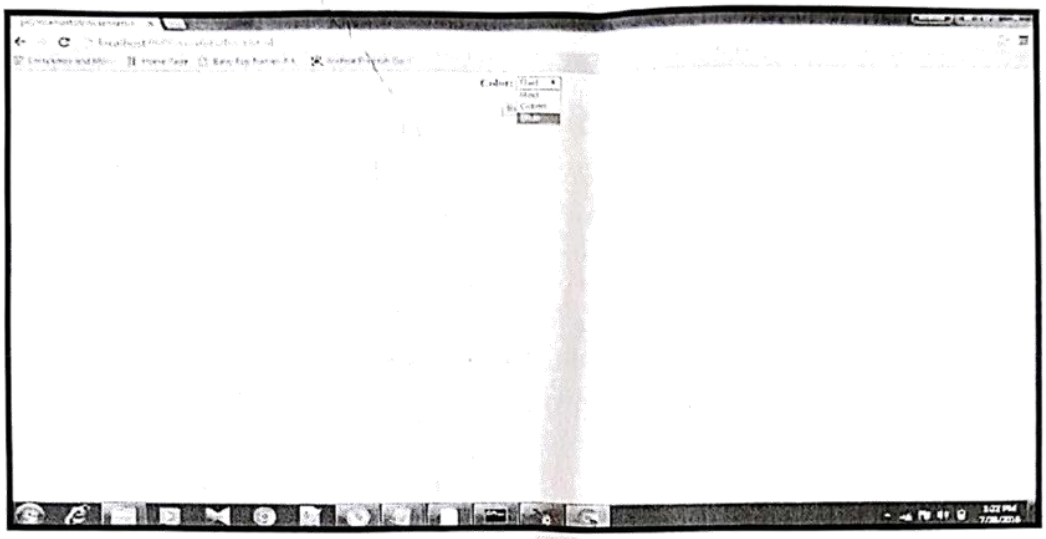
Scanned by CamScanner

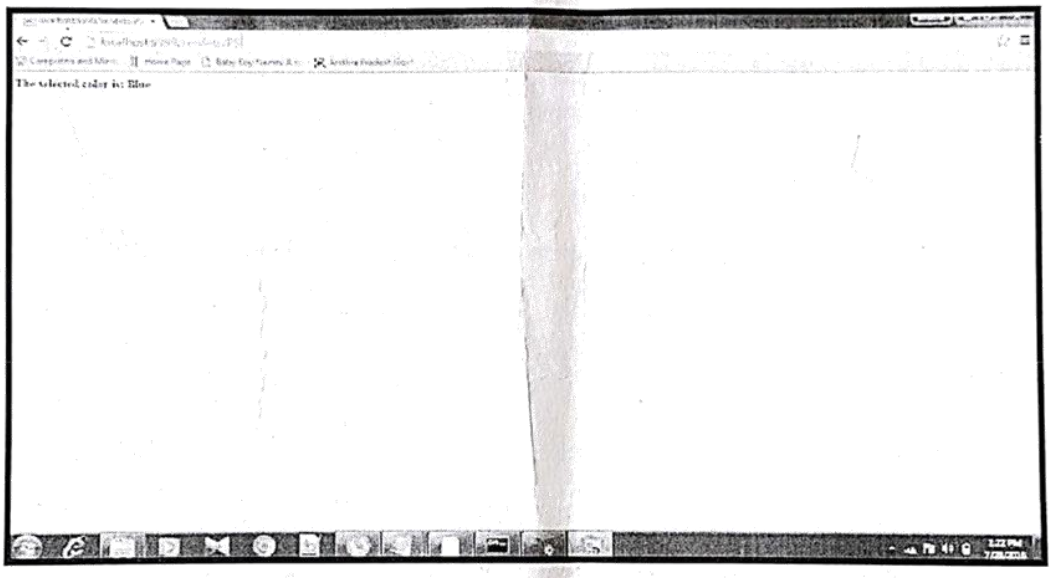
```
String color = request.getParameter("color");
response.setContentType("text/html");
PrintWriter pw = response.getWriter();
pw.println("<B>The selected color is: ");
pw.println(color);
pw.close();
}
}
```

Compile the servlet and perform the same steps as described in the previous section to test it.



Scanned by CamScanner





Note: Parameters for an HTTP POST request are not included as part of the URL that is sent to the Web server. In this example, the URL sent from the browser to the server is **http://localhost:9999/servlet/PS**

The parameter names and values are sent in the body of the HTTP request.

Scanned by CamScanner

Using Cookies

Cookies are usually small text files, given ID tags that are stored on your computer's browser directory or program data subfolders. Cookies are created when you use your browser to visit a website that uses cookies to keep track of your movements within the site, help you resume where you left off, remember your registered login, theme selection, preferences, and other customization functions.The website stores a corresponding file(with same ID tag)to the one they set in your browser and in this file they can track and keep information on your movements within the site and any information you may have voluntarily given while visiting the website, such as email address.

There are two types of cookies: session cookies and persistent cookies. Session cookies are created temporarily in your browser's subfolder while you are visiting a website. Once you leave the site, the session cookie is deleted. On the other hand, persistent cookie files remain in your browser's subfolder and are activated again once you visit the website that created that particular cookie. A persistent cookie remains in the browser's subfolder for the duration period set within the cookie's file.

Now, let's develop a servlet that illustrates how to use cookies. The servlet is invoked when a form on a Web page is submitted. The example contains three files as summarized here

File	Description
AddCookie.html	Allows a user to specify a value for the cookie named MyCookie.
AddCookieServlet.java	Processes the submission of AddCookie.html.
GetCookiesServlet.java	Displays cookie values.

The HTML source code for AddCookie.html is shown in the following listing. This page contains a text field in which a value can be entered. There is also a submit button on the page. When this button is pressed, the value in the text field is sent to AddCookieServlet via an HTTP POST request.

Scanned by CamScanner

AddCookie.html:

```
<html>
<body>
<center>
<form name="Form1"  method="post"
        action="http://localhost:8080/examples/servlet/AddCookieServlet">
<B>Enter a value for MyCookie:</B>
<input type=textbox name="data" size=25 value="">
<input type=submit value="Submit">
</form>
</body>
</html>
```

The source code for AddCookieServlet.java is shown in the following listing. It gets the value of the parameter named "data". It then creates a Cookie object that has the name "MyCookie" and contains the value of the "data" parameter. The cookie is then added to the header of the HTTP response via the addCookie( ) method. A feedback message is then written to the browser.

AddCookieServlet.java:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class AddCookieServlet extends HttpServlet
{
public void doPost(HttpServletRequest request,HttpServletResponse response)
throws ServletException, IOException
```

Scanned by CamScanner

```
{
// Get parameter from HTTP request.
String data = request.getParameter("data");
// Create cookie.
Cookie cookie = new Cookie("MyCookie", data);
// Add cookie to HTTP response.
response.addCookie(cookie);
// Write output to browser.
response.setContentType("text/html");
PrintWriter pw = response.getWriter();
pw.println("<B>MyCookie has been set to");
pw.println(data);
pw.close();
}
}
```

The source code for GetCookiesServlet.java is shown in the following listing. It invokes the getCookies( ) method to read any cookies that are included in the HTTP GET request. The names and values of these cookies are then written to the HTTP response. Observe that the getName( ) and getValue( ) methods are called to obtain this information.

GetCookiesServlet.java:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class GetCookiesServlet extends HttpServlet
{
```

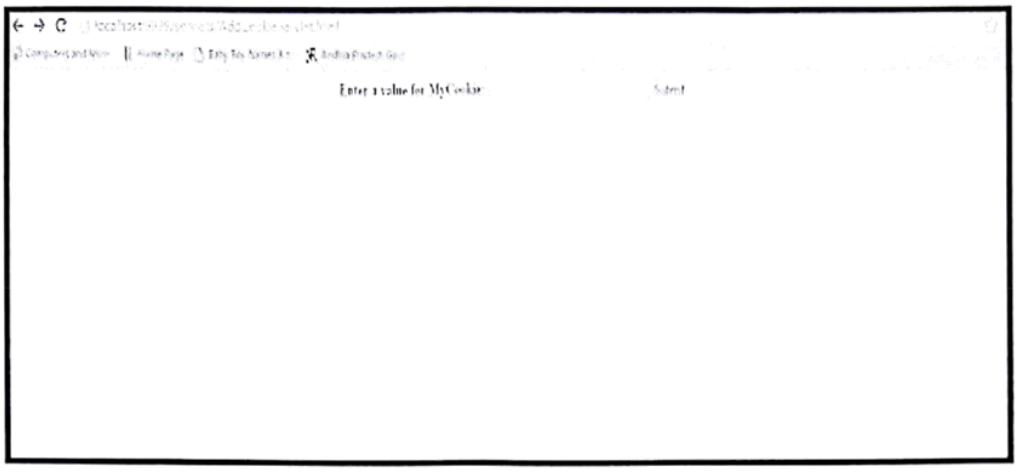
Scanned by CamScanner

```
public void doGet(HttpServletRequest request,HttpServletResponse response)
throws ServletException, IOException
{
// Get cookies from header of HTTP request.
Cookie[] cookies = request.getCookies();
// Display these cookies.
response.setContentType("text/html");
PrintWriter pw = response.getWriter();
pw.println("<B>");
for(int i = 0; i < cookies.length; i++)
 {
String name = cookies[i].getName();
String value = cookies[i].getValue();
pw.println("name = " + name +
"; value = " + value);
}
pw.close();
}
}
```

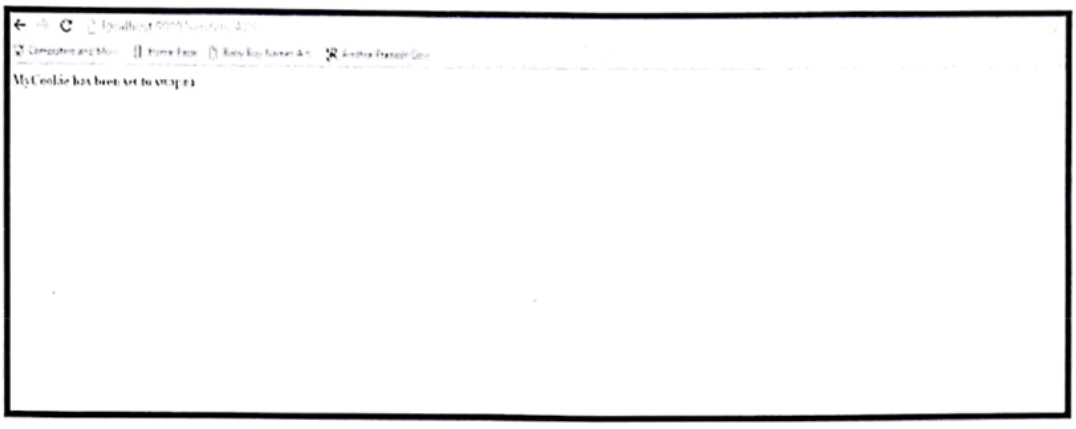
Compile the servlet and perform these steps:

1. Start Tomcat, if it is not already running.
2. Display AddCookie.htm in a browser.
3. Enter a value for MyCookie.
4. Submit the Web page.

