Unit-2 HTML

Hyper Text MarkUp Language

HTML:-

- It stand for Hyper Text MarkUp Language.
- It is the basic tool for designing a webpage.
- HTML is not a programming language.
- Hyper Text is simply a piece of text that work as a link.
- MarkUp Language is a way of writing layout information within documents.

Method of Preparing HTML Document:-

```
<html>
 <head>
 < title >  textname </title>
 </head>
 <body>
 _____
 _____

 </body>
 </html>
```

HTML Tags and Attributes:-

HTML tag is a keyword which is command giving instruction about how to display the context specifying the apperance of the context.

Tags are enclosed less than and greater than sign. A tag may have some attributes. Attribute is a

name value pair seperated by an equal sign.

1. Body Tag:-

It has several attributes.

(i) background color [bg color]
(ii) background design [background canbe image]
(iii) text color [color]

2. Paragraph Tag <P>:-

It is the most tag part of paragraph info each paragraph is alinged to left, to right or center.

<P align = "left" / "right" / "center">

3. Heading Tag:-

HTML is having 6 level of heading that are commonly used.

<h1>
<h2>
<h3>
:
<h6>

Types of Tag:-

1. Container or Pairtag eg- <html> ..... </html>
2. Singular tag eg-
 <p </br>

Physical Style Tag:-

<I> text </I>
 text
<u> text :</u>
_{text}
^{text}

List Tags are of 2 Types:-

a. Unorder list
b. Order List.

a. Unorder List:-

Start with the tag <ul> and end with unordered lists have 3 of types.

(i) Disc
(ii) Circle
(iii) Square

Eg-
 CS
 IT

OUTPUT:-
- CS
- IT

<ul type = "circle">
 CS
 IT

OUTPUT:-
- CS
- IT

b. Order List:-

Start with <ol> and end with

Ex:-

1. CS	A. CS	(i) CS	I. CS
2. IT	B. IT	(ii) IT	II IT
3. EC	C. EC	(iii) EC	III EC

Code:-

```
<ol type="1">
<li> CS </li>
<li> IT </li>
</ol>
```

O/P 1. CS 2. IT

```
<ol type="A">
<li> CS </li>
<li> IT </li>
</ol>
```

O/P. A. CS B. IT

```
<ol type="5" start="5">
<li> ME </li>
<li> CE </li>
</ol>
```

O/P 5. ME 6. CE

4. Anchor Tag [<a> ...]:-

```
<a href = "Target" > text </a>
```

Start — Anchor Tag — HyperText — Ending

- For Space :-
  [non breaking space]
- For quote :- "
- To display & - &
- For < - <
- For > - >

Table :-

```
<table border="1">
<tr>
<th> Sr.No </th>
<th> Rollno. </th>
<th> Name </th>
</tr>
<tr>
<td> 1 </td>
<td> Ashish Ojha </td>
<td> 1613210050 </td>
</tr>
<tr>
<td> 2 </td>
<td> Saurabh </td>
<td> 1613210099 </td>
</tr>
</table>
```

Sr.No	Name	Rollno
1	Ashish	1613210050
2	Saurabh	1613210099

Attributes of Table :-

1. Align:-

Horizontal allignment is control by the align attribute. It can be set left, right and center.

2. Width:-

Set the width to specific no. of pixel or to a percentage of the available screen width.

3. border:-

Control the border to the place around the table.

4. Colspan:-

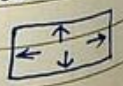
This attribute inside <th> or <td> tag instruct the browser to make the cell defined by the tag to makeup more than 1 column.

5. rowspan:-

This attribute work in the same way as the colspan attribute except that, it allow a cell to takeup more than 1 row.

6. Cellpadding:-

This attribute control the distance b/w the data in a cell and boundary of cell.



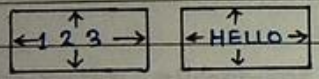
7. Cellspacing:-

Control the spacing between adjacent cell.


```
<table border= "1">
<tr>
<th> Sr.No.</th>
<th> Name </th>
<th> Branch</th>
</tr>
<tr>
<td> Ashish </td>
<td> 1 </td>
<td> CS </td>
</tr>
<tr>
<td> 2 </td>
<td> Akash </td>
<td> CS </td>
</tr>
<td.
<tr> colspan = "2" >
total = 2 </td>
</tr>
</table>
```

Sr.No	Name	Branch
1	Ashish	CS
2	Akash	CS
TOTAL=2		

Cellpadding :-




```
<table cellpadding = "10" >
<tr>
<th> Sr.No </th>
<th> Name </th>
</tr>
<tr>
<th> 123 </th>
<th> HELLO </th>
</tr>
</table>
```

Cellspacing :-

Sr.No	Name
123	ABC

```
<table cellspacing = "10" >
<tr>
<th> Sr.No </th>
<th> Name </th>
</tr>
<tr>
<th> 123 </th>
<td> ABC </td>
</tr>
</table>
```


Frame :-

Frame Set :-

```
< frame set rows = "50%, 50%" >
< frame set rows = "100, 200" >
< frame set rows = " * , 100" >
```

(frameset)

Frame Set for rows :-

" " " Column

```
< frame set cols = " 50%, 50%" >
< frame set cols = " 100, 200" >
< frame set rows = " *, 100" >
```

1. Rows :-

The rows attribute of frame set tag define horizontal frame and it can be set equal to a list of value.

- No. of pixel
- As a % of screen resolution.
- The symbol * which indicate the remaining space.

2. Column :-

The column attribute of frame set tag define vertical frame and it can be set to a list of value.

- No. of pixel
- As a % of screen resolution.
- The symbol * which indicate the remaining space.

Frame Attributes :-

1. Name :-

This attributes allow you to give a name to a frame. It can be target from hypertext link located in other frame.

2. SRC:-

This attribute is used to give the file name that should be loaded in the frame. Its value can be any URL.

Eg- SRC = "hello.html"

3. frame border:-

<frame border = "0"> it means border is off, border will not be shown.

<frame border = "1"> it means border is ON, border will be shown.

By default frame border = "1" >

4. Margin width:-

Specify the width of the space b/co left and right of the frame border and the frame contents.

<margin width = "10">

5. Margin height:-

Specify the height of the space b/co top and bottom of the frame border and the frame contents.

<margin height = "10" >

Eg-

Red	Green
Blue	

<frameset rows = "50%, 50%" >
<frameset cols = "50%, 50%" >
<frameset>
<frame name = "frame 1" style = bgcolor = Red>
<frame name = "frame 2" style = bgcolor = Green>

<frame name = "frame 3" style = bgcolor = blue >
</frameset>

image	f1	10 → Sun Publication of India
		→ IT
	f2	f3 → WT
		→ DBMS
		→ DAA

<framename = "frame 1" >
Sun Publication of India
</framename>
<framename = "frame 2" Cellpadding = "10" >
<image SRC = " " >
</framename>
<framename = "frame 3" >

 IT
 WT
 DBMS
 DAA

</framename>
</frameset>

Form :-

- A HTML form provide data gathering funtionality to a web page.
- HTML form can automatically submit data into its control to a web server.
- Form are used for creating front end that can contain elements like button, text, field in cohich you can type radio, button, check box and select list.

Application Area of Form :-

- Education skill
- Online purchasing
- Collecting feedback about a website.

Creating a Form :-

<form> tag is used to create a form.
It has 3 parts.

1. Form header
2. Inputs/ elements
3. Action button.

1. Form Header :-

In this we have following attributes.

a. Method
b. Action

a. Method → GET
→ POST

GET :-

- It just getting or returning data.
- Get method send the data capture by form element to the web server incoded into a URL which point to the web server.
  URL [Uniform Resource Location]

POST :-

- It is used for updation and sending email of data.
- The POST method send the data capture by form element back to web server as a stream bit stream of data.

Note :- By default get method is used.

b. Action :-

- It specify what CGI [Common Gateway Interface] script is used to process the data.
- Action attribute of the form tag point to the URL add. of a program on the web server that will process the form data captured and being sent back.

Eg- <formaction= "abc.php" Method = "GET" name = "form1">

2. Inputs Field / Element :-

The HTML form elements that can be specified as attributes to the i/p tags [<input>] are

(i) Text :-

<input type = "text" name = "name" maxlength = "size of length">

Eg-

<input type = "text" name = "Ashish" maxlength = "10">

(ii) Password :-

A password tag field in which each key appear as an *

<input type = "Password" name = "Password" maxlength = "size of length">

(iii) Check Box :-

It is used when more than one option is required to be selected.

<input type = "Checkbox" name = "WT" value = "WT">

Eg-1 <input type = "checkbox" name = "name" value = "name">

Eg-2 <input type = "checkbox" name = "DBMS" value = "DBMS"

(iv) Radio button :-

In this let a user select one of limit no. of choice.

Eg - <input type = "radio" name = "gender" value = "male" check on selected > male

(v) Submit button :-

<input type = "Submit" value = "Submit" >

Eg- FName

FName [ ]

LName [ ]

Submit

FName -

<input type = "text" name = "FName" value = "Ashish" >

LName -

<input type = "text" name = "LName" value = "Ojha">

Submit :-

<input type = "Submit" value = "Submit">

Alert Box Button :-

<input type = "button" onclick = "alert" ("Hello GNIOT") > click me.

(vi) Select :-

A select box is called drop down box which provide option to list down. Various option in the form of drop down list from where a user can select one or more option.

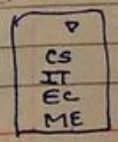
<select>

<option value = "cs"> cs </option>

:

:

:



Multipe line:-

<text area row = "10" col = "10" >

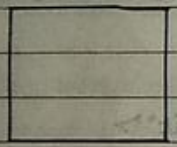
</ text area>

To move Text:-

LtoR:- <morquee > Hello GNIOT </ morquee>

TtoO:- <morquee direction = "up" > Hello GNIOT </ morquee>

RESUME



Address-

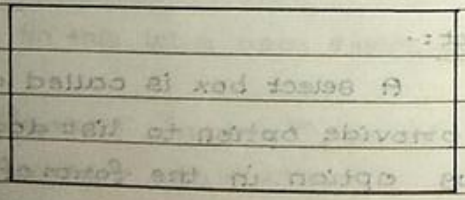
Email No-

Phone No-

Name-

Objective-

Qualification:



Declaration:

Signature-

CSS [Cascading Style Sheets]:-

If you want to apply similar setting for formating for all page in website this can be done by putting all the style rule in Stylesheet file and then importing and linking its with your HTML document. This method of linking is called Cascading Style Sheets.

Types of CSS:-

1. Inline
2. Internal → Inline → Embedded
3. External

Style:-

A style is a set of formating instruction that can be apply to a piece of text.

Syntax of Style:-

```
Select
  {
    declaration 1;
    declaration 2;
  }
```

Syntax of declaration:-

```
{
  Property 1 : value 1;
  Property 2 : value 2;
```

(i) Inline:-

By style attributes

```
<p style = "color: red ">
<p style = "Fontsize : 14"> Hello GNIOT
```

(ii) Embedded:-

By <style> tag

```
<html>
<head> <title>      </title>
<style>
    p
     { color: red ;
       fontsize : 14 ;
      }
</style>
</head>
<body>
   <P>  Hello GNIOT </P>
</body>
</html>
```

Note:- <style> tag is used within <head> tag always.

2. External:-

```
<link rel = " Stylesheet " type = " text / css "
                href = " abc . css ">
```

Eg-

```
p
{
 color: red;
}
```


Advantage of CSS:-

1. Save timing
2. Maintenance
3. Faster Downloading.

Style Rule :-

It is a set of HTML tag specifying the formating element. A style rule can be basically be split into two parts.

1. Selector
2. Declaration

1. Selector :-

It is a string that identify what elements corresponding rule apply to and is the 1st part of rule.

(i) Simple Selector :-

~~It is a string that identify what elements corresponding~~

```
H1
{
 Colour: blue;
}
```

(ii) HTML Selector :-

These selectors use the name of html element without brackets like <p> tag.

(iii) Class Selector:-

It is defined by a . (dot) followed by the class like class name.

(iv) ID Selector:-

In this style, id selector has a unique identifiers. An id-selector is processed by a (#) hash mark.

2. Declaration:-

This part of the rule is enclosed within curly brackets.

```
Selector
    {
    Property: value;  // any attribute like font
    }
```

03Aug-2018

Import css:-

```
<style type = "text/css">
@ import URL ("filename css");
</style.
```


Example- CS - blue
IT - Red
MCA - Green

abc.css

```
• CS {
    colour: blue;
  }
• IT {
    colon: red;
  }
• MCA
    { colon: green;
  }
```

main.html

```
<head>
<style type = "text/css">
 @ import URL ("abc.css");
</style>
<body>
    <P class = "cs"> CS </P>
    <P class = "IT"> IT </P>
    <P class = "MCA"> MCA </P>
 </body>
```

Ques:- What do you mean by HTML in the WebTech?