Scanned by CamScanner



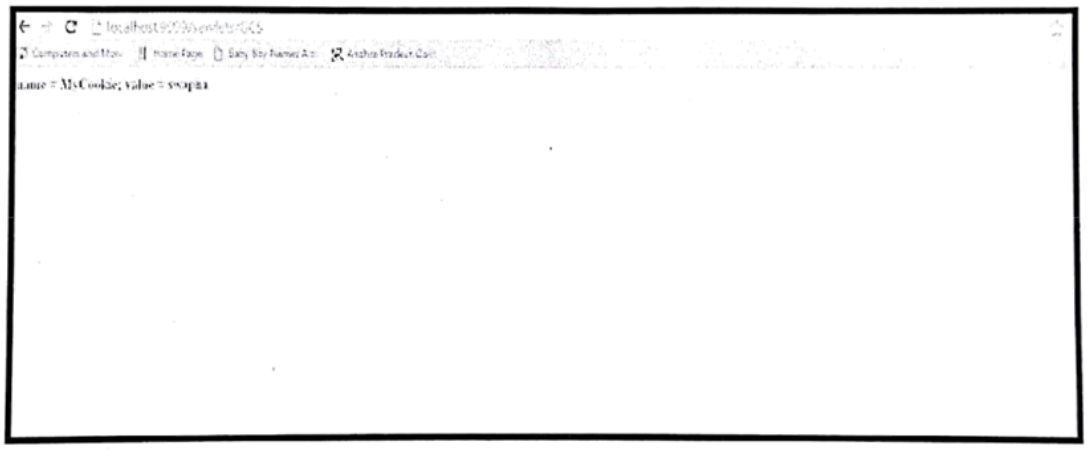
Scanned by CamScanner



After completing these steps you will observe that a feedback message is displayed by the browser.

Next, request the following URL via the browser:

http://localhost:9999/servlets/GCS



Observe that the name and value of the cookie are displayed in the browser. In this example, an expiration date is not explicitly assigned to the cookie via the setMaxAge( ) method of Cookie. Therefore, the cookie expires when the browser session ends. You can experiment by using setMaxAge( ) and observe that the cookie is then saved to the disk on the client machine.

Scanned by CamScanner

Session Tracking

HTTP is a stateless protocol. Each request is independent of the previous one. However,in some applications, it is necessary to save state information so that information can be collected from several interactions between a browser and a server. Sessions provide such a mechanism.

A session can be created via the getSession( ) method of HttpServletRequest. An HttpSession object is returned. This object can store a set of bindings that associate names with objects. The setAttribute( ), getAttribute( ), getAttributeNames( ), and removeAttribute( ) methods of HttpSession manage these bindings. It is important to note that session state is shared among all the servlets that are associated with a particular client.

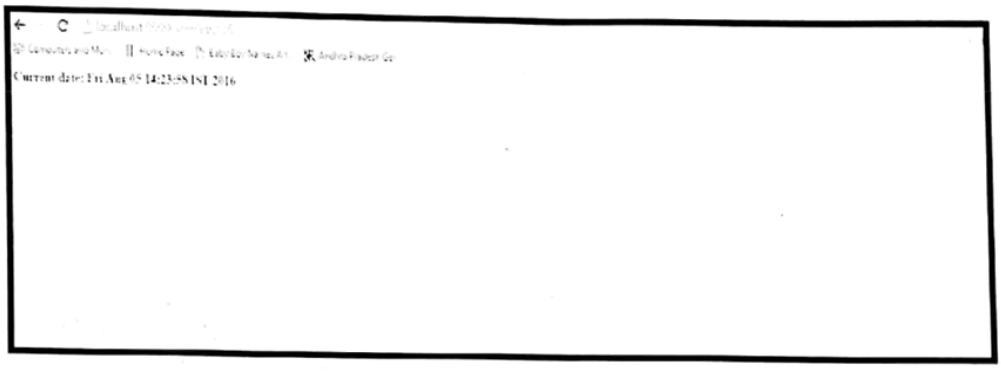
The following servlet illustrates how to use session state. The getSession( ) method gets the current session. A new session is created if one does not already exist. The getAttribute( ) method is called to obtain the object that is bound to the name "date". That object is a Date object that encapsulates the date and time when this page was last accessed. (Of course, there is no such binding when the page is first accessed.) A Date object encapsulating the current date and time is then created. The setAttribute( ) method is called to bind the name "date" to this object.

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class DateServlet extends HttpServlet
{
public void doGet(HttpServletRequest request,HttpServletResponse response)
throws ServletException, IOException
{

// Get the HttpSession object.
HttpSession hs = request.getSession(true);

// Get writer.
response.setContentType("text/html");
PrintWriter pw = response.getWriter();
pw.print("<B>");

// Display date/time of last access.
Date date = (Date)hs.getAttribute("date");

if(date != null)
{
pw.print("Last access: " + date + "<br>");
}
```

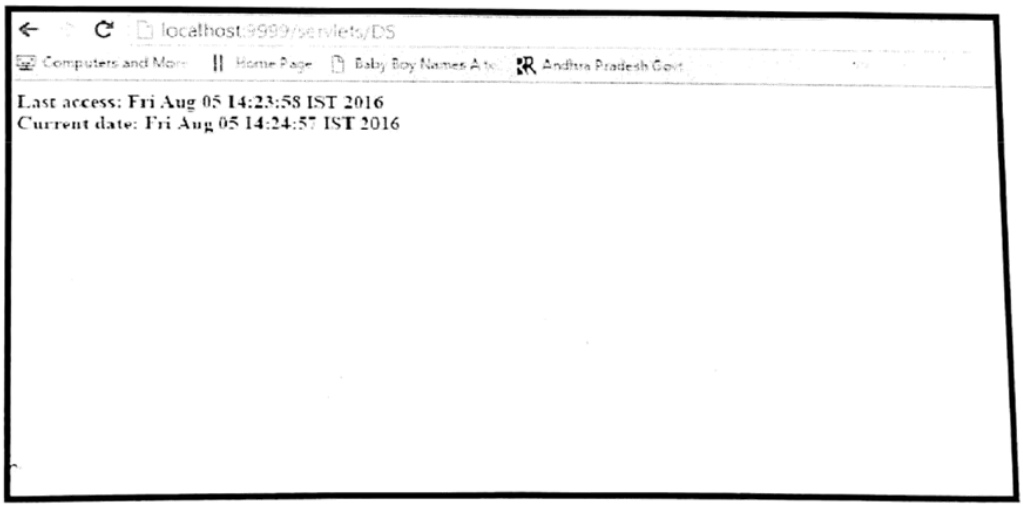
Scanned by CamScanner

```
// Display current date/time.
date = new Date();
hs.setAttribute("date", date);
pw.println("Current date: " + date);
}
}
```

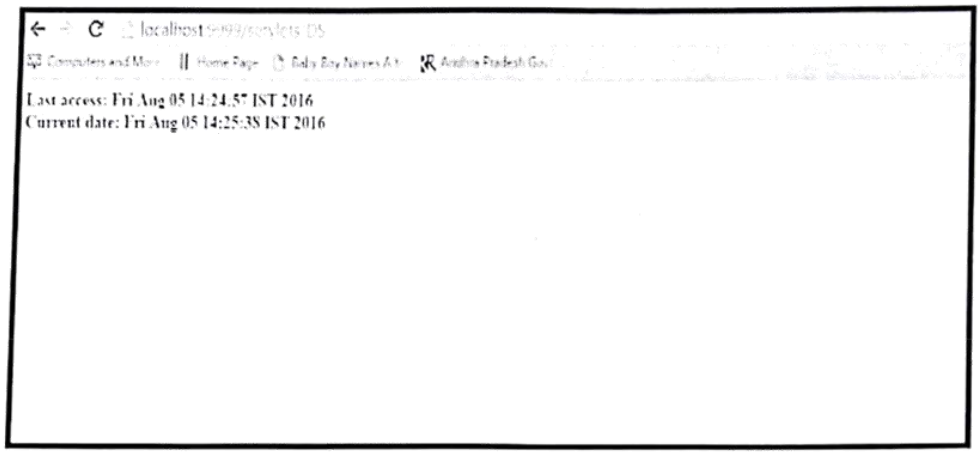
When you first request this servlet, the browser displays one line with the current date and time information.



On subsequent invocations, two lines are displayed. The first line shows the date and time when the servlet was last accessed. The second line shows the current date and time.



Scanned by CamScanner



Servlet Chaining | Servlet Request Despatcher

If a client request is processed by group of servlets, then that servlets are known as servlet chaining or if the group of servlets process a single client request then those servlets are known as servlet chaining.

In order to process a client request by many number of servlets then we have two models, they are forward model and include model.

Forward model:

In this model when we forward a request to a group of servlets, finally we get the result of destination servlet as a response but not the result of intermediate servlets.

Include model:

If a single client request is passed to a servlet and that servlet makes use of other group of servlets to process a request by including the group of servlets into a single servlet.

Note: One servlet can include any number of servlets where as one servlet can forward to only one servlet at a time.

FirstServlet.java

Scanned by CamScanner

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class FirstServlet extends HttpServlet
{
  public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException {
    res.setContentType("text/html");
    PrintWriter pw = res.getWriter();
    pw.println("<h2>I AM FROM FirstServlet BEGINNING</h2>");
    ServletContext ctx = getServletContext();
    RequestDispatcher rd = ctx.getRequestDispatcher("/SS");
    rd.include(req, res);
    pw.println("<h2>I AM FROM FirstServlet ENDING</h2>");
  }
}
```

SecondServlet.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class SecondServlet extends HttpServlet
{
```

Scanned by CamScanner

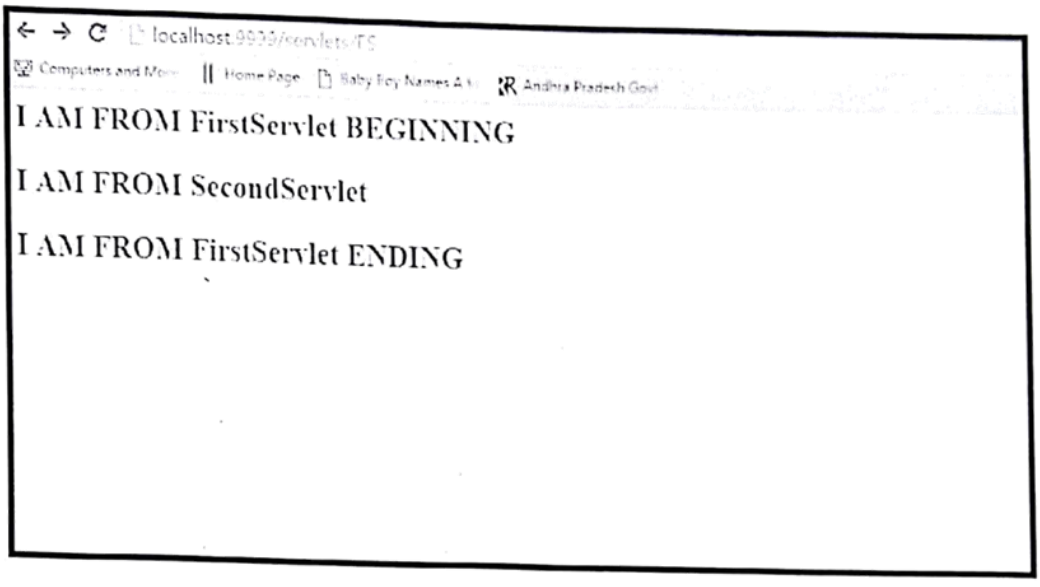
```
  public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException
{
      res.setContentType("text/html");
      PrintWriter pw = res.getWriter();
      pw.println("<h2>I AM FROM SecondServlet</h2>");
   }
}
```

Update web.xml with :

```
<servlet>
    <servlet-name>FS</servlet-name>
    <servlet-class>FirstServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>FS</servlet-name>
    <url-pattern>/FS</url-pattern>
  </servlet-mapping>
<servlet>
   <servlet-name>SS</servlet-name>
   <servlet-class>SecondServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>SS</servlet-name>
    <url-pattern>/SS</url-pattern>
  </servlet-mapping>
```

Scanned by CamScanner

Output:



1) **Write a program to store the user information into Cookies. Write another program to display the above stored information by retrieving from Cookies.**

Program:

AddCookieServlet.html:

```
<html>
<body>
<center>
<form name="Form1" method="post" action="ACS">
<B>Enter a value for MyCookie:</B>
<input type=textbox name = "d1" size=25 value="">
<input type=textbox name = "d2" size=25 value="">
```

Scanned by CamScanner

```
<input type=textbox name = "d3" size=25 value="">
<input type=submit value = "Submit">
</form>
</body>
</html>
```

AddCookieServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class AddCookieServlet extends HttpServlet
{
public void doPost(HttpServletRequest request,HttpServletResponse response)
throws ServletException, IOException
{
// Get parameter from HTTP request.
String a = request.getParameter("d1");
String b = request.getParameter("d2");
String c = request.getParameter("d3");

// Create cookie.
Cookie cookie1 = new Cookie("MyCookie1", a);
Cookie cookie2 = new Cookie("MyCookie2", b);
Cookie cookie3 = new Cookie("MyCookie3", c);
// Add cookie to HTTP response.
```

Scanned by CamScanner

```
response.addCookie(cookie1);
response.addCookie(cookie2);
response.addCookie(cookie3);
// Write output to browser.
response.setContentType("text/html");
PrintWriter pw = response.getWriter();
pw.println("<B>MyCookie has been set to");
pw.println(a+"  "+b+"  "+c);
pw.close();
}
}
```