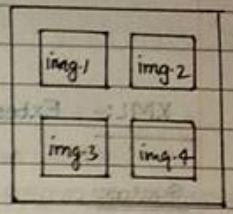
```
<p> what do you mean by < span class = "1" >
  HTML</span> in the WebTech ? </P>
```

main.html

```
<link rel = " Stylesheet" type= "text/ css"
                          href = "abc.css">
<body>
     <P class = " ans. "> HyperText Markup Language</p>
</body>
```

abc.css

```
ques
   {
    font-size : 12
    font- decloration : Bold
   }
one
   {
    fontsize : 16
    font declaration : Bold Italic
   }
```


```
<html>
<head>
<title>     </title>
</head>
<body>
     <div id = "box1" style = "bgcolour : Red" ; position : absolute
          ; top = 150; left : 200>
     <img SRC = "img1.gif" width = "30" height = "30">
     </div>

     <div id = "box 2". Style = "bg colon : Blue" ; position : absolute;
          top = 150; left : 350>
     <img SRC = "img2.gif" width = "30" height = "30">
     </div>

     <div id = "box 3" style = "bg color : Green" position : absolute
      top : 150; left : 500>
     <img SRC = "img3.gif  width = 30  height = 30>
     </div>

     <div id = "box 4"  style = "bgcolor : yellow" position :
          absolute ; top : 150; left : 650>
     <img SRC = "img4.gif  width = 30  height = 30>
     </div>
</body>
</html>
```

Unit-

XML:- Extensible Markup Language

Syntax:

```
<? xml version = "1.0" ?>
<root>
<child> Ashish </child>
<subchild> </subchild>
</root>
```

Example:-

```
<? xml version= "1.0"?>
<student>
<firstname> Ashish </firstname>
<lastname> Ojha </lastname>
<rollno> 1613210050 </rollno>
<gmail> abc@gmail.com </gmail>
<mob.no> 9838887766 </mobno.>
</student>
```


XML [Extensible Markup Language]:-

- XML is design to store transportdata.
- XML is designed to be self descriptive data.
- XML is Platform independent and Language independent.
- The main purpose of XML is not to format the data to displayed.

Ques: Why XML platform independent and language independent?

Ans:- • The main benefit of XML is that you can use it to take data from a program like Microsoft, SQL, convert it into XML then share that XML with other programs and platforms.

- You can communicate b/w two platform which are generally very difficult.

Difference b/w HTML and XML

HTML	XML
• HyperText Markup Language	• Extensible Markup Language
• HTML is case insensitive.	• XML is case sensitive
• HTML has it own predefined tag.	• while what makes XML flexible is that custom tag can be defined.
• HTML is about displaying data, Hence Static	• XML is about carrying information Hence dynamic.
• HTML is presentation Language.	• XML is neither a programming language nor a presentation language.

XML Rules:-

- All XML elements must have a closing tag.

Eg- <P> ..... </P>

- All XML element must be properly nested.

<name>
<email>
</email>
</name>
} in XML only

- XML tag are case sensitive.

<message> ..... </Message> -[Not allowed in XML]
<message> ..... </message> [Allowed]

- All XML document must have a root tag.

Example:-

<Student>
<fname> ..... </fname>
<lname> ..... </lname>
</student>

Entity Reference:-

Some character have a special meaning in XML, if you place a character like less than [" < "] inside in XML, it will generates an error because the parser interprets it as the start of a new element like-

&gt, &lt, ', ", ",

Example:- <message> Age <20 </message> error

<message> Age < 20 </message>

imp # Document Type Definition [DTD]:-

- A DTD can be declared inline XML documents or as an external documents.
- XML allow to define the structure of the documents so that the validity of the data can be checked. This structure is defined in our documents called Document Type Definition [DTD].
- DTD is like the Table layout of an XML documents. It defines the legal building blocks of an XML documents.

Syntax:- <!DOC TYPE>

Types of DTD:-

1. Internal
2. External

1. Internal:-

- The DTD is included inside the XML documents.

Syntax:- <! DOC TYPE root element name
[element_declaration]>

- Internal DTD can used on local system without logging on.

2. External DTD:-

The DTD reference is include inside the XML documents. The DTD saves with .dtd extension at same place.

Syntax:- <!DOCTYPE root element name SYSTEM "filename.dtd">.

Example:- <!DOC TYPE College SYSTEM "College.dtd">.

where

<!DOCTYPE is the tag that start a document type declaration.

College is the name of DTD is used which must correspond with the name of roots elements for the document.

SYSTEM "College.dtd" instruct the process on to fetch an external documents college are .dtd.

08 Aug 2018

Namespace:-

Building Block of an xml document:-

Five Building Block of an xml document.

1. Element
2. Attributes
3. Entity
4. PC data [Parse character DATA]
5. CDATA

1. Elements:-

- Elements are the main building blocks of both xml and html documents.
- xml elements can contain text and other elements.

* Assign the value to the element name.

<! Element Element name (element content) >

* Element with parse character data

<! Element element name (# PCDATA) >

* Element with children

<! Element elementname (child1, child2, ...)>

Eg- <! Element Address(Name, phone, email)>

<! Element Address (Name, phone, email)>

<! Element Name (# PCDATA)>

<! Element phone (# PCDATA)>

<! Element email (# PCDATA)>

2. Attributes:-

- It provide extra information about elements.
- Attributes are declared with an ATTLIST declaration.

<!ATTLIST element name attributename attribute type attributevalue>

Eg- <!ATTLIST Payment type CDATA "cheque">

3. Entity:-

- It can be declared internal or external.
- Entity should be define in doctype in xml document or in dtd.

(i) Internal Entity Declaration:-

Syntax- <! Entity entityname "entity value">

DTD- Example- <! Entity writer "Katori">

xml- <author> &writer; </author>

(ii) External Entity Declaration:-

Syntax: <! Entity entityname SYSTEM "URL">

4. PCDATA:-

- xml parser are used to parse all the text in an xml documents.
- Tag inside the PCDATA will be treated as markup and entity will be expanded.

Ex- <! Doctype employee SYSTEM "emp.dtd">
<address>
<name> Ashish </name>
<phone> 8826269389 </phone>
</address>

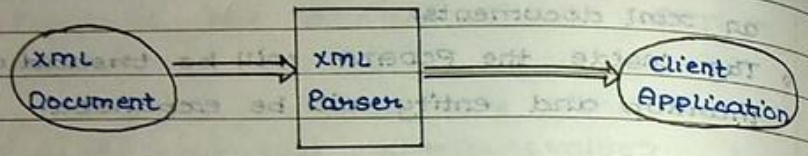
5. CDATA:-

- It contain the text which is not parse further in an xml document.
- Tag inside the CDATA tag are not treated as markup and entity will not be expanded.
- CDATA is used just after root element to make the data/ text unparse, so it will give the value of employee.

Root element → <address>
<![CDATA[
<name> Ashish </name>
<phone> 8826269389 </phone>]]>
</address>

14 Aug 2018

XML Parser:-



- An XML parser is slw library or package that provides interface for client application to work with an XML document.

- XML Parser validate the document and check that the document is well formatted.

- XML Parser is designed to read the XML document and create a way for program to use XML documents.

Old Version Internet Explorer Example:-

```
if (window.DOMParser)
    // Code for modern browser.
  { parser = new DOMparser();
    xmlDoc= parser.parseFromString(text,"text/xml");
  }
  else
    {
     xmlDoc= new ActiveXObject("microsoft.xmlDOM");
     xmlDoc.async = false;
     xmlDoc.loadXML(text);
    }
```


Old version of Internet explorer [Internet Explorer 5,6,7,8] donot support the DOM parser object then create an ActiveXObject.

Parsing a text String:-

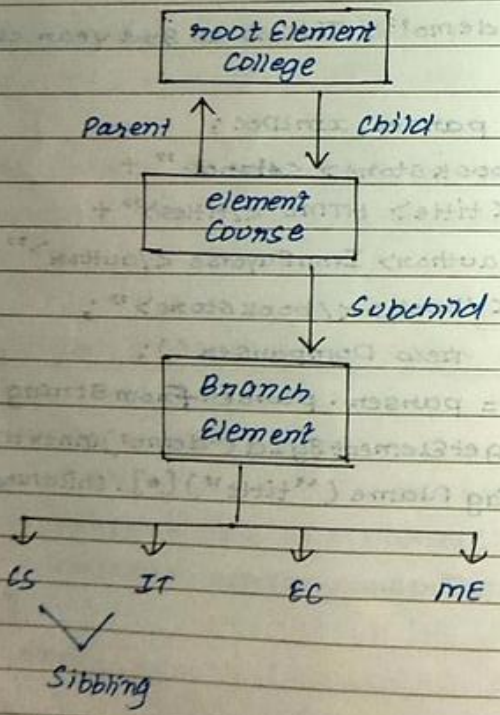
- This example parse a text string into an xml DOM object and extract the information from it with Javascript.

```
<body>
<h1 id="demo"> This is CS 3rd year class </h1>
<script>
 var text, parser, xmlDoc;
 text="<bookstore> <book>" +
      "<title> HTML </title>" +
      "<author> IvonBayorse </author>" +
      "</book></bookstore>";
  parser = new DOMparser();
  xmlDoc = parser.parseFromString(text,"text/xml");
document.getElementById('demo').innerHTML=xmlDoc.get
ElementByTagName("title")[0].childNode[0].NodeValue;
</script>
</body>
```

Types of Parsers:-

1. Tree Based (DOM) Document Object Model
2. Event Based (SAX) Simple API for xml.

```
<college>
<course>
<Branch>
<cs>
<it>
<ec>
```



1. DOM [Document Object Model]:-

- It is a collection of nodes or piece of information in a hierarchy.
- It define standard way to access and manipulate documents.
- Programmer modify/delete their elements and can also create new element.
- Element their content [text and attributes] are known as nodes.

16 Aug 18

Example of xml DOM:-

```
<?xml version="1.0" ?>
<college>
<Branch Category = "CS">
    <Student> 69 </Student>
    <Name> XYZ </Name>
    <id> 123 </id>
</Branch>
<Branch Category = "IT">
    <Student> 70 </Student>
    <Name> ABC </Name>
    <id> 789 </id>
</Branch> </college>
```

text = xmlDOC.getElementByTagName ("Student") [0].Childnode [0].Node value;
(Method)
(64)

DOM Method:-

x = node object.

1. x.getElementByTagName ("Name")

2. x.append child ("node")

We are Intgerting a child node to x

3. x.remove child ("node")

Xml DOM Property:-

1. x.Node name

2. x.Node_value

3. x.Parents node.

4. x.childnode

5. x.attributes

API = Application Program Interface

2. SAX [Simple API for XML]:-

- SAX is a Serial Access Parser API for XML.
- SAX only Load a small part of the XML file in memory.
- SAX provide a mechanism for reading data from an xml document.

20 Aug 2018

SAX for Parser Processing:-

A Parser which implement SAX function as a String parser, with an event driven API.

The user define a no. of call back method that will be called when event occur during parsing.

The SAX event include -

1. XML text node
2. xml element node
3. xml processing instructions.
4. xml comment

features of SAX parser:-

- It doesnot create any internal structure.
- Client doesnot know what method to call, they just overwrite the API and place his form code inside method.

- It work like an event handler in Java.

The xml document when parse through a SAX parser will generate a sequence of event like the following.

1. xml processing inst., name xml with attribute version = "1.0".

2. xml element start name root element, with an attributes parser = "value".

3. xml element start name first element.

4. xml text node, with data = "Some text".

5. xml element, end name and first element

6. xml element start, name 2nd element, with an attribute "Ashish" = value.

7. xml text node with data = "Ashish".

8. xml element start, named inside.

9. xml text node, with data = "Inline text"

10. xml element end, named Inline.

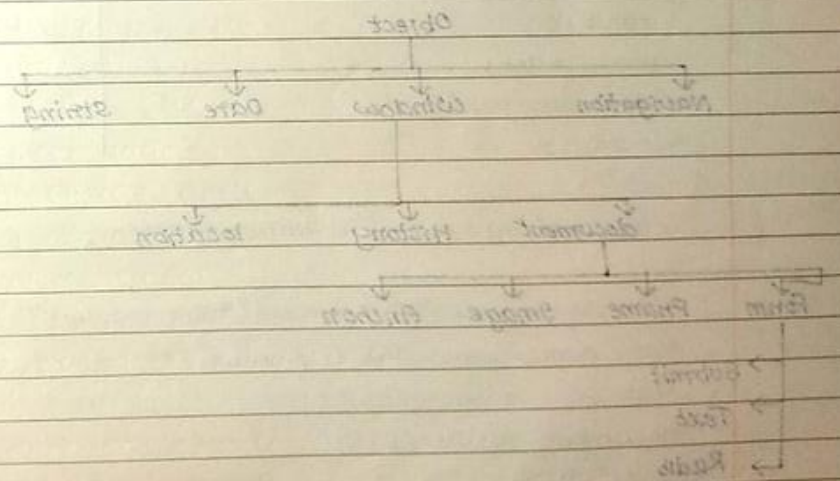
11. xml text node, with data = "Ashish" or "Post text"

12. xml element end, name second element.