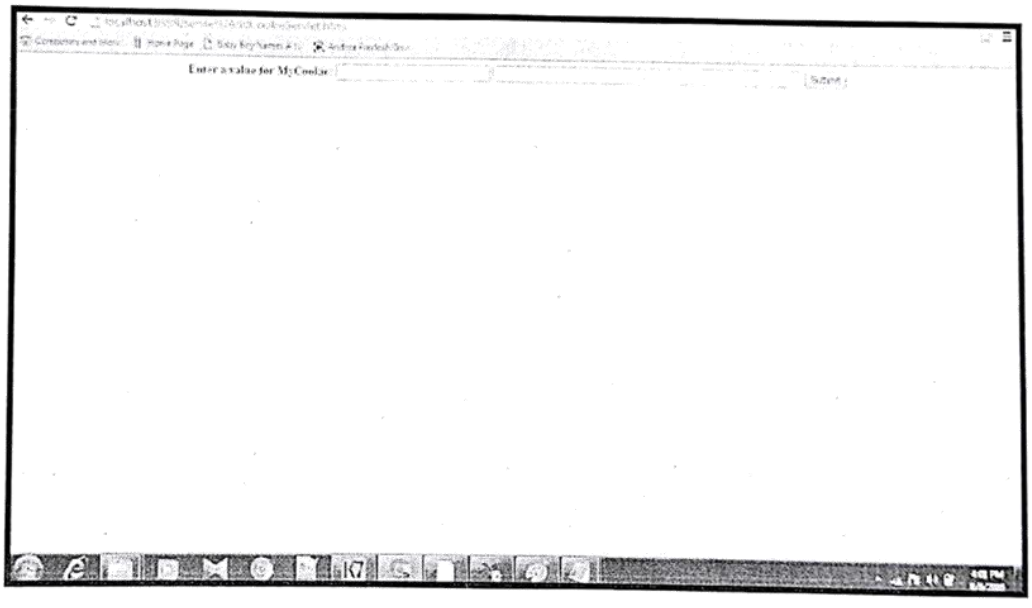
GetCookieServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class GetCookiesServlet extends HttpServlet
{
public void doGet(HttpServletRequest request,HttpServletResponse response)
throws ServletException, IOException
{
// Get cookies from header of HTTP request.
Cookie[] cookies = request.getCookies();
// Display these cookies.
response.setContentType("text/html");
```

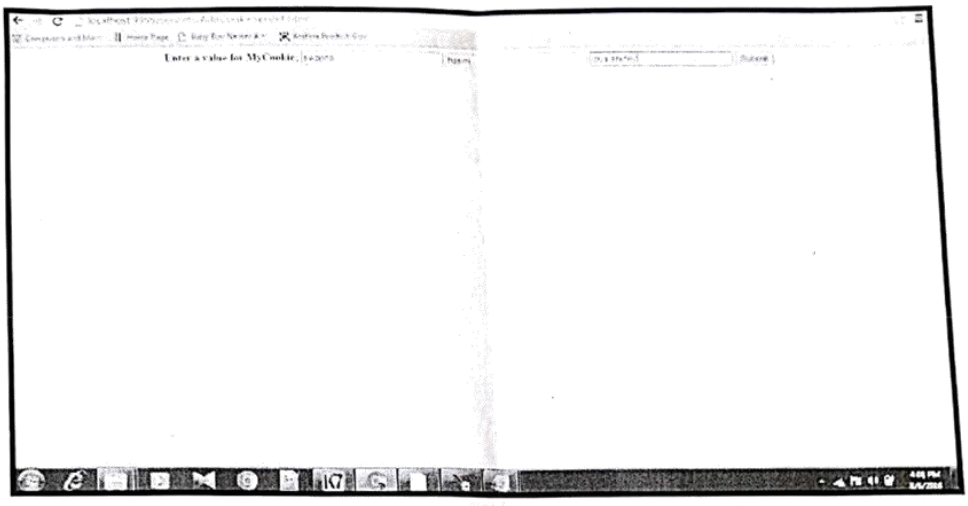
Scanned by CamScanner

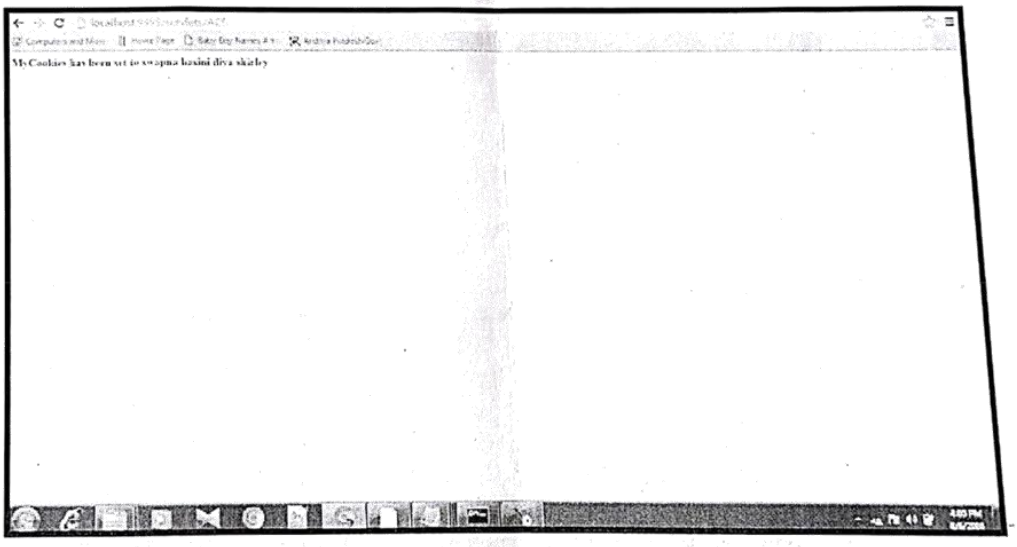
```
PrintWriter pw = response.getWriter();
pw.println("<B>");
for(int i = 0; i < cookies.length; i++)
 {
String name = cookies[i].getName();
String value = cookies[i].getValue();
pw.println("name = " + name +
"; value = " + value);
}
pw.close();
}
}
```

Output:



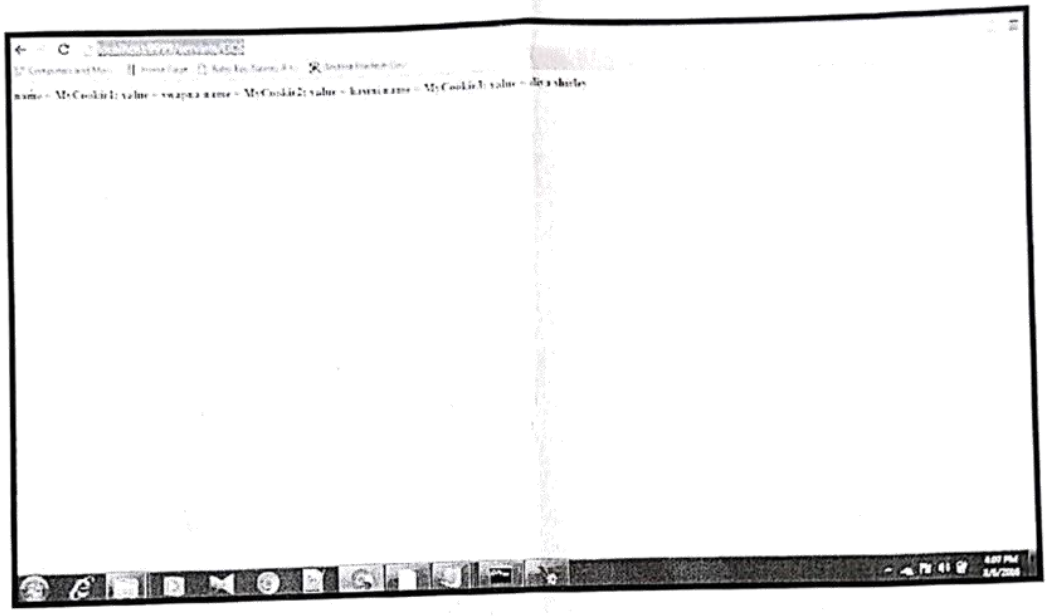
Scanned by CamScanner





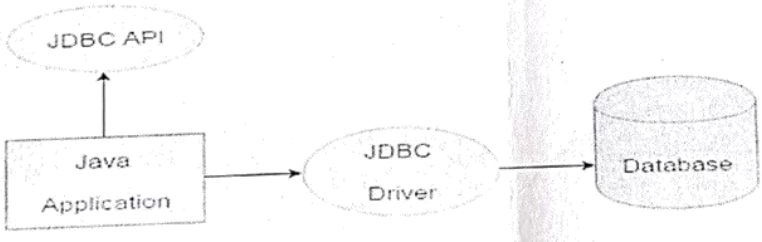
Scanned by CamScanner

Type http://localhost:9999/servlets/GCS to retrive cookies information



Connecting to a database using JDBC

Java JDBC is a java API to connect and execute query with the database. JDBC API uses jdbc drivers to connect with the database.

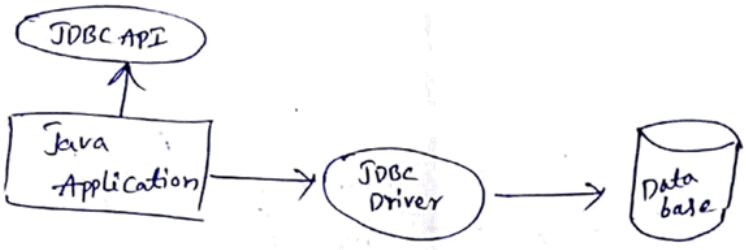


Why use JDBC

Scanned by CamScanner

Connecting to a database using JDBC:-

JDBC Stands Java Database Connectivity & it is a java API to connect & execute query with db. JDBC API uses jdbc drivers to connect with the db.



Why use JDBC,

Before JDBC, ODBC API was the db API to connect & execute query with the db. But, ODBC API uses ODBC driver while written in c lang (i.e, platform dependent & unsecured). That's why Java has define its own API (JDBC API) that uses JDBC driver (written Java lang) (i.e, platform independent & secured).

JDBC driver

It is a s/w component that enables java application to interact with db. There 4 types of JDBC drivers.

① JDBC - ODBC bridge driver,

It uses ODBC driver to connect to the db. this driver converts JDBC method calls to the odbc function calls. This is now discouraged because of thin driver.

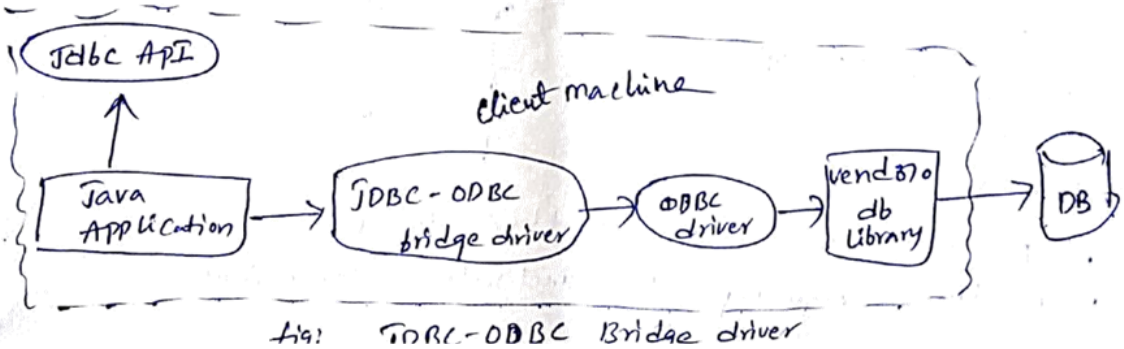


fig: JDBC-ODBC Bridge driver

Scanned by CamScanner

Adv:-

- easy to use
- Can be easily connected to any db.

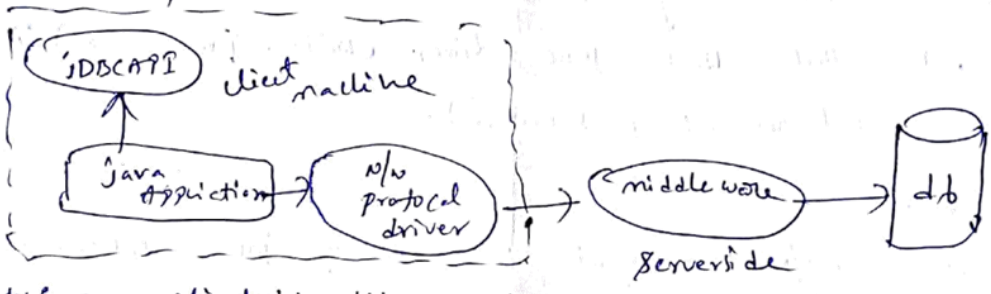
2) Native- API driver

It uses the client side libraries of the db. The driver converts JDBC method calls into native calls of the db API. It is not written entirely in java.

Adv:- performance upgraded then JDBC-ODBC driver

3) Network protocol driver

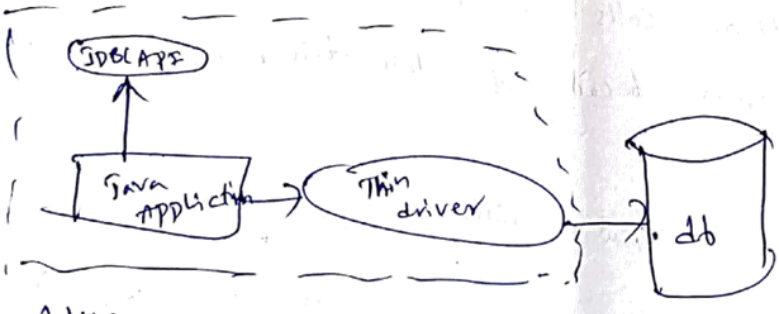
It uses middleware (application server) that converts JDBC calls directly or indirectly into vendor-specific db protocol. It is fully written in java.



Adv:- no client side library require bcz of application server

4) Thin driver

This driver converts JDBC calls directly into the vendor specific db protocol. It is fully written in java



Adv:-

- Better performance then all other drivers
- No s/w is required at client side or serverside

Scanned by CamScanner

Before JDBC, ODBC API was the database API to connect and execute query with the database. But, ODBC API uses ODBC driver which is written in C language (i.e. platform dependent and unsecured). That is why Java has defined its own API (JDBC API) that uses JDBC drivers (written in Java language).

JDBC Driver

JDBC Driver is a software component that enables java application to interact with the database.There are 4 types of JDBC drivers:

1. JDBC-ODBC bridge driver
2. Native-API driver (partially java driver)
3. Network Protocol driver (fully java driver)
4. Thin driver (fully java driver)

Explain all these 4 drivers (refer your class Notes)

1) JDBC-ODBC bridge driver

The JDBC-ODBC bridge driver uses ODBC driver to connect to the database. The JDBC-ODBC bridge driver converts JDBC method calls into the ODBC function calls. This is now discouraged because of thin driver.

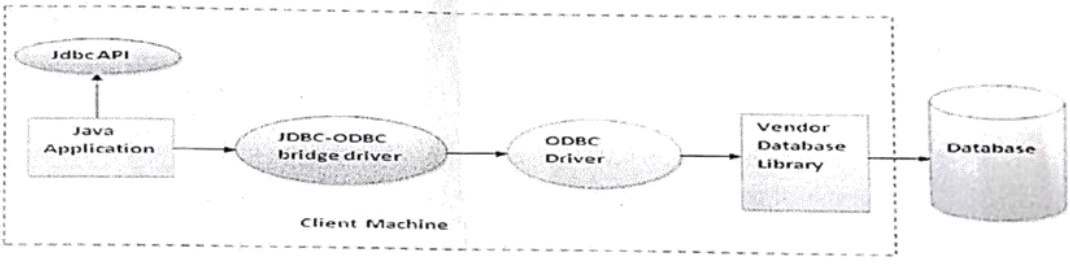


Figure- JDBC-ODBC Bridge Driver

Advantages:

- easy to use.
- can be easily connected to any database.

Disadvantages:

- Performance degraded because JDBC method call is converted into the ODBC function calls.
- The ODBC driver needs to be installed on the client machine.

Scanned by CamScanner

5 Steps to connect to the database in java

There are 5 steps to connect any java application with the database in java using JDBC. They are as follows:

1) Register the driver class

The forName() method of Class class is used to register the driver class. This method is used to dynamically load the driver class.

Syntax of forName() method

```
1. public static void forName(String clasSName)throws ClassNotFoundException
```

Example to register the OracleDriver class

```
Class.forName("oracle.jdbc.driver.OracleDriver");
```

2) Create the connection object

The getConnection() method of DriverManager class is used to establish connection with the database.

Syntax of getConnection() method

```
1. 1) public static Connection getConnection(String url)throws SQLException
2. 2) public static Connection getConnection(String url,String name,String password)
3. throws SQLException
```

Example to establish connection with the Oracle database

```
1. Connection con=DriverManager.getConnection(
2. "jdbc:oracle:thin:@localhost:1521:xe","system","password");
```

3) Create the Statement object

The createStatement() method of Connection interface is used to create statement. The object of statement is responsible to execute queries with the database.

Syntax of createStatement() method

```
1. public Statement createStatement()throws SQLException
```

Scanned by CamScanner

Example to create the statement object

```
1. Statement stmt=con.createStatement();
```

4) Execute the query

The executeQuery() method of Statement interface is used to execute queries to the database. This method returns the object of ResultSet that can be used to get all the records of a table.

Syntax of executeQuery() method

```
1. public ResultSet executeQuery(String sql)throws SQLException
```

Example to execute query

```
1. ResultSet rs=stmt.executeQuery("select * from emp");
2.
3. while(rs.next()){
4. System.out.println(rs.getInt(1)+" "+rs.getString(2));
5. }
```

5) Close the connection object

By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection.

Syntax of close() method

```
1. public void close()throws SQLException
```

Example to close connection

```
1. con.close();
```

Example to connect to the Oracle database in java

For connecting java application with the oracle database, you need to follow 5 steps to perform database connectivity. In this example we are using Oracle10g as the database. So we need to know following information for the oracle database:

1. **Driver class:** The driver class for the oracle database

Scanned by CamScanner

is **oracle.jdbc.driver.OracleDriver**.

2. **Connection URL:** The connection URL for the oracle10G database is **jdbc:oracle:thin:@localhost:1521:xe** where jdbc is the API, oracle is the database, thin is the driver, localhost is the server name on which oracle is running, we may also use IP address, 1521 is the port number and XE is the Oracle service name. You may get all these information from the tnsnames.ora file.
3. **Username:** The default username for the oracle database is **system**.
4. **Password:** Password is given by the user at the time of installing the oracle database.

Let's first create a table in oracle database.

```
1. create table emp(id number(10),name varchar2(40),age number(3));
2. import java.sql.*;
3. class OracleCon{
4. public static void main(String args[]){
5. try{
6. //step1 load the driver class
7. Class.forName("oracle.jdbc.driver.OracleDriver");
8.
9. //step2 create  the connection object
10. Connection con=DriverManager.getConnection(
11. "jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
12.
13. //step3 create the statement object
14. Statement stmt=con.createStatement();
15.
16. //step4 execute query
17. ResultSet rs=stmt.executeQuery("select * from emp");
18. while(rs.next())
19. System.out.println(rs.getInt(1)+"  "+rs.getString(2)+"  "+rs.getString(3));
20.
21. //step5 close the connection object
22. con.close();
23.
24. }catch(Exception e){ System.out.println(e);}
```

Scanned by CamScanner

1. **Driver class:** The driver class for the mysql database is **com.mysql.jdbc.Driver**.
2. **Connection URL:** The connection URL for the mysql database is **jdbc:mysql://localhost:3306/sonoo** where jdbc is the API, mysql is the database, localhost is the server name on which mysql is running, we may also use IP address, 3306 is the port number and sonoo is the database name. We may use any database, in such case, you need to replace the sonoo with your database name.
3. **Username:** The default username for the mysql database is **root**.
4. **Password:** Password is given by the user at the time of installing the mysql database. In this example, we are going to use root as the password.

Let's first create a table in the mysql database, but before creating table, we need to create database first.

```
1. create database sonoo;
2. use sonoo;
3. create table emp(id int(10),name varchar(40),age int(3));
```

Example to Connect Java Application with mysql database

In this example, sonoo is the database name, root is the username and password.

```
1. import java.sql.*;
2. class MysqlCon{
3. public static void main(String args[]){
4. try{
5. Class.forName("com.mysql.jdbc.Driver");
6. Connection con=DriverManager.getConnection(
7. "jdbc:mysql://localhost:3306/sonoo","root","root");
8. //here sonoo is database name, root is username and password
9. Statement stmt=con.createStatement();
10. ResultSet rs=stmt.executeQuery("select * from emp");
11. while(rs.next())
12. System.out.println(rs.getInt(1)+"  "+rs.getString(2)+"  "+rs.getString(3));
13. con.close();
14. }catch(Exception e){ System.out.println(e);}
15. }
```

Scanned by CamScanner

```
25.
26. }
27. }
```

To connect java application with the Oracle database ojdbc14.jar file is required to be loaded.

Two ways to load the jar file:

1. paste the ojdbc14.jar file in jre/lib/ext folder
2. set classpath

1) paste the ojdbc14.jar file in JRE/lib/ext folder:

Firstly, search the ojdbc14.jar file then go to JRE/lib/ext folder and paste the jar file here.

2) set classpath:

There are two ways to set the classpath:

- temporary
- permanent

How to set the temporary classpath:

Firstly, search the ojdbc14.jar file then open command prompt and write:

1. C:>set classpath=c:\folder\ojdbc14.jar;.;

How to set the permanent classpath:

Go to environment variable then click on new tab. In variable name write **classpath** and in variable value paste the path to ojdbc14.jar by appending ojdbc14.jar;.; as C:\oraclexe\app\oracle\product\10.2.0\server\jdbc\lib\ojdbc14.jar;.;

Example to connect to the mysql database in java

For connecting java application with the mysql database, you need to follow 5 steps to perform database connectivity.

In this example we are using MySql as the database. So we need to know following informations for the mysql database:

Scanned by CamScanner

1. **Driver class:** The driver class for the mysql database is **com.mysql.jdbc.Driver**.
2. **Connection URL:** The connection URL for the mysql database is **jdbc:mysql://localhost:3306/sonoo** where jdbc is the API, mysql is the database, localhost is the server name on which mysql is running, we may also use IP address, 3306 is the port number and sonoo is the database name. We may use any database, in such case, you need to replace the sonoo with your database name.
3. **Username:** The default username for the mysql database is **root**.
4. **Password:** Password is given by the user at the time of installing the mysql database. In this example, we are going to use root as the password.

Let's first create a table in the mysql database, but before creating table, we need to create database first.

```
1. create database sonoo;
2. use sonoo;
3. create table emp(id int(10),name varchar(40),age int(3));
```

Example to Connect Java Application with mysql database

In this example, sonoo is the database name, root is the username and password.

```
1. import java.sql.*;
2. class MysqlCon{
3. public static void main(String args[]){
4. try{
5. Class.forName("com.mysql.jdbc.Driver");
6. Connection con=DriverManager.getConnection(
7. "jdbc:mysql://localhost:3306/sonoo","root","root");
8. //here sonoo is database name, root is username and password
9. Statement stmt=con.createStatement();
10. ResultSet rs=stmt.executeQuery("select * from emp");
11. while(rs.next())
12. System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));
13. con.close();
14. }catch(Exception e){ System.out.println(e);}
15. }
```

Scanned by CamScanner

16. }

Download this example

The above example will fetch all the records of emp table.

To connect java application with the mysql database mysqlconnector.jar file is required to be loaded.

download the jar file mysql-connector.jar

Two ways to load the jar file:

1. paste the mysqlconnector.jar file in jre/lib/ext folder
2. set classpath

1) paste the mysqlconnector.jar file in JRE/lib/ext folder:

Download the mysqlconnector.jar file. Go to jre/lib/ext folder and paste the jar file here.

2) set classpath:

There are two ways to set the classpath:

- temporary
- permanent

How to set the temporary classpath

open command prompt and write:

1. C:>set classpath=c:\folder\mysql-connector-java-5.0.8-bin.jar;.;

How to set the permanent classpath

Go to environment variable then click on new tab. In variable name write **classpath** and in variable value paste the path to the mysqlconnector.jar file by appending mysqlconnector.jar;.; as C:\folder\mysql-connector-java-5.0.8-bin.jar;.;

- Example of Registration form in servlet

Scanned by CamScanner

Here, you will learn that how to create simple registration form in servlet. We are using oracle10g database. So you need to create a table first as given below:

```
CREATE TABLE "REGISTERUSER"
  ( "NAME" VARCHAR2(4000),
   "PASS" VARCHAR2(4000),
   "EMAIL" VARCHAR2(4000),
   "COUNTRY" VARCHAR2(4000)
  )
/
```

To create the registration page in servlet, we can separate the database logic from the servlet. But here, we are mixing the database logic in the servlet only for simplicity of the program. We will develop this page in JSP following DAO, DTO and Singleton design pattern later.

Example of Registration form in servlet

In this example, we have created the three pages.

- register.html
- Register.java
- web.xml

register.html

In this page, we have getting input from the user using text fields and combobox. The information entered by the user is forwarded to Register servlet, which is responsible to store the data into the database.

```
<html>
<body>
<form action="servlet/Register" method="post">

Name:<input type="text" name="userName"/><br/><br/>
Password:<input type="password" name="userPass"/><br/><br/>
Email Id:<input type="text" name="userEmail"/><br/><br/>
Country:
<select name="userCountry">
<option>India</option>
```

Scanned by CamScanner

```
<option>Pakistan</option>
<option>other</option>
</select>

<br/><br/>
<input type="submit" value="register"/>

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

Register.java

This servlet class receives all the data entered by user and stores it into the database. Here, we are performing the database logic. But you may separate it, which will be better for the web application.

```
import java.io.*;
import java.sql.*;
import javax.servlet.ServletException;
import javax.servlet.http.*;

public class Register extends HttpServlet {
public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

response.setContentType("text/html");
PrintWriter out = response.getWriter();

String n=request.getParameter("userName");
String p=request.getParameter("userPass");
String e=request.getParameter("userEmail");
String c=request.getParameter("userCountry");

try{
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con=DriverManager.getConnection(
"jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
```

Scanned by CamScanner

```
22.
23. PreparedStatement ps=con.prepareStatement(
24. "insert into registeruser values(?,?,?,?)");
25.
26. ps.setString(1,n);
27. ps.setString(2,p);
28. ps.setString(3,e);
29. ps.setString(4,c);
30.
31. int i=ps.executeUpdate();
32. if(i>0)
33. out.print("You are successfully registered...");
34.
35.
36. }catch (Exception e2) {System.out.println(e2);}
37.
38. out.close();
39. }
40.
41. }
```

web.xml file

The is the configuration file, providing information about the servlet.

```
1.  <web-app>
2.
3.  <servlet>
4.  <servlet-name>Register</servlet-name>
5.  <servlet-class>Register</servlet-class>
6.  </servlet>
7.
8.  <servlet-mapping>
9.  <servlet-name>Register</servlet-name>
10. <url-pattern>/servlet/Register</url-pattern>
11. </servlet-mapping>
12.
13. <welcome-file-list>
```

Scanned by CamScanner

```
    <welcome-file>register.html</welcome-file>
</welcome-file-list>

</web-app>
```

Scanned by CamScanner

lets start with Unit-IV

JSP (Java Server pages)

the Anatomy of JSP Page,

this is given by the Sun microsystems. This is the most important technology in real time. By using this we can develop dynamic web pages.

First to go for JSP tech, try to understand,

when ever we are using servlet tech there is a drawback,

Drawbacks of servlet technology

Somany are there

① when ever we are developing one servlet must & should we configure inside the web.xml file (assume that 100 servlets are there in our 100 x we have to applet configure in web.xml. unnecessarily increasing burden on developers

② whenever we are modifying the servlet must & should we need to stop the server & we need to compile the servlet & once again redeploy the application inside the server & restart the server.

③ Servlets are allowed by the only java code but not text & html code.

④ Whenever we are using the servlets presenting the data is very slow.

these many disadvantages are there when we are in servlet tech while developing the dynamic webpages to overcome these dis we have to go for JSP.

Advantages of JSP technology:

① whenever we are developing the one JSP page we no need to configure inside the web.xml file.

② presenting the data is very fast compare to servlets

③ whenever we are modifying the JSP we no need to stop the server & restart the server

④ The JSP pages is allowed by the html code & textual code & Java code also. These many adv are there, whenever we are developing one JSP page where you can place that page under root folder only when we develop one JSP page & save with the extention .jsp

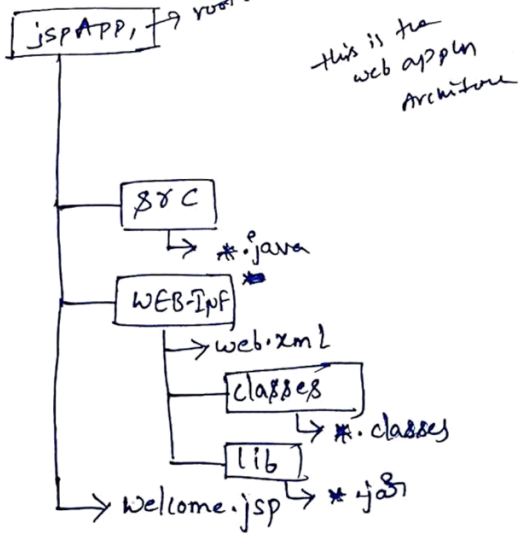
Scanned by CamScanner

Rules of Jsp Page

① whenever we are developing one jsp Page must & should we need to save extenstion is .jsp.