13. xml element end, name root element

Example:-

```
<?xml version="1.0"?>
<RootElement ABC="value">
<FirstElement> SomeText </firstElement>
<SecondElement> XYZ="Hello" > Pretext
<inline> Inlinetext </inline> Post text.
</SecondElement>
</RootElement>
```



22 Aug. 2018

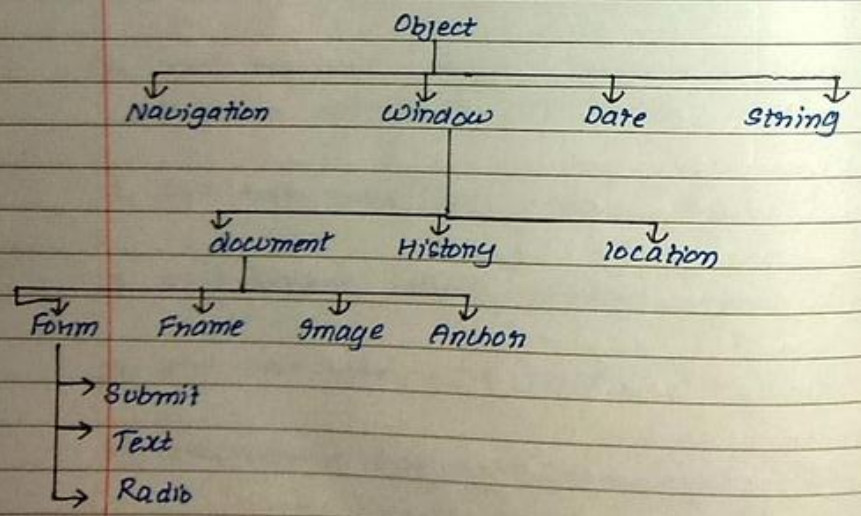
JavaScript :-

```
<script language = "JavaScript">
    .
    .
    .
</script>
```

Example:-

```
<body>
<script language = "JavaScript">
 document.write ("Hello");
</script>
</body>
```

JavaScript Object Hierarchy Models :-



- Java script consider everything as an object. and all object have property, methods & events.

1. within <head>
2. within <body>
3. with both
4. with External file.

```
<script language = "JavaScript" >
 var a=5 ;
  var b=6;
  var s= a+b;
  document.write (s);
  </script>
```

23 Aug 2019

Bitwise Operator:-

AND, OR, XOR, Left Shift, Right Shift
& | ^ << >>

Dialog Box:-

1. alert()
2. prompt()
3. confirm()

Display a Number:-

```
<script language = "Java script">
  var a=5;
  var b=6;
  var s=a+b;
  document.write(s);
</script>
```

By default

```
<script language = "JavaScript">

var num1 = parseInt(prompt("Enter num1"));
var num2 = parseInt(prompt("Enter num2"));
var s = num1+num2;
document.write(s);

<script>
```


Odd or Even Program using Javascript :-

```
<script language = "JavaScript">
 var num = parseInt(prompt("Enter the no."));
 if (num % 2 == 0)
 {
    document.write("Even");
 else
   {
    document.write("Odd");
   }
 }
 </script>
```

Factorial Program Using Javascript:-

```
<script>
 function show()

 var, i, no., fact;
 fact = 1;
 no = Number(document.getElementById("num").value);
 for (i=1; i<=no; i++)
 {
  fact = fact *i;
 }
 document.getElementById("ans").value = fact;
 }
 </script>
```

→ 3*7+5-3

```
<script>
var num = eval("3*7+5-3");
document.write(num);
</script>
```

Ans:- WAP to Square with in a Function

```
<html>
<head>
<body>
<script language = "JavaScript">
var a = parseInt(prompt("Enter the no"));
function square(a)
{
  a = a*a;
  document.write(a);
}
square(a)
</body>
</head>
</html>
```

↑ or

Program:

```
<html>
<title> function </title>
<head>
<script language = "JavaScript">
function xyz(a)
{
  return a*a;
}
</script>
<body>
```



```
<script language= "JavaScript">
var a = parseInt(prompt("Enter num"));
document.write(xyz(a));
</script>
</body>
</html>
```

24 Aug 2018

1 alert():-

```
<html>
<head>
<button id = "OK" Onclick = abc()>
OK
</button>
<script>
function abc()
{
  alert("Hello");
}
</script>
</head>
<body>
</body>
</html>
```

2. confirm :-

```
<html>
<head>
<button id="ok" Onclick = xyz() >
   OK
</button>
<script>
function xyz()
{ if
   (confirm("OK Pressed"))  { a="ok" }
   else      a="cancel"
      document.write(a);
}
</script>
</head>
<body> </body>
</html>
```

Armstrong Program

```
<html>
<head>
<script>
<
```

27/08/2019

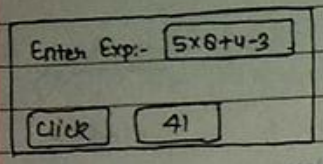
Enter no1 :
Enter no2 :
Multiply
No. is 15

ZOOM Page No. 49

#

```
<html>
< head > < title > For Exp </title>
< script language = "Javascript">
 - function mul()
{ var a = document.fi.n1.value;
 - var b = document.f1.n2.value;
 - var c = a*b; alert(" The multiply a : " + c);
 - </script> </head>
<body>
<form name = "f1">
 Enter No.1 <input type = "text" name = "n1"><br>
 Enter No.2 <input type = "text" name = "n2"><br>
 <a href = "" Onclick = "mul()"> multiply </a>
</form>
</body>
</html>
```

#




```
<html>
<head> <title> for Exp </title>
< script language = "JavaScript" >
 function Expression ()
<button id = "ok"  Onclick = "click" ()>
{
Enter expression < input type= "text"  name = "n1"><br>
 onclick = "
```

```
<html>
<head> <title>  Expression </title>
< Body>
< form name = "f1" >
 Enter expression : < input type = "text"  name = a1 ><br>
<button  onclick = "Evaluate()" > click <button>
 <input type = "text"  name = a2 > <br>
</ form>
</ body>
</ html
```

```
<script language = "javascript" >
   fun. Evaluate
   {
    var a= document f1 a1 .value;
    var b= eval (a).;
      b= document. f1 . a2 . value =b ;
</script>
```



```
<html>
< head>
<title>  Expression <title>
<script language = "Javascript">
 function Evaluate ()
{
  var a = document . f1 . n1 . value;
  var b =  evaluate (a);
  document . f1 . n2 . value = b;
</script>
</head>
</body>
<  form name = "f1">
Enter expression : <input type = "text" name = n1> <br>
<button  onclick = "Evaluate ()" > click </button>
<input type = "text" name = n2> <br>
< /form>
</body>
</html>
```

28/08/2018

Try and Catch :-

The Try statement allow you to define a block of code to be tested for error while it is being executed.

The Catch statement allow you to define a block code to be executed if an error occur in the try block.

Example:

```
<html>
<head>
<title> try and Catch </title>
<script language = "JavaScript">
 try
   {
     window.alert (" welcome GrNIOT");
     document.write (" Hello");
   }
 Catch (abc)
   {
     document. write (abc);
   }
 </Script>
</head>
<body> <h1> About CS </h1>
</body>
</html>
```

O/P - Definition type error Text is not function (If we write

Throw :-

- The throw statement allow you to create a custom error. [user make its own cond.
- Technically you can throw an exception (throw an error)
- Error may be in string or number.

Example:

```
<html>
<head>
<title> throw </title>
<Input ID = "demo" type = "text" >
< Input type = " button" Onclick = "xyz()"> Result
<P id = "message"> </p>
<script language = "JavaScript" >
  function xyz()
 {
   Var x;
   x= document.getElementById ("demo").value;
 try {
      if (x== "    ") throw "Empty";
      if (x<5) throw "low";
      if (x>10) throw "high";
      if (is NaN (x)) throw " Not a number";
      }
Please input a number b/w 5 to 10
Catch (abc)
 {
   document.get Element By Id ("message").inner HTML
                = " Input is " abc;
 }
if (x>=5 && x<=10)
```

```
{
document.getElementById ("message");
inner HTML= " Input is right";
}
```

29|08|2018

this Keyword :-

Example (i)

```
<script>
var Student= new Object();
Student.id =12;
Student_name= Ashish;
document.write( student.name);
</script>
```

(ii)

```
var Student = {
firstName = " Ashish" ;
lastName = " Ojha";
fullName = " Ashish Ojha";
fullName : function ()
{
return, Student.firstname +
        Student.lastname ;
}
// var res = Student.fullName ();
// document.write (res );
var Student2 = Student ; // Create object
document.write( student2.fullname);
```

or

```
return this.firstName +
       this.lastName;
```


This keyword is a reference variable that refer to the current object.

30|08|2018

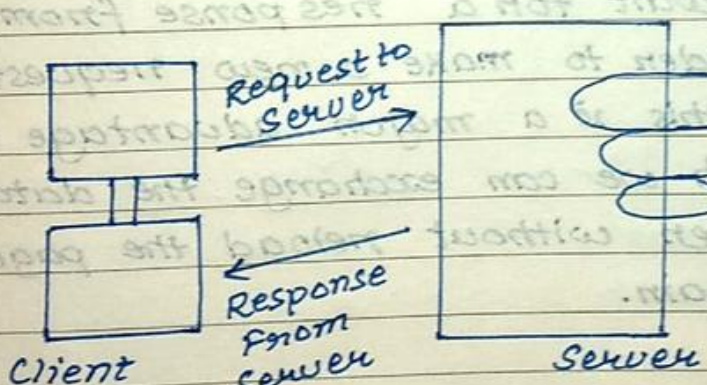
31/08/2018

Introduction to AJAX:-

1. AJAX stands Asynchronous, Javascript and XML.
   Asynchronous means that we are exchanging data to/from the server in the background without having to refresh the page again and again.
2. It is a group of interrelated web development technique used on the client side to create interactive web application.

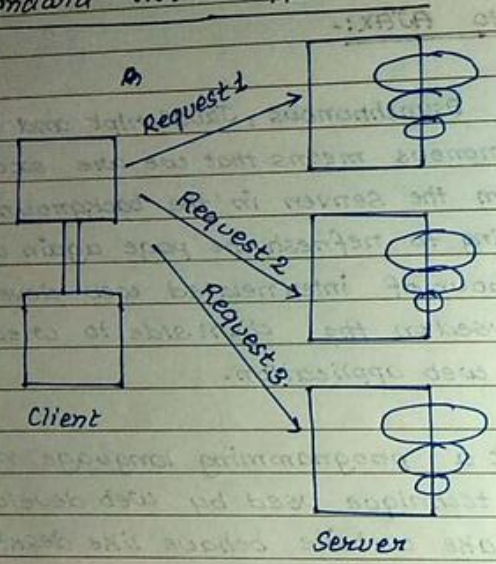
3. AJAX is not a programming language rather AJAX is a technique used by web developer in order to make website behave like desktop applications.

Standard Client Server Application:-



These are standard client server application where client has to wait for the response from the server to execute to other task on the webpage.
So these are Static Application.

Standard AJAX Application :-



Multiple Request are happening concurrently in the same time on client does not have to wait for a response from a server in order to make a new request.

So this is a major advantage of AJAX by which we can exchange the data from the server without reload the page again and again.

Example of Application Using AJAX :-

Google Map, Youtube, Instagram.