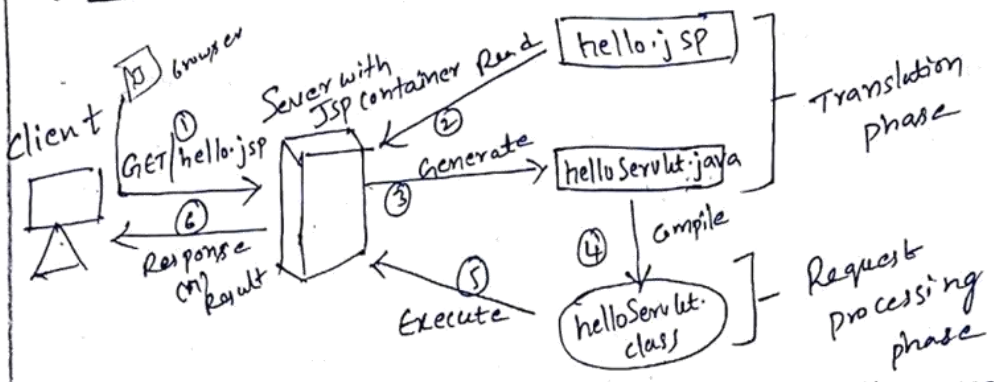
② After developing the jsp Page must & should we need to place inside the application scope/root folder.

Now we see the Architecture of the jsp,



Scanned by CamScanner

Jsp Processing:-



1. Web browser sends an HTTP request to the web server requesting Jsp page.
2. Web server recognizes that the HTTP request by webbrowser is for Jsp page by checking the extension of the file (i.e, jsp).
3. webserver forwards HTTP Request to Jsp Container.
4. Jsp Container loads the Jsp page from disk & converts it into a servlet
5. Jsp contains then compiles the servlet into an executable class and forward original request to a servlet container.
6. Servlet Container loads & executes the Servlet class.
7. Servlet produces an output in HTml format
8. output produced by Servlet Container is then passes to the webserver inside an HTTP response.
9. webserver sends the HTTP response to webbrowser in the form of static Htmlcontent.
10. web browser loads the static page into the browser & thus user can view the dynamicaly generated page.

Scanned by CamScanner

Jsp Declaration tag:-

It is used to declare fields & methods.
the code written inside the jsp declaration tag is placed outside the service() method of auto generated Servlet.
so it doesn't get memory at each request.

Syntax:-

<%! field or method declaration %>

Example:- In this example of Jsp Declaration tag, we are declaring the field & printing the value of the declared field using the jsp expression tag.

index.jsp

```
<html>
  <body>
    <%! int data=50; %>
    <%= "value of the variable is :" + data %>
  </body>
</html>
```

Example of Jsp declaration tag that declares method.

index.jsp

```
<html>
  <body>
    <%!
    int cube(int n)
    {
      return n*n*n;
    }
    %>
    <%= "cube of 3 is :" + cube(3) %>
  </body>
</html>
```

Scanned by CamScanner

JSP directives

The **jsp directives** are messages that tells the web container how to translate a JSP page into the corresponding servlet.

There are three types of directives:

- page directive
- include directive
- taglib directive

Syntax of JSP Directive

<%@ directive attribute="value" %>

JSP page directive

The page directive defines attributes that apply to an entire JSP page.

Syntax of JSP page directive

<%@ page attribute="value" %>

Attributes of JSP page directive

- import
- contentType
- extends
- info
- buffer
- language
- isELIgnored
- isThreadSafe
- autoFlush
- session
- pageEncoding
- errorPage
- isErrorPage

1)import

The import attribute is used to import class,interface or all the members of a package.It is similar to import keyword in java class or interface.

Example of import attribute

Scanned by CamScanner

```
1. <html>
2. <body>
3.
4. <%@ page import="java.util.Date" %>
5. Today is: <%= new Date() %>
6.
7. </body>
8. </html>
```

2)contentType

The contentType attribute defines the MIME(Multipurpose Internet Mail Extension) type of the HTTP response.The default value is "text/html;charset=ISO-8859-1".

Example of contentType attribute

```
1. <html>
2. <body>
3.
4. <%@ page contentType=application/msword %>
5. Today is: <%= new java.util.Date() %>
6.
7. </body>
8. </html>
```

9. 3)extends

The extends attribute defines the parent class that will be inherited by the generated servlet.It is rarely used.

4)info

This attribute simply sets the information of the JSP page which is retrieved later by using getServletInfo() method of Servlet interface.

Example of info attribute

```
1. <html>
2. <body>
3.
4. <%@ page info="composed by Sonoo Jaiswal" %>
5. Today is: <%= new java.util.Date() %>
6.
7. </body>
8. </html>
```

The web container will create a method getServletInfo() in the resulting servlet.For example:

Scanned by CamScanner

```
1. public String getServletInfo() {
2.   return "composed by Sonoo Jaiswal";
3. }
```

5)buffer

The buffer attribute sets the buffer size in kilobytes to handle output generated by the JSP page.The default size of the buffer is 8Kb.

Example of buffer attribute

```
1. <html>
2. <body>
3.
4. <%@ page buffer="16kb" %>
5. Today is: <%= new java.util.Date() %>
6.
7. </body>
8. </html>
```

6)language

The language attribute specifies the scripting language used in the JSP page. The default value is "java".

7)isELIgnored

We can ignore the Expression Language (EL) in jsp by the isELIgnored attribute. By default its value is false i.e. Expression Language is enabled by default. We see Expression Language later.

```
1. <%@ page isELIgnored="true" %>//Now EL will be ignored
```

8)isThreadSafe

Servlet and JSP both are multithreaded.If you want to control this behaviour of JSP page, you can use isThreadSafe attribute of page directive.The value of isThreadSafe value is true.If you make it false, the web container will serialize the multiple requests, i.e. it will wait until the JSP finishes responding to a request before passing another request to it.If you make the value of isThreadSafe attribute like:

```
<%@ page isThreadSafe="false" %>
```

The web container in such a case, will generate the servlet as:

```
1. public class SimplePage_jsp extends HttpJspBase
```

Scanned by CamScanner

```
2.  implements SingleThreadModel{
3.  .......
4.  }
```

9)errorPage:**The errorPage attribute is used to define the error page, if exception occurs in the current page, it will be redirected to the error page.**

Example of errorPage attribute

```
1.  //index.jsp
2.  <html>
3.  <body>
4.
5.  <%@ page errorPage="myerrorpage.jsp" %>
6.
7.   <%= 100/0 %>
8.
9.  </body>
10. </html>
```

10)isErrorPage

The isErrorPage attribute is used to declare that the current page is the error page.

Example of isErrorPage attribute

```
1.  //myerrorpage.jsp
2.  <html>
3.  <body>
4.
5.  <%@ page isErrorPage="true" %>
6.
7.   Sorry an exception occured!<br/>
8.  The exception is: <%= exception %>
9.
10. </body>
11. </html>
```

Jsp Include Directive

The include directive is used to include the contents of any resource it may be jsp file, html file or text file. The include directive includes the original content of the included resource at page translation time (the jsp page is translated only once so it will be better to include static resource).

Advantage of Include directive

Scanned by CamScanner

Code Reusability

Syntax of include directive

```
1. <%@ include file="resourceName" %>
```

Example of include directive

In this example, we are including the content of the header.html file. To run this example you must create an header.html file.

```
1. <html>
2. <body>
3.
4. <%@ include file="header.html" %>
5.
6. Today is: <%= java.util.Calendar.getInstance().getTime() %>
7.
8. </body>
9. </html>
```

JSP Taglib directive

The JSP taglib directive is used to define a tag library that defines many tags. We use the TLD (Tag Library Descriptor) file to define the tags. In the custom tag section we will use this tag so it will be better to learn it in custom tag.

Syntax JSP Taglib directive

```
1. <%@ taglib uri="uriofthetaglibrary" prefix="prefixoftaglibrary" %>
```

Example of JSP Taglib directive

In this example, we are using our tag named currentDate. To use this tag we must specify the taglib directive so the container may get information about the tag.

```
<html>

  <body>

<%@ taglib uri="http://www.javatpoint.com/tags" prefix="mytag" %>

<mytag:currentDate/>

</body></html>
```

Scanned by CamScanner

JSP Scriptlet tag (Scripting elements)

In JSP, java code can be written inside the jsp page using the scriptlet tag. Let's see what are the scripting elements first.

JSP Scripting elements

The scripting elements provides the ability to insert java code inside the jsp. There are three types of scripting elements:

- scriptlet tag
- expression tag
- declaration tag

JSP scriptlet tag

A scriptlet tag is used to execute java source code in JSP. Syntax is as follows:

```
1. <% java source code %>
```

Example of JSP scriptlet tag

In this example, we are displaying a welcome message.

```
1. <html>
2. <body>
3. <% out.print("welcome to jsp"); %>
4. </body>
5. </html>
```

Example of JSP scriptlet tag that prints the user name

In this example, we have created two files index.html and welcome.jsp. The index.html file gets the username from the user and the welcome.jsp file prints the username with the welcome message.

File: index.html

```
1. <html>
2. <body>
3. <form action="welcome.jsp">
4. <input type="text" name="uname">
```

Scanned by CamScanner

```
5. <input type="submit" value="go"><br/>
6. </form>
7. </body>
8. </html>
```

File: welcome.jsp

```
1. <html>
2. <body>
3. <%
4. String name=request.getParameter("uname");
5. out.print("welcome "+name);
6. %>
7. </form>
8. </body>
9. </html>
```

JSP expression tag

The code placed within **JSP expression tag** is *written to the output stream of the response*. So you need not write out.print() to write data. It is mainly used to print the values of variable or method.

Syntax of JSP expression tag

```
1. <%= statement %>
```

Example of JSP expression tag

In this example of jsp expression tag, we are simply displaying a welcome message.

```
1. <html>
2. <body>
3. <%= "welcome to jsp" %>
4. </body>
5. </html>
```

Example of JSP expression tag that prints current time

Scanned by CamScanner

To display the current time, we have used the getTime() method of Calendar class. The getTime() is an instance method of Calendar class, so we have called it after getting the instance of Calendar class by the getInstance() method.

index.jsp

```
1. <html>
2. <body>
3. Current Time: <%= java.util.Calendar.getInstance().getTime() %>
4. </body>
5. </html>
```

Example of JSP expression tag that prints the user name

In this example, we are printing the username using the expression tag. The index.html file gets the username and sends the request to the welcome.jsp file, which displays the username.

File: index.jsp

```
1. <html>
2. <body>
3. <form action="welcome.jsp">
4. <input type="text" name="uname"><br/>
5. <input type="submit" value="go">
6. </form>
7. </body>
8. </html>
```

File: welcome.jsp

```
1. <html>
2. <body>
3. <%= "Welcome "+request.getParameter("uname") %>
4. </body>
5. </html>
```

JSP Declaration Tag

The **JSP declaration tag** is used *to declare fields and methods.*

The code written inside the jsp declaration tag is placed outside the service() method of auto generated servlet.

Scanned by CamScanner

So it doesn't get memory at each request.

Syntax of JSP declaration tag

The syntax of the declaration tag is as follows:

1. <%! field or method declaration %>

Difference between JSP Scriptlet tag and Declaration tag

Jsp Scriptlet Tag	Jsp Declaration Tag
The jsp scriptlet tag can only declare variables not methods.	The jsp declaration tag can declare variables as well as methods.
The declaration of scriptlet tag is placed inside the _jspService() method.	The declaration of jsp declaration tag is placed outside the _jspService() method.

Example of JSP declaration tag that declares field

In this example of JSP declaration tag, we are declaring the field and printing the value of the declared field using the jsp expression tag.

index.jsp

```
1. <html>
2. <body>
3. <%! int data=50; %>
4. <%= "Value of the variable is:"+data %>
5. </body>
6. </html>
```

Example of JSP declaration tag that declares method

In this example of JSP declaration tag, we are defining the method which returns the cube of given number and calling this method from the jsp expression tag. But we can also use jsp scriptlet tag to call the declared method.

index.jsp

```
1. <html>
2. <body>
3. <%!
```

Scanned by CamScanner

```
4. int cube(int n){
5. return n*n*n*;
6. }
7. %>
8. <%= "Cube of 3 is:"+cube(3) %>
9. </body>
10. </html>
```

JSP Implicit Objects

There are **9 jsp implicit objects**. These objects are *created by the web container* that are available to all the jsp pages.

The available implicit objects are out, request, config, session, application etc.

A list of the 9 implicit objects is given below:

Object	Type
out	JspWriter
request	HttpServletRequest
response	HttpServletResponse
config	ServletConfig
application	ServletContext
session	HttpSession
pageContext	PageContext
page	Object
exception	Throwable

1) JSP out implicit object

For writing any data to the buffer, JSP provides an implicit object named out. It is the object of JspWriter. In case of servlet you need to write:

```
1. PrintWriter out=response.getWriter();
```

Scanned by CamScanner

But in JSP, you don't need to write this code.

Example of out implicit object

In this example we are simply displaying date and time.

index.jsp

```
1. <html>
2. <body>
3. <% out.print("Today is:"+java.util.Calendar.getInstance().getTime()); %>
4. </body>
5. </html>
```

Output

javatpoint.com

2) JSP request implicit object

The **JSP request** is an implicit object of type HttpServletRequest i.e. created for each jsp request by the web container. It can be used to get request information such as parameter, header information, remote address, server name, server port, content type, character encoding etc.

It can also be used to set, get and remove attributes from the jsp request scope.

Let's see the simple example of request implicit object where we are printing the name of the user with welcome message.

Example of JSP request implicit object

index.html

```
1. <form action="welcome.jsp">
```

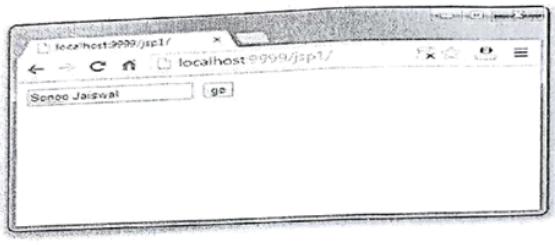
Scanned by CamScanner

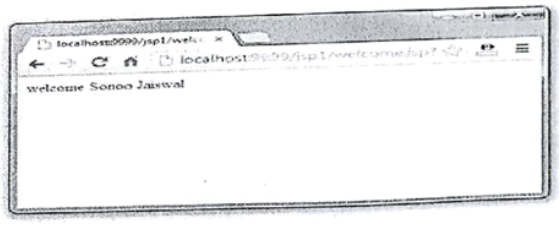
```
2. <input type="text" name="uname">
3. <input type="submit" value="go"><br/>
4. </form>
```

welcome.jsp

```
1. <%
2. String name=request.getParameter("uname");
3. out.print("welcome "+name);
4. %>
```

Output





3) JSP response implicit object

In JSP, response is an implicit object of type HttpServletResponse. The instance of HttpServletResponse is created by the web container for each jsp request.

It can be used to add or manipulate response such as redirect response to another resource, send error etc.

Let's see the example of response implicit object where we are redirecting the response to the Google.

Scanned by CamScanner

Example of response implicit object

index.html

```
1. <form action="welcome.jsp">
2. <input type="text" name="uname">
3. <input type="submit" value="go"><br/>
4. </form>
```

welcome.jsp

```
1. <%
2. response.sendRedirect("http://www.google.com");
3. %>
```

Output

4) JSP config implicit object

In JSP, config is an implicit object of type *ServletConfig*. This object can be used to get initialization parameter for a particular JSP page. The config object is created by the web container for each jsp page.

Generally, it is used to get initialization parameter from the web.xml file.

Example of config implicit object:

index.html

Scanned by CamScanner

```
1. <form action="welcome">
2. <input type="text" name="uname">
3. <input type="submit" value="go"><br/>
4. </form>
```

web.xml file

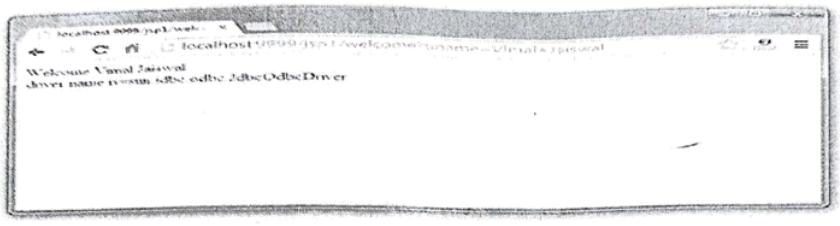
```
1. <web-app>
2.
3. <servlet>
4. <servlet-name>sonoojaiswal</servlet-name>
5. <jsp-file>/welcome.jsp</jsp-file>
6.
7. <init-param>
8. <param-name>dname</param-name>
9. <param-value>sun.jdbc.odbc.JdbcOdbcDriver</param-value>
10. </init-param>
11.
12. </servlet>
13.
14. <servlet-mapping>
15. <servlet-name>sonoojaiswal</servlet-name>
16. <url-pattern>/welcome</url-pattern>
17. </servlet-mapping>
18.
19. </web-app>
```

welcome.jsp

```
1. <%
2. out.print("Welcome "+request.getParameter("uname"));
3.
4. String driver=config.getInitParameter("dname");
5. out.print("driver name is="+driver);
6. %>
```

Output

Scanned by CamScanner



5) JSP application implicit object

In JSP, application is an implicit object of type *ServletContext*.

The instance of ServletContext is created only once by the web container when application or project is deployed on the server.

This object can be used to get initialization parameter from configuaration file (web.xml). It can also be used to get, set or remove attribute from the application scope.

This initialization parameter can be used by all jsp pages.

Example of application implicit object:

index.html

```
1. <form action="welcome">
2. <input type="text" name="uname">
3. <input type="submit" value="go"><br/>
4. </form>
```

web.xml file

```
1. <web-app>
2.
3. <servlet>
```

Scanned by CamScanner

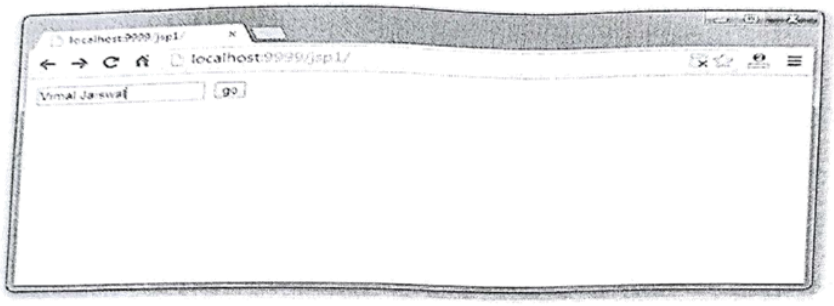
```
<servlet-name>sonoojaiswal</servlet-name>
<jsp-file>/welcome.jsp</jsp-file>
</servlet>

<servlet-mapping>
<servlet-name>sonoojaiswal</servlet-name>
<url-pattern>/welcome</url-pattern>
</servlet-mapping>

<context-param>
<param-name>dname</param-name>
<param-value>sun.jdbc.odbc.JdbcOdbcDriver</param-value>
</context-param>

</web-app>
```

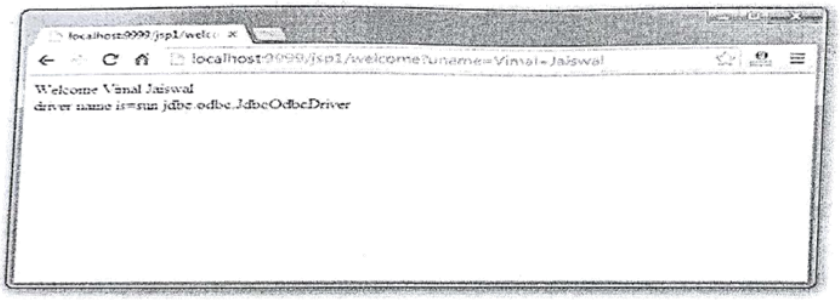
welcome.jsp

```
<%

out.print("Welcome "+request.getParameter("uname"));

String driver=application.getInitParameter("dname");
out.print("driver name is="+driver);

%>
```

Output

Scanned by CamScanner





6) session implicit object

In JSP, session is an implicit object of type HttpSession.The Java developer can use this object to set,get or remove attribute or to get session information.