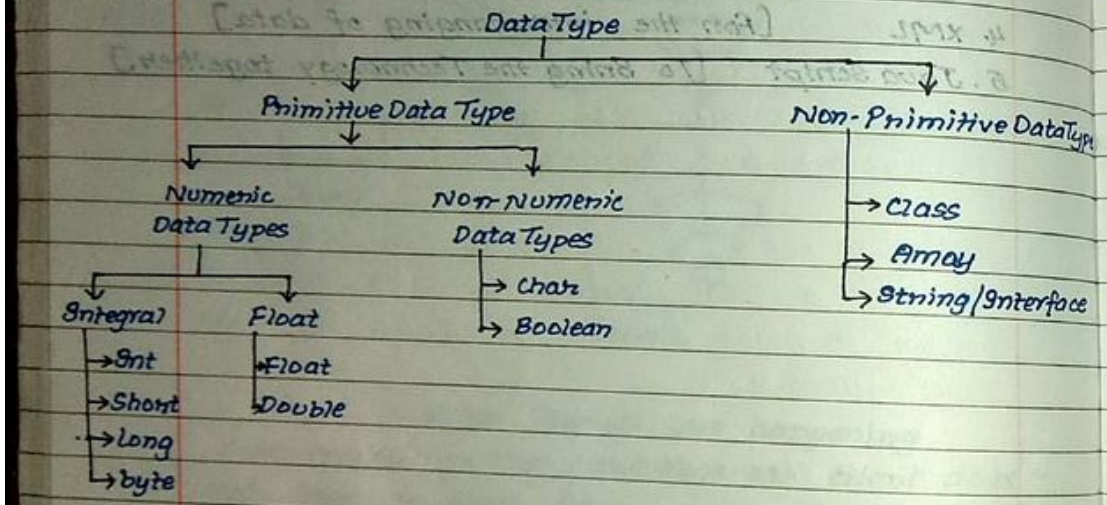
What Technology comes in AJAX ?

1. HTML
2. CSS } for presentation
3. DOM [for dynamic display and interation with data]
4. XML [for the interchanging of data]
5. Java Script [To Bring the Technology together]

18 Sept. 2018

Unit-1

DataTypes:-



In Java -

1. **int** = 4byte [Range is -2^{31} to $2^{31}-1$]
   32 bit Signed
   It gives by default zero value.

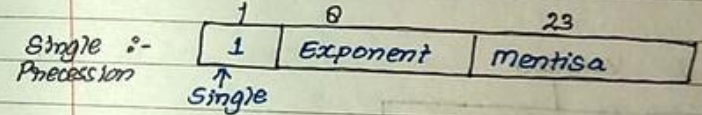
2. **Short**:- 2byte
   Range is [-32768 to 32767]
   16 bit Signed

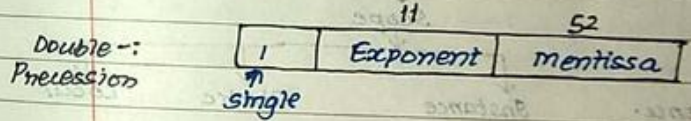
3. **Long**:- 8byte
   64 bit Signed 2's Compliment integer
   Range is [-2^{63} to $2^{63}-1$]

4. **byte**:- 1byte
   Range is [-128 to 127]
   8 bit Signed

Float:- 4byte Range is [-1.7e37 to +1.7e38]
32 bit Representation
Single Precession

Double:- Range is [
64 bit Representation
Double Precession.





Char:- 2bytes in Java

17 Sept. 2018

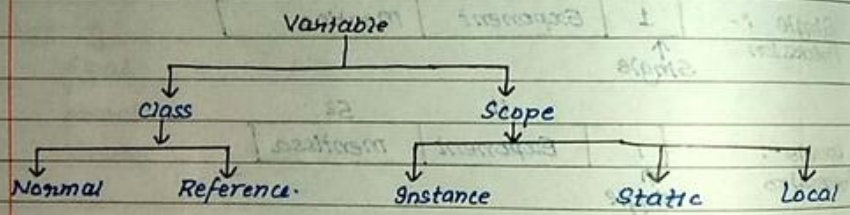
Variable :-

1. Class
   (i) Normal / primitive
   (ii) Referenu
2. Scope
   (i) Instance
   (ii) Static
   (iii) Local

int x; ← (i)
String; ← (ii)
arr[]; ← (ii)

- Instance variable defined just after class.
- if Instance variable is not defined then its bydefault value is 0.
- Static variable cannot access Instance variable

Variable :-



Variable is nameof reserve area allocated in memory.

Variable is name of memory location.

1. Instance Variable :-

- Valueof variable is vary from object to object.
- Cannot be access from static area directly.
- Object notation use for refering instance variable from static area.
- It should be declare with in the class directly.

Example :-

```
Class ABC
{
int x=10; // instance variable
public static void main string s[];
ABC a= new ABC();
   a.x;
```

18 Sept. 2018

2. Static Variable :-

- It can used refer the common property of an object.
- It can be Access by class name.
- Makes program memory efficient.
- Value is not vary from object to object.

College
CS IT

```
Class College
{
Static int count = 0;
College()
{
Count++;
System.out.println(count);
}
Public Static void main (String S[])
{
College cs1 = new college(),!
College cs2 = new college(),!
College IT1 = new college(),!
}
}
```

Without using static

Static Method :-

1. Static method is just like Static Variable.
2. It also belong to class rather then object of class.
3. It can access static data member and can change the value of it.

```
class abc
{
int i;
static String name="cs";
static void change()
{
  name = "cs class";
}
void display()
{
  System.out.printx(name);
}
public static void main (String s[])
{
  change();
  abc a= new abc();
  a.display();
}
}
```


2. Static Block :-

1. Static Block is used to initalise the static data member.
2. It is executed before main method at the time of data loading.
3. It can also execute without main method.

Example :-

```
Class abc
{
  Static()
  {
    System.out.println("CS gpd");
  }
  Public Static void main (String S[])
}
```

3. Local Variable :-

1. Local variable are declareany block and method. 2. Initilization is not mandatory.
3. We can directly access and initalize local variable without any object or class name.

20 Sept 2018

Constructor :-

21 Sept 2018

Inheritance :-

- It is a mechanism of deriving from Sub class to Super class.
- In inherit all instance variable & method defined by Superclass & it also add its own unique elements.

Type of Inheritance :-

1. Single Inheritance [Fig-1]
2. Multilevel Inheritance [Fig-2]
3. Hierarchial Inheritance. [Fig-3]

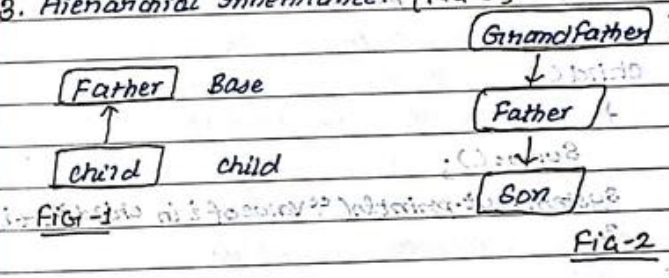


Fig-1

Fig-2

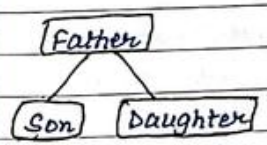


Fig-3

Single Inheritance in Java :-

```
Class Parent
{
    Static int i=10;
    Parent()
    {
        System.out.println("value of i in Parent class",+i);
    }
}
Class child extend parent
{
    child()
    {
        Super();
        System.out.println("value of i in child", +i);
    }
}
```

Example

```
Class main
{
public Static Void main (String [] s)
{
    child b= new child()
}
}
```


multilevel Inheritance :-

```
Class Gnand parent
{
    Static int i=18;
    Gnand parent()
    {
        System.out.println("value of i in Gnand
                                parent class", +i);
    }
}
Class parent extend gnand parent
{
    parent()
    {
        Super()
        System.out.println("value of i in Parent class",
                                +i);
    }
Class child extend parent
{
    child()
    {
        Super();
    System.out.println("value of i in child",+i);
    }
}
```

24 Sept 2018

Main method :-

This is the name which is configure inside JVM.

```
Class demo
{   To call by JVM from anywhere
    Public Static void main ( String args [] )
    {   without creating object also JVM has to call this method.
    }
}
```

main method would not return any thing to JVM.

Command line argument

1. Whether class contain main method or not and whether also main method is declared acc. to requirement or not these thing would not be check by Compiler. At run time, JVM is responsible to check these things.

2. ~~If Even~~ Even though above syntax is very strict or compulsory the following changes are acceptable.

(a) Instead of Public Static we can take Static public. The order of modifier is not important.

(b) We can declare String [] in any acceptable form as
String [] args
String [] args
String args []

(c) Instead of args we can take any Java identifier.


```
class demo
{
  public static void main (String args[])
    {
      int a=5 , b=10 , s;
      s=a+b;
      System.out.println ("The sum is= +s);
    }
}
```

Scanner Sc = new scanner (System.in);

```
Class add
{
  public static void main (String [] args)
    {
      int a,b,s;
      Scanner sc= new scanner( System.in);
      System.out.println ("Enter 1 No");
      a= sc.nextInt ();
      System.out.println ("Enter No");
      b= Sc.nextInt();
      s= a+b;
      System.out.println ("The sum is ="+s)
    }
}
```

25 Sept 2018

Array :-

1. Single Dimensions [1D Array]
2. Multi Dimensions [2D Array and 3D Array]

(1) 1D-Array :-

- **declaration :-**

```
int a[];
int []a;        (recommended)
int []a;
```

→ This method recommended bcoz name is clearly seperated from datatypes.

Note :- we can not declare the size of array at the time of declaration.

- **Creation :-**

```
int []a;        // declaration
a = new int[5]; // Creation
```

(i) declare the size at the time of creation.

(ii) Every array in Java is an object only. Hence we can create array by using new operator.

// Creation and declaration in a single line.

```
int a[] = new int[5];
```


Note :- Both are Array

```
int []a,b;
int  []a,b;
int a[],b;
```

a[] → Arry
b → Simple

(2) 2D-Array :-

- **declaration :-**

```
int a[][];
```

- **Creation :-**

```
int [][]a = new int [3][3];

a[0] = new int[3];
a[1] = new int[2];
a[2] = new int[4];
```

26 Sept 2018

Package:-

- Package is a collection of related classes.
- Package are nothing but more than the way we organised files into different directories according to their functionality, usability as well as categories they should belong too.
- Files in one directory [or Package] would have different functionality from those have another directory.
- Packaging also help us to avoid class name collision when we use the same class name as that of others.
- The Benefits of package reflects the easy of maintenance organisation and increase collaboration among developers.

Run - java - gniot.HelloGniot.java

Note:- 1. We can have only one public class in a single java file.
2. Name of the file should be same as the name of public class.

- Package Access method -
  (i) Fully Qualified
  (ii) Import Statement

Pack1
A
B

Pack1.A obj = new Pack.A();

27 Sept 2018

Interface:-
- Interface contain behaviour that are class implements.
- Interface is a collection of abstract method.
- Interface just specify the method declaration [public and abstract] and can only contain field [public, static, final].
- Interface is used to implement multiple inheritance in java.
- files are stored in .java extension.

Example.

```
interface i1
{
  public void m1();
  int i = 10;
}

interface i2.
{ public void m1();
  int i=20;
}

⇒ class Interface1 implement i1, i2
{ public void m1()
  {
    S.o.Pln();
  }
  public static void main (String [] args)
  {
    System.out.println("Hello");
    interface1 obj = new interface1();
    obj.m1();
  }
}
```

S.o.Pln(i1.i).
S.o.Pln(i2.i)

Function Overriding and Function Overloading :-

1. Overriding :-

1. Whatever method parent has by default available to the child through inheritance. If child class not satisfy with parent class implementation then child class is allowed to define that method based on its requirement, this method is called function overriding.

2. The parent class method which is overriding/ridden is called overridden method & child class method which is overriding method.

3. In overriding method resolution always take care by JVM based on runtime objects and Hence overriding is also considered as runtime polymorphism or dynamic or ~~early~~ late binding.

Example :-

```
class P
{
 public void property ()
{ System.out.println ("Cash+Gold+Land");
} public void marriage ()
{System.out.println(" Subhalakshmi") ;
}
```



```
}
 class C extends P
{ public void marriage ()
  {
   System.out.println ("Isha/Madusri");
  }
}

class test
{
 public static void main (String [] args)
  { P O1 = new P ();
     O1.property ();
     O1.marriage ();

    C O2 = newC ();
     O2.marriage ();
   P O1 = new C ();
     O1.marriage ():
  }
}
```

2. Over Loading :-

1. If two method of a class have the same name but signature that are not same, then the method name is called overloading.

2. In function overloading method resolution always take care by compiler base on reference type. Hence overloading is also consider as compile time polymorphism or static or Early binding.

3. In Java, we can declare multiple method with same name, but different arguement type, such type method are called overloading methods.

Example

```
abc();
abc (int i);
abc (float f);
```

Pattern

```
class P
{
 pv m, (int i)
 {
  S.O.P. ("int arg");
 }
pvm, (float f);
 {
  SoP. ("float a");
 }
}
```


01 Oct. 2018

Exception Handling :-

↓

Mechanism ⇒ Runtime Stack Mechanism.

Exception: Exception is a problem that arise during the execution of the program.

OR

An unexpected, unwanted event that disturb normal flow of a program that disturb is called exception.

Ex. User input, invalid data, File not found

Exception Handling :-

It doesnot mean repairing an exception, it is rather defining alternative way to continue rest of the program normally.

OR

It is a task to maintain normal flow of the program.

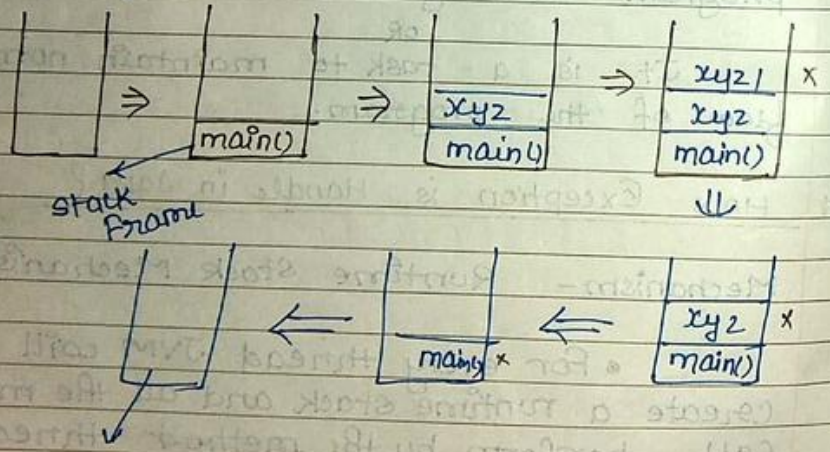
Ques # How Exception is Handle in Java ?

Mechanism - Runtime Stack Mechanism

- For every thread JVM will create a runtime stack and all the method call perform by the ~~method~~ thread will be store in the stack.

Example.

```
class test
{
Public Static Void main (String [] args)
   {
     xyz();
   }
Public static Void xyz()
   {
     xyz1();
   }
   PSV xyz1()
   {
     System.out.println("Hello");
   }
}
```

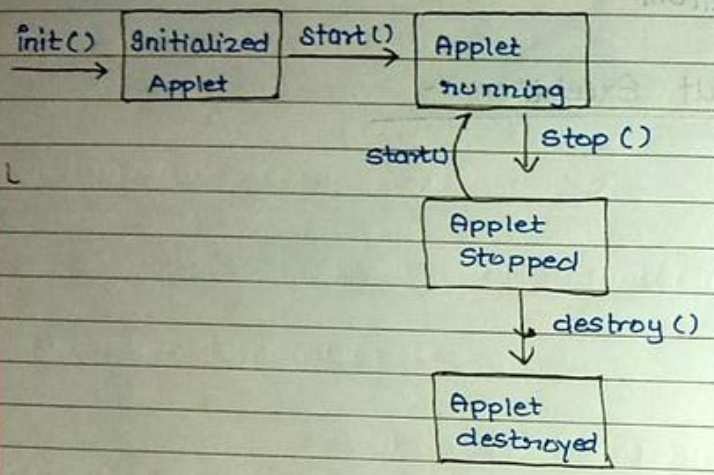


(1) Exception Name
(2) Description
(3) Location.

Default Exception :-

08 Oct. 2018

imp # Applet Lifecycle :-



Java.Applet.Applet *

Applet :-

1. An applet is a Java program that run in a web browser.
2. No main method required.
3. Applet are designed to be embedded with in an html page
4. An applet is a Java class that extend the java.applet.Applet class

Applet Life cycle :-

The Java.applet.Applet class is lifecycle method and java.awt.component class provide one lifecycle method for an applet.

Java.applet.Appletclass :-

For creating any applet "java.applet.Applet class" must be inherit. It provide four lifecycle method of applet.

1. public void init() :-

It is used to initalise the applet. It is call only once.

2. public void start() :-

It is used to start the applet. It is call after the init() method orfunction or browser is maximised.

3. public void stop() :-

It is used to stop the applet. It is call when applet is stop or Browser is minimized.

4. public void destroy() :-

It is used to destroy to display the applet. It is call only once.

Java.awt.component class :-

public void paint() method.

Public void paint() :-

It is used to paint the applet and its provide graphic class object that can we use drawing oval, rectangle, etc.

Ques # Who is Responsible to manage a lifecycle of an object ?

Ans Java plugin software.

Examples of Applet :-

(i) In Java file :- [xyz.java]

```
import java.applet.*;
      OR          or
                Applet.
import java.awt.Graphics;
                or
                *
public class xyz extends Applet.
 {
   public void paint (Graphic g)
     {
       g.drawing ("HELLO GINIOT");
     }
 }
```

(ii) In HTML [a.html] :-

```
<html>
<title> Applet </title>
<applet code = xyz.class>

</applet>
</html>
```

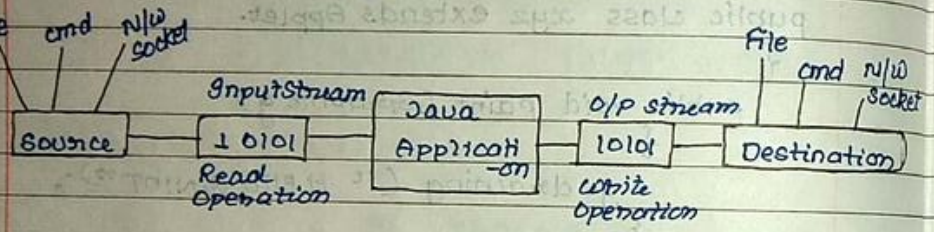
Run method :- appletviewer a.html

09 Oct. 2018

- I/O Stream concept to make I/O operation fast

I/O Stream :- • It is also known as Filehandling
• java.io package is used.

Stream:-Stream
- System.out (output stream)
- System.in (input stream)
- System.err (error stream)



• Stream is a sequence of data which is composed of byte.

Important methods in Input Stream class:-

1. int read() throws ioexception:- It is used to read a byte from the input string.

2. int available() throws ioexception:- It return the number of byte available in the input.

3. Void close() throws ioexception:- It close current input ~~stream~~ stream.

Important Methods in Output Stream class:-

1. int write() throws ioexceptions:- It is used to write byte to current output stream.

2. int write (byte[]) throws ioexceptions:- It write array of byte.

3 void close() throws ioexceptions:- It close current input stream.

4. Void flush() throws ioexceptions:- It flush (clear) the current stream or output stream.

Example of Input Stream class :-

```
import java.io.*;
class file1
{
Public static void main (String [] args)
 {
  FileInput Stream in file = null;
    try
      {
       infile = new file Input Stream("a.text");
       int b;
       while (b= infile.read() != 1)
         {
          System.out.println ((char) b);
         }
          infile.close();
       }
    }
}
```


Example of Output Stream Class :-

```
import java.io.*;
class file 2
{
 Public Static Void main ( String [] args)
  {
   file Output Stream ofile= null;
       byte b [] = {'A', 'B'};
   try
     {
      ofile = new File Output Stream ("a2txt");
      ofile . write(b1);
      ofile . close();
     }
   catch ()
     {

     }
  }

}
```

10 Oct. 2018

Example-1:-

```
Public class xyz
{
 Public static void main (String [] args)
 {
  String s1 = new String ("Computer");
 int i = s1.indexOf ('m');
  System.out.println ("Index is" +i);
 }
}
```

O/P = 2

Example-2:-

```
Public class xyz
 {
    PSVm (String [] args)
    {
      String s1= new string ("computer");
      int i =s1 . index of ('m', 3);
      S.O.P. ("Index is" +i);
    }
 }
```

Output:- is (-1)

Example-3

```
Public class xyz
 {
  psvm (String [] args)
  {
    String s1= new string ("computer");
    int i = s1. index of ("put", 2);
    S.O.P ("Index is" +i);
  }
}
```

Output is 3

imp #

```
Public class xyz
{
 Public static void main (String [] args)
 {
  String s1 = "Computer";
  String s2 = "Computer";
  String s3 = new String ("Computer");
 S.O.P ("Result1" + (s1==s2));   // True
 S.O.P ("Result2" + s1.equals(s2));  // True
 S.O.P ("Result3" + (s1==s3));   // false
 S.O.P ("Result4" + s1.equals (s3));  // True.
```

Compare String:-

* equals (some string)
* Compare to (String)
* equal IgnoreCase (String, anotherString)

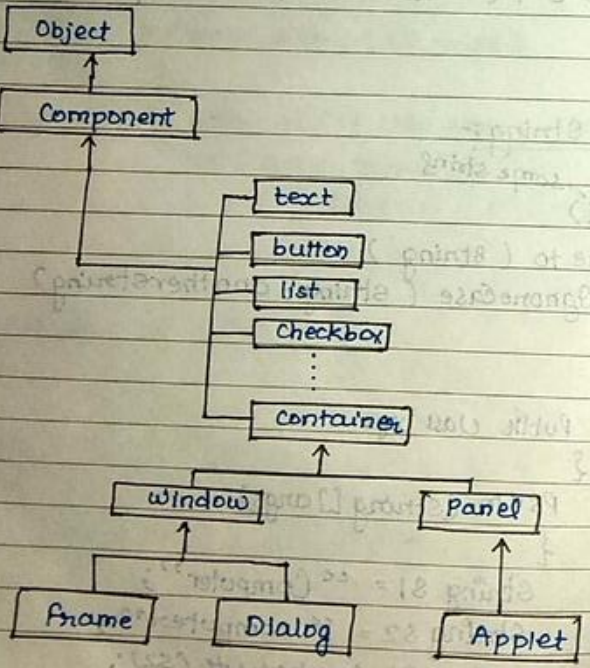
Compareto Ex.

```
Public class xyz
{
 PSVm (String [] args)
 {
  String s1 = "Computer";
  String s2 = "computer";
  Int i = s1. Compare to (s2);
 if (i==0)
  { S.O.P ("Same");
```

```
elseif (i>0)
      { S.O.P ("notsame");
      }
}
  else { S.O.P ("Dictionary");
      }
```

imp # AWT [Abstract Window Toolkit] :-

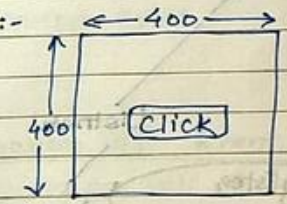
AWT Hierarchy :-




```
import java.awt.*;

public class xyz extends frame
{
    xyz()
    {
     button b = new button ("click");
     add (b); // add button on frame
     b.setBounds (30,100,80,30);
     Set size (400,400);
     Setlayout (null);
     Setvisible (true);
    }
      psvm (String [] s)
      {
       xyz a1 = new xyz();
      }
}
```

It uses as a Readin

O/P :-

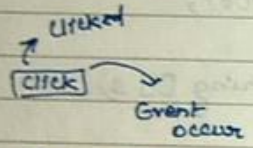


11 Oct. 2018

Important method of Component class :-

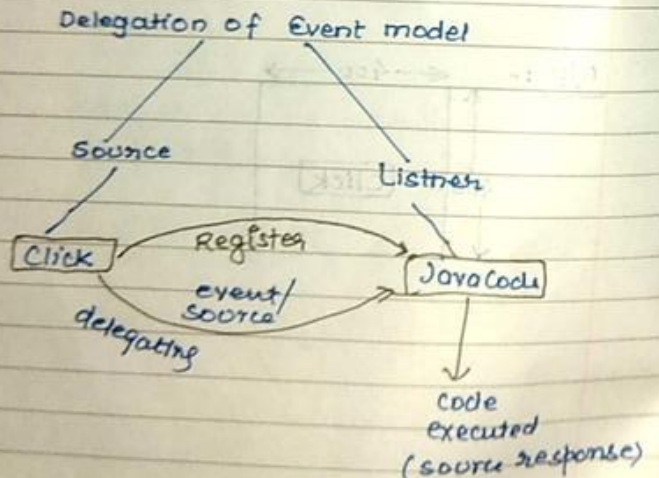
1. public void add (Component c)
2. public void setsize (int width, int length)
3. public void setlayout (layout manager m)
4. public void setvisible (boolean status.)

Java Event Handling :-



Private State - unclicked.

Delegation of Event Model :-



Java Event Handling :-

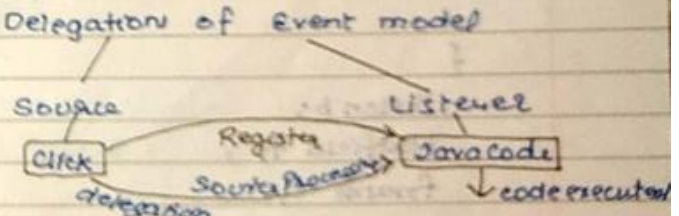
Changing the state of an object is known as event.