Example of session implicit object

index.html

```
1. <html>
2. <body>
3. <form action="welcome.jsp">
4. <input type="text" name="uname">
5. <input type="submit" value="go"><br/>
6. </form>
7. </body>
8. </html>
```

Scanned by CamScanner

welcome.jsp

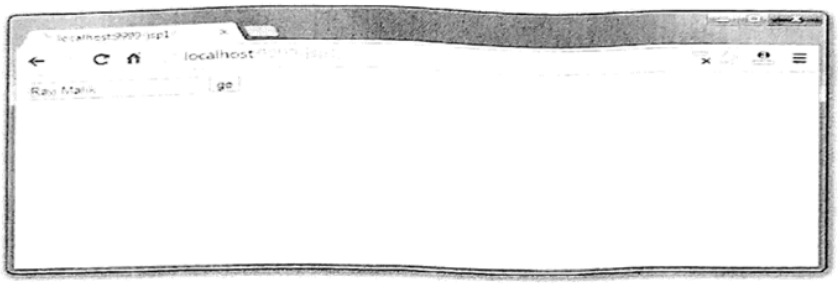
```
<html>
<body>
<%

String name=request.getParameter("uname");
out.print("Welcome "+name);

session.setAttribute("user",name);

<a href="second.jsp">second jsp page</a>

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

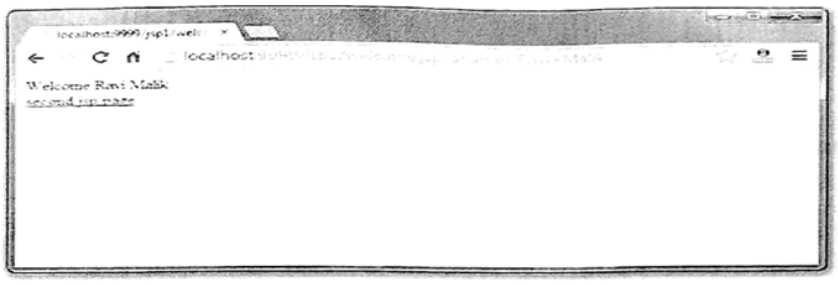
second.jsp

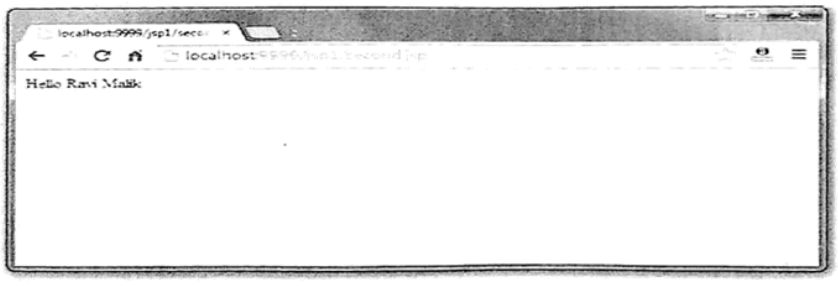
```
<html>
<body>
<%

String name=(String)session.getAttribute("user");
out.print("Hello "+name);

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

Output

Scanned by CamScanner







7) pageContext implicit object

In JSP, pageContext is an implicit object of type PageContext class.The pageContext object can be used to set,get or remove attribute from one of the following scopes:

- page
- request
- session
- application

In JSP, page scope is the default scope.

Scanned by CamScanner

Example of pageContext implicit object

index.html

```
1. <html>
2. <body>
3. <form action="welcome.jsp">
4. <input type="text" name="uname">
5. <input type="submit" value="go"><br/>
6. </form>
7. </body>
8. </html>
```

welcome.jsp

```
1. <html>
2. <body>
3. <%
4.
5. String name=request.getParameter("uname");
6. out.print("Welcome "+name);
7.
8. pageContext.setAttribute("user",name,PageContext.SESSION_SCOPE);
9.
10. <a href="second.jsp">second jsp page</a>
11.
12. %>
13. </body>
14. </html>
```

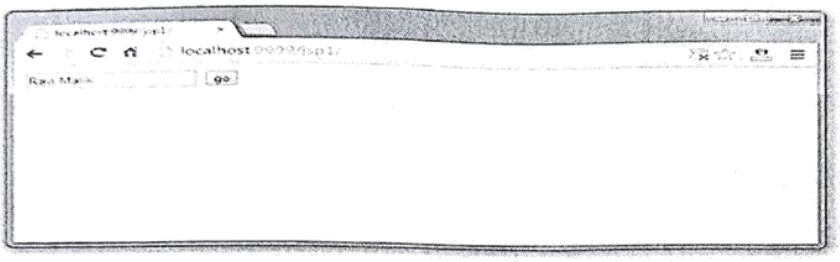
second.jsp

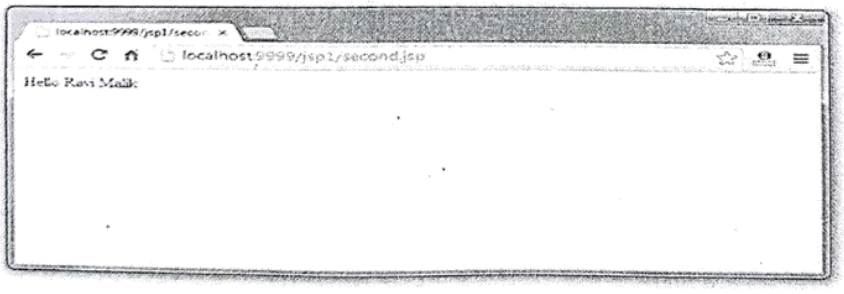
```
1. <html>
2. <body>
3. <%
4.
5. String name=(String)pageContext.getAttribute("user",PageContext.SESSION_SCOPE);
6. out.print("Hello "+name);
7.
```

Scanned by CamScanner

8. %>
9. </body>
10. </html>

Output





8) page implicit object:

In JSP, page is an implicit object of type Object class.This object is assigned to the reference of auto generated servlet class. It is written as:

Object page=this;

Scanned by CamScanner

For using this object it must be cast to Servlet type.For example:

<% (HttpServlet)page.log("message"); %>

Since, it is of type Object it is less used because you can use this object directly in jsp.For example:

<% this.log("message"); %>

9) exception implicit object

In JSP, exception is an implicit object of type java.lang.Throwable class. This object can be used to print the exception. But it can only be used in error pages.It is better to learn it after page directive. Let's see a simple example:

Example of exception implicit object:

error.jsp

```
1. <%@ page isErrorPage="true" %>
2. <html>
3. <body>
4.
5. Sorry following exception occured:<%= exception %>
6.
7. </body>
8. </html>
```

To get the full example, click here full example of exception handling in jsp. But, it will be better to learn it after the JSP Directives.

Scanned by CamScanner

Java Bean

A Java Bean is a java class that should follow following conventions:

- It should have a no-arg constructor. (A JavaBean must have a public default Constructor)
- It should be Serializable. interface.
- It should provide methods to set and get the values of the properties, known as getter and setter methods.

Why use Java Bean?

According to Java white paper, it is a reusable software component. A bean encapsulates many objects into one object, so we can access this object from multiple places. Moreover, it provides the easy maintenance.

Simple example of java bean class

```
//Employee.java

package mypack;
public class Employee implements java.io.Serializable{
private int id;
private String name;

public Employee(){}

        public void setId(int id){this.id=id;}

        public int getId(){return id;}

        public void setName(String name){this.name=name;}

        public String getName(){return name;}

        }
```

How to access the java bean class?

To access the java bean class, we should use getter and setter methods.

```
package mypack;
public class Test{
public static void main(String args[]){

Employee e=new Employee();//object is created

e.setName("Arjun");//setting value to the object

System.out.println(e.getName());
```

Scanned by CamScanner

```
10.
11.        }}
```

jsp:useBean action tag

The jsp:useBean action tag is used to locate or instantiate a bean class. If bean object of the Bean class is already created, it doesn't create the bean depending on the scope. But if object of bean is not created, it instantiates the bean.

Syntax of jsp:useBean action tag

```
1. <jsp:useBean id= "instanceName" scope= "page | request | session | application"
2. class= "packageName.className" type= "packageName.className"
3. beanName="packageName.className | <%= expression >" >
4. </jsp:useBean>
```

Attributes and Usage of jsp:useBean action tag

1. **id:** is used to identify the bean in the specified scope.
2. **scope:** represents the scope of the bean. It may be page, request, session or application. The default scope is page.
   - **page:** specifies that you can use this bean within the JSP page. The default scope is page.
   - **request:** specifies that you can use this bean from any JSP page that processes the same request. It has wider scope than page.
   - **session:** specifies that you can use this bean from any JSP page in the same session whether processes the same request or not. It has wider scope than request.
   - **application:** specifies that you can use this bean from any JSP page in the same application. It has wider scope than session.
3. **class:** instantiates the specified bean class (i.e. creates an object of the bean class) but it must have no-arg or no constructor and must not be abstract.
4. **type:** provides the bean a data type if the bean already exists in the scope. It is mainly used with class or beanName attribute. If you use it without class or beanName, no bean is instantiated.
5. **beanName:** instantiates the bean using the java.beans.Beans.instantiate() method.

Simple example of jsp:useBean action tag

In this example, we are simply invoking the method of the Bean class.

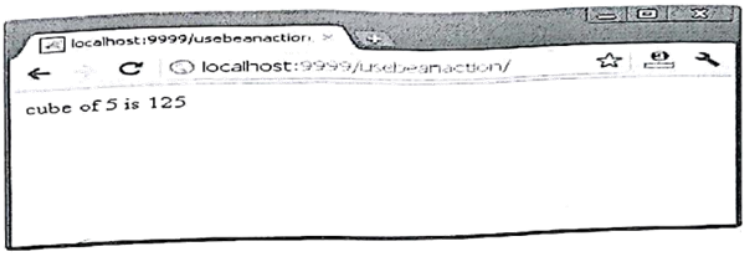
Calculator.java (a simple Bean class)

```
1. package com.javatpoint;
2. public class Calculator{
3.
4. public int cube(int n){return n*n*n;}
5.
6. }
```

Scanned by CamScanner

index.jsp file

```
<jsp:useBean id="obj" class="com.javatpoint.Calculator"/>
<%
int m=obj.cube(5);
out.print("cube of 5 is "+m);
%>
```



Connecting to database in JSP:

```
<%@ page import="java.sql.*" %>

<% Class.forName("sun.jdbc.odbc.JdbcOdbcDriver"); %>

<HTML>
  <HEAD>
    <TITLE>The Publishers Database Table </TITLE>
  </HEAD>

  <BODY>
    <H1>The Publishers Database Table </H1>

    <%
      Connection connection = DriverManager.getConnection(
        "jdbc:odbc:data", "userName", "password");

      Statement statement = connection.createStatement() ;
      ResultSet resultset = statement.executeQuery("select * from Publishers") ;
    %>

    <TABLE BORDER="1">
      <TR>
        <TH>ID</TH>
        <TH>Name</TH>
        <TH>City</TH>
        <TH>State</TH>
        <TH>Country</TH>
      </TR>
      <% while(resultset.next()){ %>
      <TR>
        <TD> <%= resultset.getString(1) %></td>
```

Scanned by CamScanner

```
        <TD> <%= resultset.getString(2) %></TD>
        <TD> <%= resultset.getString(3) %></TD>
        <TD> <%= resultset.getString(4) %></TD>
        <TD> <%= resultset.getString(5) %></TD>
      </TR>
      <% } %>
    </TABLE>
  </BODY>
</HTML>
```

A Java Bean is an java class

Accessing JavaBean in JSP:-

Basic JSP tags:-

```
<jsp : useBean        id = "bean's name"    scope = "bean's scope"    class = "bean's classname"/>
<jsp : setProperty    name = "bean's id"    property = "property name"    value = "value"/>
<jsp : getProperty    name = "bean's id"    property = "property name" />
```

Scanned by CamScanner