For. Eg - Button click, Mouse drag.

The java.awt.event package provide many event classes and listner interface for event handling.

Event Classes	Listener Interface
Action Event	Action Listener
Mouse Event	Mouse Listener
Key Event	Key Listener
Item Event	Item Listener
Text Event	Text Listener
Window Event	Window Listener

imp # Deligation of Event Model :-



Source :-

It is an object on which event occur and it is responsible for providing information of occured event to the handler.

Listeners:-

Listener is also called event handler. It is responsible for generating response to an event.

Note:- • Every source has to be register with the Listener with if state change or any event occur java code is executed

Benefit of Deligation of Event Model:-

It is called deligation event model because the user interface element is available to deligate the processing of any event to seperate piece of code.

Example of Action Listener:-

```
import java.awt.*;
impont java.awt.event.*;
class eventtest extends frame implements ActionListener.
{
Button b;
Textfield ff;
frame t,t';
eventtest()
{
t.newframe("test");
```



```
b.newbutton("click");
tf new textfield (10);
f.add(b); // add component and size
f.add(tf);
b.addActionListner(this); // Registration Lister.
}
```

Event Handling Code —

```
public void actionPerformed (ActionEvent ae)
{
String str = ae.getActionCommand();
tf.setText(str);
}
psvm (string ar[])
{
eventtest e = new event test
}
}
```

12 Oct. 2018

mmp#

Layout Manager:-

→ used in Java at in GUI

that use Alignment ↓ [By default show in Center]

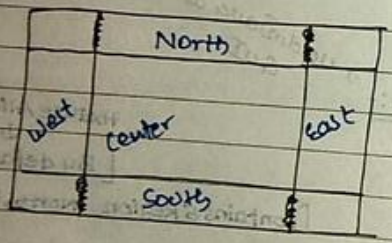
1. Flow Layout
2. Border Layout [Contains 5 Region North, E, W, S, Center]
3. Card Layout
4. Grid Layout

⇒ Border B = new layout

Example Flow Layout :-

```
import java.applet.*;
import java.awt.*;
public class xyz extends Applet
{
Border Flow Layout f1 = new Border Flow Layout();
Button b1 = new Button ("Submit");
Button b2 = new Button ("Stop");
public void init()
    {
        SetLayout (f1);
        add ("North", b1);
    }
}
```

In Border Layout manager the entire region of the window is divided into 5 Regions [North, South, West, East, Center].



2. Border Layout :-

- It is default layout manager of Java for applet and panel.
- It is used to arrange component in aline, one after the another.


```
import java.applet.*;
import java.awt.*;
public class xyz extends Applet
{
 FlowLayout f1 = new FlowLayout();   (Flow Layout (left))
Button b1 = new button ("Submit");
Button b2 = new button ("Stop");
public void init()
  {
    Set layout (f1);
    add (b1);
    add (b2);
  }
}
```

3. Card Layout :-

- It arrange each component in the container as a card.
- Only one card appear on the screen at a time.

```
import java.applet.*;
import java.awt.*;
public class xyz extends Applet
{
 CardLayout f1 = new CardLayout();
 Button b1 = new Button("Card1");
 Button b2 = new Button("Card2");
    f1.setSize(200,200);
    f1.setVisible(true);
    f1.add(b1, "Card1");
    f1.add(b2, "Card2");
    f1.addActionListener(this);
   public class actoPerformed(ActionEvent ae)
   { // use to display nextCard
     card.next(f1);
    }
}.
```


22 Oct 2018

Unit - 3

Inet Address :-

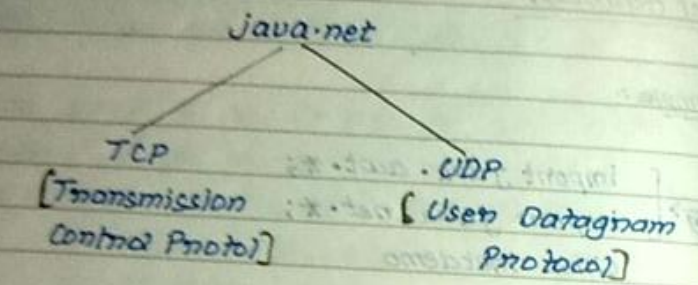
Example :-

Java networking Package →

```
import java.awt.*;
import java.net.*;
class inetdemo
{
 Public static void main (String [] args)
  {
    try
     {
      InetAddress ip = InetAddress.getByName
                          ("www.abc.com");
     System.out.println(ip.getHostName());
                                  www.abc.com
     System.out.println(ip.getHostAddress());
                                  ip Address
     }
    Catch()
    {
     }
   }
}
```

It is used to represent an ip Address and provide method to get the ip of any host name.

Java Networking :-



- TCP is mostly used in Java networking.

Example of URL class

How to/we can parse URL using URL class of Java networking Package?

```
import java.awt.*;
import java.net.*;
class Url_demo
{
Public static void main( String [] args)
  {
    URL u1 = URL ("http://CSE:80/Index.html");

    System.out.println (u1.getProtocol());
    System.out.println( u1.getHost());
    System.out.println( u1.getPort());
    System.out.println (u1.getFile());
  }
}
```

(Protocol, Host name, Port No, File name)

URL Connection Class :-

- It represent the/a communication link b/w URL and its application.
- It can be used to Read/write data to the specify resource refer by URL

Example of URL connection class :-

```
import java.io.*;
import java.net.*;
 Class read_data
  {
   Public static void main (String [] args)
    {
      URL u1 = new URL("http://www.abc.com/a.txt");

      URL Connection u2 = u1.openConnection();
      InputStream is = u2.getInputStream();
      int i;
      while ( i=is.read() != -1)
        {
          System.out.println ((char) i );
        }
    }
}
```

(-1 = end of file; returns ASCII value)

imp

Factory Method:-

- A factory method is one whose return type is similar to class name in which class is present.
- The purpose of factory method is to create an object without using new operator.

Java.awt.GraphicsEnvironment

Example- Public Static GraphicsEnvironment getLocalGraphicEnvironment()

GraphicsEnvironment ge = GraphicsEnvironment.getLocalGraphicsEnvironment();

Rules For Writing Factory Methods:-

1. Every Factory method in Java is static.
2. The return type of the factory method must be similar to class name in which class represent.
3. The access specifier of the factory must be public.

- The init address class has no visible constructor. To create an init address object, we have to use one of the available factory method

- Three commonly used init address factory method are-
1. Static InetAddress getLocalHost()
2. Static InetAddress ByName (String HostName)
3. Static InetAddress [] getAllByName (String HostName)

→ The getLocalHost method simply return the init InetAddress object that represent the local host.

→ The getByName method return an InetAddress for a host name pass to it.
If these methods are unable to resolve the host name, they through an unknown host exceptions.

1. Static InetAddress getLocalHost() throws UnknownHostException.
2. Static InetAddress ByName (String HostName) throws UnknownHostException

→ The getAllByName factory method return an array of InetAddress that represents all of the address that a particular name resolve to.

It will also throw an unknown Host Exception if it cannot resolve the name to atleast one address.

3. Static InetAddress [] getAllByName (String HostName) throws UnKnown Host Exception.

Java Soket Progromming:-

- It is used for communication between the application on different JRE [Java Runtime Environment]

- Java Soket Programming can be connectionless or connection oriented.

- Soket and Server soket classes are used for connection oriented soket programming and Datagram soket and Datagram packet classes are used for connectionless soket programming.

- The Client in soket programming must know two ~~too~~ information
1. Ip address of Server
2. Code number.

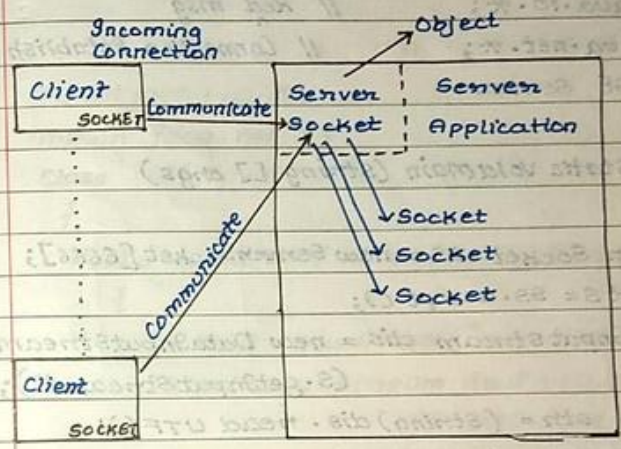
23 Oct 2018

imp#

Socket Class :-

- A socket is simply an end point for communication b/w the machines.
- The socket class can be use to create a socket.

Architecture of Socket Programming:-



Steps in Establishing a TCP Connection Using Socket:-

1. Server —create→ Server Socket Object [Port No]
2. Server —invokes→ accept() method of Server Socket class.
3. Client —wait→ Socket object → Server name [hostname], Port No [6666]
4. Attempt by client to connect.

// Senven Socket Java Code:-

```
import java.io.*;          // Red msg
import java.net.*;         // Connection Establish
Class CSE Senven
{
  Public Static Void main (String [] args)
  {
    Senven Socket SS = new SenvenSocket [6666];   //Serve
    Socket S = SS.accept ();                      // Socke
    Data InputStream dis = new DataInputStream
                           (S.getInputStream());
    String str = (String) dis.read UTF();
    System.out.println (str);
    SS.close ();
  }
}
```

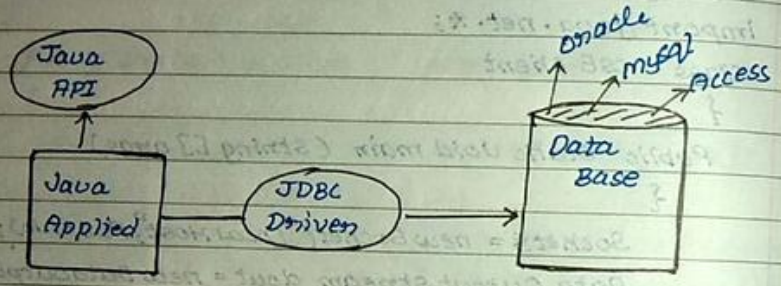

// Socket Client Java Code:-

```
import java.io.*;
import java.net.*;
Class CSE Client
{
  Public Static Void main (String [] args)
  {
    Socket S = new Socket("Local Host", Port No);
    Data Output Stream dout = new DataOutputStream
                              (S.getOutputStream());
    dout.write UTF ("Hello");
    dout.flush ();
    dout.close();
    S.close ();
  }
}
```

24 Oct 2018

Unit-4

JDBC :- Java DataBase Connectivity.



JDBC Architecture

JAva JDBC is ~~the~~/a Java API to connect and exicute query with ~~the~~ database

Steps to Connect to ~~the~~ Database in Java :-

1. Register Driver Class
   * for Name () method

Example :-

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

2. Create Connection Object.

Ex- Connection Con = DriverManager.getConnection ("Path", "UserName", "Pass");

3. Create Statement Object

Ex- Statement stmt = con.createStatement ();

4. Execute the Query

Ex- Result_Set rs = stmt.executeQuery ("Select * from Emp");
while (rs.next())
rs.getInt () + rs.getString (2);

5. Close the Connection object

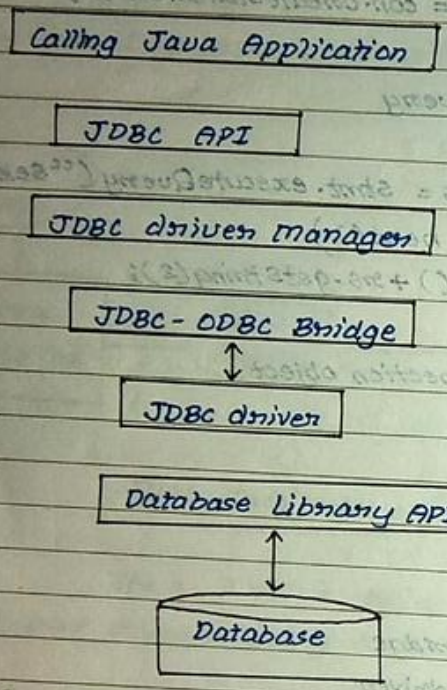
Example- con.close()

JDBC Driver :-

1. JDBC ODBC Bridge
2. Native API driver
3. Network Protocol driver [Middle Ware] Mostly use.
4. Type-4 driver [100% Pure]

Note:- • ODBC = Oracle Data Base Connectivity
• It communicate with Database.

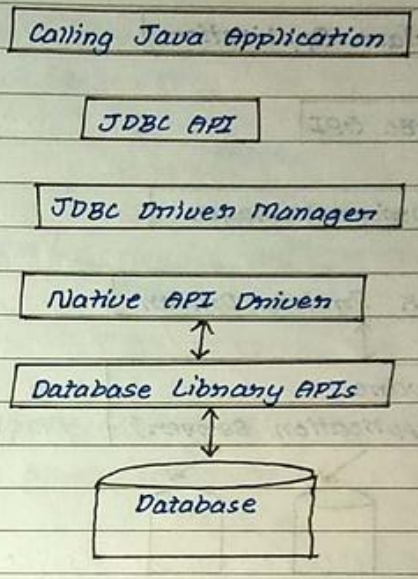
1. JDBC ODBC driver:-



- It is platform dependent
- ODBC is must be setup on install on every client.
- Driver is platform dependent as it make use of ODBC which in turn dependson native library of the underlying O.S. the Java (JVM) is running upon.

Advantage :-

2. Native API Driver :-



- It convert JDBC to Database vender native SQL call.
- It will like/ similar to 1st driver.

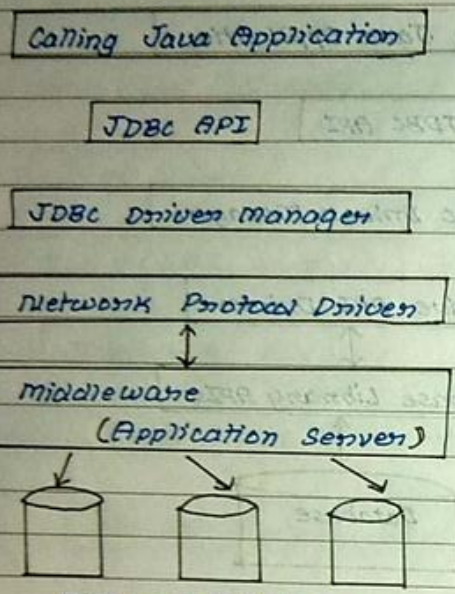
Advantage:-

- As there is no implementation of JDBC-ODBC bridge
- It may be considerable faster than type-1 driver

Disadvantage:-

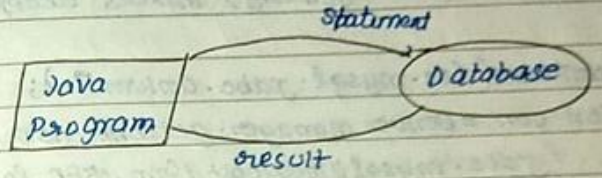
- The vender client library needs to be installed on the client machine.
- Not all Database have a client side library.

8. Network Protocol Driver or middle ware:-



- Translate JDBC to a DBMS independent Network protocol.
- Typically communicate directly with a middleware product which in turn talk to the RDBMS.
- Most flexible Driver type.

1. Statement
2. Prepared Statement
3. Collable Statement



(i) excute Update (String SQL) It will return
(ii) execute query (String SQL)

(i) execute update:-

- It will return the no. of row affected by the execution of the SQL statement.
- Use this method to execute SQL statement for which you expect to get a no. of rows affected.

Ex- Insert, Update, delete.

(ii) execute Query:-

- It will return a set object,
- Use this method when you expect to get a result set

like Select statement

```
import java.mysql.*;
class Statement exm
{
 Public static void main ( String[] arg ) throws exception
  {
    class.forName( "com.mysql.jdbc.driver" );
    Connection con = Driver manager.getconnection
              ("jdbc.mysql//localhost/emp, USER, Pass");
    Statement stmt = con.createstatement();
    String s1 = "delete from Student where id = 101";
    Resultset rs = stmt.executeQuery (s1);
  while ( rs.next())
  {
  System.out.println (rs.getInt ());
  System.out.println ( rs.getstring (2));
  }
    rs.close ();
    con.close();
  }
}
```

- The Statement interface cannot access Parameter.

2. • Prepared Statement:-

It is a subinterface of statement interface.
It is used to execute parameterised query.

30 Oct 2018

#3. Collable Statements:-

- It is used to store the procedure and function of any database like oracle, mysql
- we can have business logic on the database by the use of store procedure and function that will make the performance better bcoz these are precompiled.

Diff. b/w Store Procedure & Function:-

Store Procedure	Function.
• It is used to perform business logic.	• It is used to perform calculation.
• No returntype.	• It is Return type
• It may return 0 or more value.	• It may return only one value.
• It support input and Output parameter.	• It support only input parameter.

H

How to call stone procedure using JDBC :-

```
Create procedure "insert1" (id IN Number,
        nome IN Varchar2)
is begin
insert into student voive (id ,name);
end;
```

Example:-

```
import java.sql.*;
public class StoneProcedure
{
 Public static void main( String [] args ) throws exception.
  {
   Class.forName("Oracle.jdbc.driver.OracleDriver");

   Connection con = DriverManager.getConnection
          ("jdbc:oracle:// localhost /1520/ "system",
                                  "Oracle");

   String Query = " { call insert1 (??) } ";
   Collable Statement Stmt=con.preparecall ("Query");
   Stmt.setInt (1,1001);
   Stmt.setString( 2, "AK");
   Stmt. execute();
   System.out.println ("success");
   }
}
```


Example to Call function Using JDBC :-

```
Create function Sum1 (n1 IN Number, n2 IN Number
return number
 )is
temp number(8);
temp = n1+n2;
return temp,
end;
```

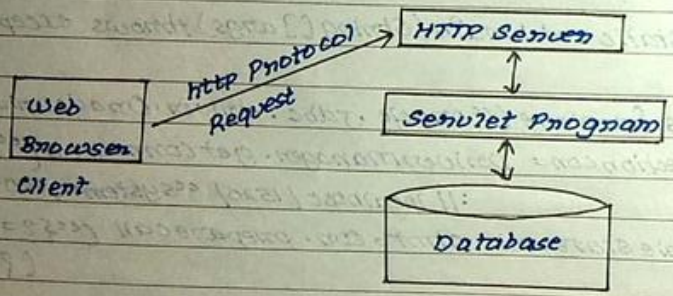
```
import java.sql.*;
public class Function
{
Public static void main(String [] args) throws exception.
 {
  Class.forName ("Oracle.jdbc.driver.OracleDriver");
  Connection con= DriverManager.getConnection ("jdbc:oracle
                :// localhost /1520/ "system", "Oracle");
Collable statement stmt= con.prepare call ("{?=Call Sum1
                                        (?,?)}");

  Stmt.setInt (2,10);
  Stmt.setInt (3,43);
  Stmt.registerOutParameter( 1 TYPE INTEGER);
  Stmt.execute();
  System.out.println (Stmt.getInt());
   }
}
```

Unit-5

Introduction to Servelet:-

- Java Servlet are the programs that run on the web application server

Architecture of Servelet:-



- It act as a middle layer b/w a request from web browser and database application on http server.

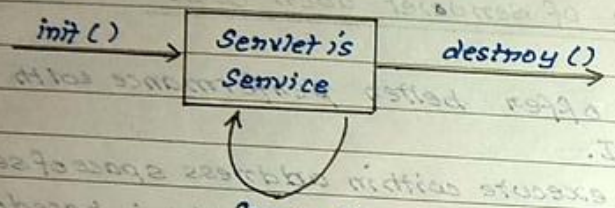
Advantage of Servelet over CGI:-

- Servlet offer better performance with the ment CGI.
- Servelet execute within address space of server
- Servelet is a technology which is based on java that is why servelet are platform independent
- More secure
- We can communicate with other application easily with the help of API like RMI

Use of Servlet:-

- To read the explicit data send by browser.
- To read implicite http request data
- You can easily process the data before sending into the database application or any other application.
- You can send both explicit & implicit data back to browser.

imp # Servlet life cycle :-



Client – Servlet Container [Web browser]

init() :-

- It is called only once.
- Init method called when servlet is created.
- init code is

```
public void init() throws exceptions.
{ // Initialization code
}
```

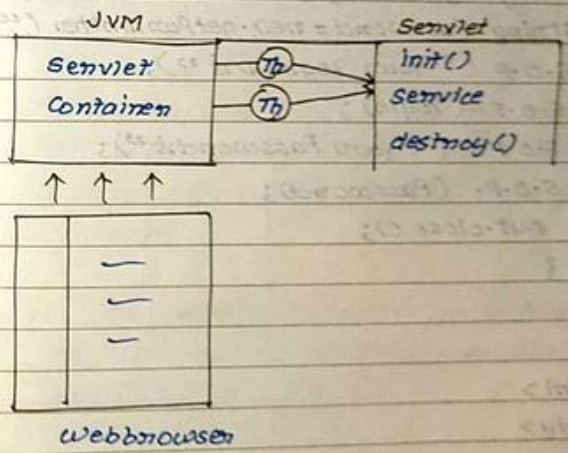
Service method() :-

- It is main method that perform actual task.
- whenever a request is made from the browser the servlet container call the service.
- Service method check the http request type like get and post method and in response call the doGet and doPost method
- Depending on the type of method get, host that is used to call the servlet from the html form.
- public void service (ServiceRequest req, ServiceResponse res)
  { - throws servletexceptions, io Exception
  }

Public void doget (HTTP Request req, http Response res)
throws Servlet Exception, io Exception
{
}

destroy() :-

```
Public void destroy()
{
  // cleanup code
}
```



Architecture of Servlet

Handling HTTP Get Request:-

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
public class servletGetExample extends http servlet
{
 public void doGet(HttpServletRequest req,
                   Http servletResponse res) throws
                   servlet Exception, ioException
   {
    Printwriter out = res.getwriter();
    String login = req.getParameter("login ID");
    String Password = req.getParameter("password");
    S.O.P ("Your login ID is ");
    S.O.P (login);
    S.O.P. ("Your Password is ");
    S.O.P. (Password);
     out.close();
    }
}
```

```
<html>
<body>
<form name = "form1" method = "Get"
      Action = "http://localhost:8080.01 servlet Example">
<P> login id </P>
<input type = "text" > <br>
<P> Password </P>
<input type="text" > <br>
<P> Submit </P>
<input type= "button">
</form>
</body>
</html>
```

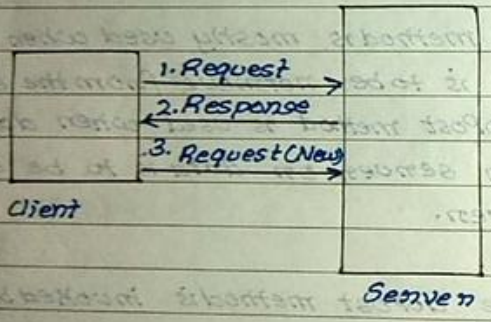

Handling HTTP Post request:-

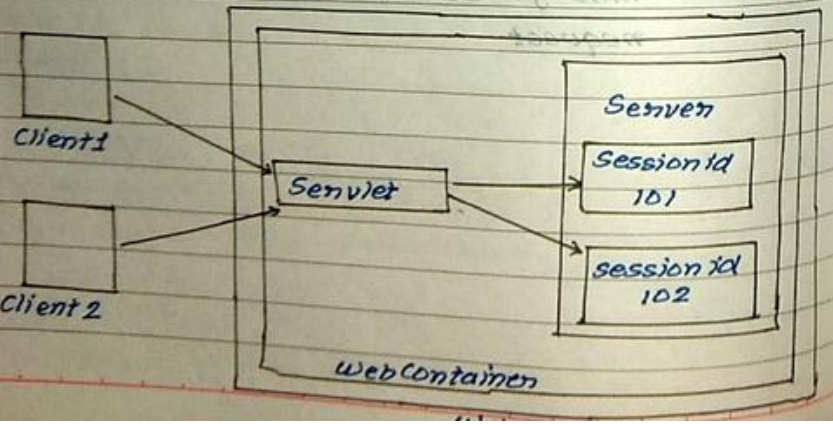
- In doGet method parameters are appended to the URL whereas in doPost method parameters are sent in seperate line in the http request body.
- doPost method is used when large amount of data is required to be passed to the server.
- In doGet method is mostly used when some information is to be retrieve from the server and the doPost method is used when data is to be uploaded on server or data is to be submitted to the server.

Note:- The doPost method is invoked by server through service method to handle http post request.

Session Tracking:- (Apache tomcat server)

- Session tracking is a stateless.
- It is a way to maintain state of any user.
- It is also known as session management in servlet.
- Session tracking is mechanism of tracking the client provide data and making it available to the next request from the same client and this process is continued until the user choose to logout or terminate the session.
- HTTP is a stateless i.e. each request is considered as a new request.





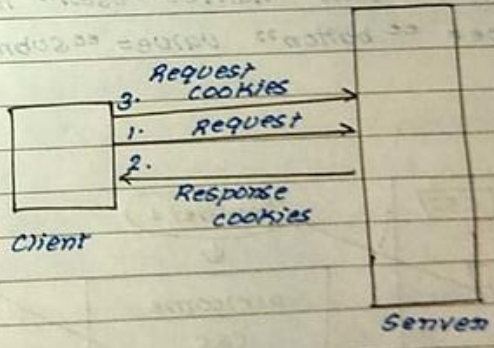
Techniques:-

1. Cookies
2. Hide from field
3. URL Rewriting
4. http session

18 Nov. 2018

1. Cookies:-

- A cookie is a small piece of information that is persist b/w the multiple client request.
- A cookie has a name, a single value and optional attributes such as comment, path, max age, domain qualifier and versionno.



Types of Cookies:-

1. Non-persistent [Valid for single session value only]
2. Persistent [Valid for multiple session]

Step to Set Cookies :-

1. Create new cookie / create cookie object

```
cookie ck = new cookie ("Name", "Value");
```

Ex. `cookie ck = new cookie ("USER", "CSS");`

2. Set max age.

```
ck.setmaxAge(30*60)
```

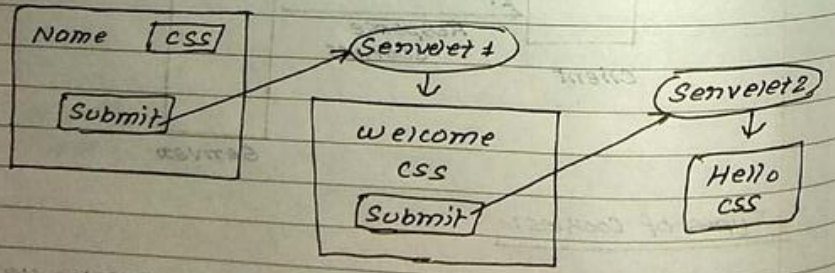
3. Send the cookie into http response header so that it can be stored on the client side.

```
response.addcookie (ck);
```

xyz.html

```
<form name = "f1" > name :
<input type = "text" name = "user" method = "post">
<input type = "button" value = "submit" >
</form>
```

Example :-



index.html

```
<form action = "servelet 1" method = "post">
  name = <input type = "text" name = "n">
        <input type = "button" value = "submit">
</form>
```

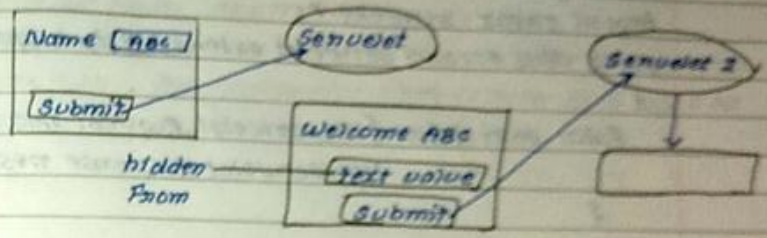
First Servelet.java

```
import java.io.*;
import javax.servelet.*;
public void doPost ( HttpServeletRequest req,
                     HttpServelet Response res ) throws
                servelet Exception, io Exception.
{
  response.setcontent type ("text/html");
  Print writer out = response getwriter ();
  String n = request.getParameter ("user name");
  out.print ("welcome", n);

  // creating cookie object
      cookie ck = new cookie ("username", n);
  // adding cookie in the response
      response.add cookie (ck);
  // creating submit button
  out.print ("<form action = "servelet 2" >");
  out.print ("<input type = "submit" value = "go" >");
  out.print ("</form>");
  out.close ();
}
```

Second servlet.java

```
import java.io.*;
import javax.servelet.*;
Public class second servelet extends
        httpservelet
{
 Public void doPost (http servelet Request req,
                     httpServelet Response res)
  {
  Response.set Content Type ("text/html");
  printwriter out = response.getwriter();
  cookie ck[] = request.getCookies();
  out.print ("Hello" ck[0].getvalue());
  out.close();
  }
}
```


2. Hidden From field :-



index.html

First servelet.java

```
import java.io.* ;
import javax.servelet.*;
Public class firstservelet extends HttpServelet
{
  public void doGet (HttpServeletRequest req,
                     HttpServelet Response res) throws
    {
     response.SetcontentType ("Text/ html");
     printWriter out = response.getWriter();
     String n = request.getParameter ("UserName");
     Out.print ("Welcome" + n);
  // Creating form that have invisible text field
     out.print ("<form action = "Servelet 2">");
     out.print ("<input type = "hidden" name = "uname"
                                value = " + n + ">");
     outprint ("<input type = "Submit" value = "Submit">");
     out.print ("</form>");
     Out.close();
}
```

Second servelet.java

```
import java.io.*;
import javax.servelet.*;
public class second servelet extends Http request
{
  Public void doGet (Httpservelet Request req,
                     Httpservelet Response res)
  {
    try
    {
      response.setContentType ("text/html");
      print writer out = nesponse.getWriter();
      // Getting the value from the hidden field
      String n = request.getParameter ("uname");
      out.print ("Hello" +n );
      out.close();
    }
    Catch()
    {
    }
  }
}
```

3. URL Rewriting:-

In this we append a token on an identifier to the URL of the next servelet, we can send parameter name/value pair using the following format:

URL ? name: value1 & name2 : value2 && ??....

A name and a value is seperated using an equal '=' sign a parameter name/value pair is seperated from another parameter. name/value using '&' sign when user click the hyperlink, the parameter name/value pair coin be pass to the server from a servelet.

15 Nov 18

Java Servlet Page [JSP]:-

- A JSP consists of html tags and JSP tags.
- The JSP pages are easier to maintain than servelet because we can seperate designing and development.
- JSP helps developer to insert java code in html page by making use of special JSP tags.
- JSP starts with <%. and end with %>
- It helps in building webpages that support dynamic content

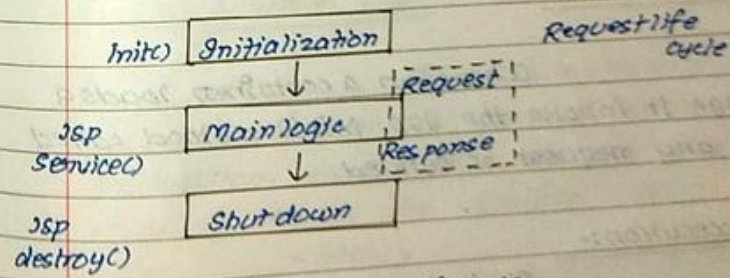


Fig: JSP Life cycle

JSP Life cycle:-

1. JSP Compilation
2. JSP initialization
3. JSP main logic [Execution]
4. JSP cleanup

1. JSP Compilation:-

- Whenever a browser ask for JSP, JSP engine first check to see whether it need to compile the page or not.
- If the page has never beeen compile we can say that the page has been compile into a servelet the JSP compile the page.
- Compilation process invoke 3 main steps.

(i) Parsing the JSP
(ii) Turning the JSP into servelet
(iii) Compiling the servelet.

2. JSP initialization:-

Whenever a container loads a JSP page it invoke the JSP init() method called before any request is served.

3. JSP execution:-

It is main logic.

- Whenever browser request a JSP and the page has been loaded and initialize.

When JSP engine invoke JSP service method()

- JSP service method() takes 2 parameters
  (a) httpServelet Request
  (b) httpServelet Response

4. JSP Cleanup:-

Whenever working of JSP gets completed we have to call the construction and this is removed with the JSP destroy().

16 Nov. 18

JSP Tags:-

1. Scriptlet
2. declaration
3. Expression
4. comment
5. directive.

1. JSP Scriptlet Tag:-

It contain any no of java language statement, variable or method declaration or expression. so html tags and JSP element written must be outside the scriptlet tag.

Eg - <html>
<% Java code %>
</html>

16th NOV 18

* JSP Tags

1. Scriptlet
2. ~~Compitation~~ declaration.
3. ~~Exetution~~ Expression
4. comment
5. directive

1. JSP scriptlet tag - It contain any no. of java language statement, variable or method declaration or expression.
   so html tags and JSP element written must be outside the scriptlet tag.

   ex: <html>
   <% java code %>
   </html>.

2. declaration tag - In this, declaration are to used to dellare one or more variables or method.
   syntax: <% declaration; %>

3. Expression tag - It contains scripting language exp. that is evaluated and converted to a string.
   Syntax: <%= Expression %>

4. JSP comment tag -
   Syntax: <%-- comment --%>

5. directive tag - JSP directive are used to affect the overall structure of the servelet class.
   Syntax: <%@ Directivename attribute = "value" %>

There are 3 directives

i) Page
ii) include
iii) taglib

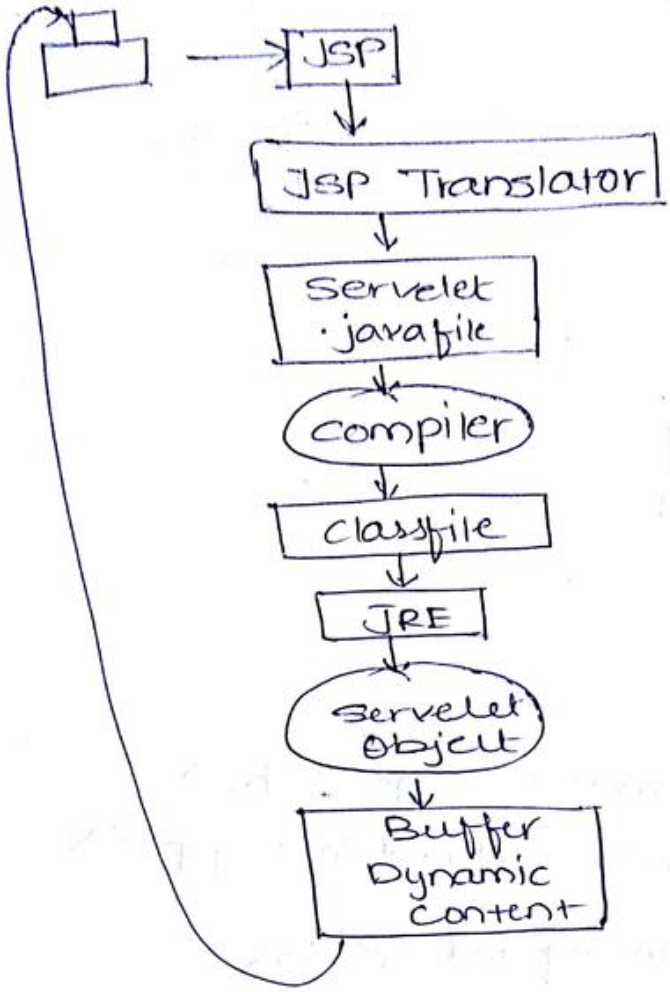
directive name	description
1. Page	Page dependent attributes and these attribute can be like any scripting language that we need to use & we're using error page.
2. include	It include a file during translation time.
3. taglib	It declare a tag library which contains custom actions.

JSP action -

It constructs in xml syntax to control the behaviour of the servelet engine.

syntax: <jsp:action, name attribute="value'>

1. jsp: include - it include a file at the time page is required.
2. jsp: usebean - it find and initialize javabean
3. jsp: forward - it forwards the rqst to a new page
4. jsp: text - it is used to write template text in jsp page



Ex:

index.html

```
<html>
<body>
<form method="post" name="f1" action="form.j
<input type="text" name="fname">
<input type="submit" value="click">
</form>
</body>
</html>
```

form.jsp

```
<body>
<% = request.getParameter("fname") %>
</body>
```

JSP Action Tags

19th NOV 18

i) jsp : include
ii) jsp : forward
iii) jsp : usebeans
iv) jsp : param
v) jsp : plugin
vi) jsp : set property
vii) jsp : get property

i) jsp : include

ex:

index.jsp

```
<h2> this is index page </h2>
<jsp:include page="printdate.jsp"/>
<h2> end section of index </h2>
```

printdate.jsp

```
<% out.print("Today is" + java.util.Calender.
                getInstance.getTime());

%>
```

ii) jsp : forward

jsp:forward action tag is used to forward the request to another resource.

It may be jsp, html.

Tag without parameter

Syntax : <jsp:forward page="Relative URL"/

<%=expression %>" >

Tag with parameter

Syntax:

```
<jsp:forward page="Relative URL/<%= expression %>" />
<jsp:param name="parametername" value="parametervalue
        /<%= expression %>" />
</jsp:forward>
```

ex: index.jsp

```
<h2> first statement </h2>
<jsp:forward page="printdata.jsp" >
<jsp:param name="name" value="google.com" >
</jsp:forward >
```

printdata.jsp

```
<body>
<% out.print("Today is" + java.util.Calender.
        getInstance().getTime() ); %>
<%= request.getParameter("name") %>
</body>
```