UNIT - 2

JAVA SERVER PAGES

It is server site technology for developing dynamic web pages.

this is mainly used for implementing presen-
-tation layer (GUI) path of an application.

A complete JSP code is more like a HTML with bits of java code in it

JSP is an extension of servelets & every JSP page first get converted into Servelet by JSP container before processing the clients request.

JSP has the all the advantages that a servelet has like better performance than CGI (comman gateway Interface) built in session features. It also inherits the feature of java technology like multithreading, exception handling, database connectivity, etc.

JSP enables the separation of content generation prompt; content presentation, which make its a more flexible.

With the JSP, it is now easy for web designers to show the information what is needed.

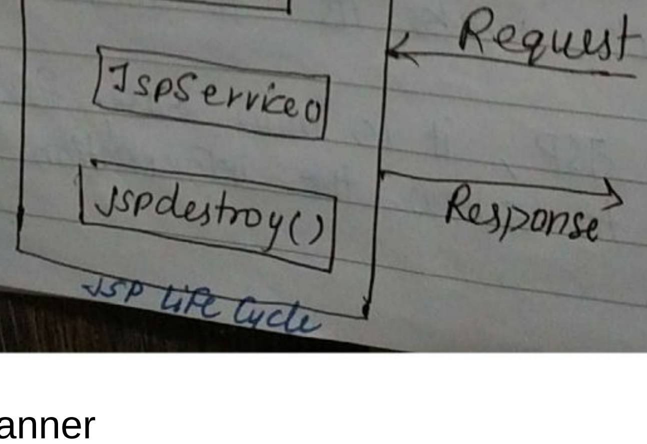
web application programmer can concentrate on how to process / built the information.

LIFECYCLE OF JSP PAGE

1) TRANSLATION
2) COMPILATION
3) LOADING
4) INSTANTIATION
5) INITIALISATION
6) REQUEST PROCESSING
7) DESTRUCTION

JSP Pages are saved with JSP which lets the server know that this is a JSP page and needs to go to JSP lifecycle stages.

JSP pages first gets converted into Servelet & than the corresponding servelet gets processed by server.



JSP DECLARATION TAG

SYNTAX ÷

```
<%! Decleration %>
```

eg ÷ `<%! int i ; %>`

`<%! int i = 10 ; %>` initialisation

MethOD:

```
<%! int sum()
    {
      . . . . .
    }
%>
```

JSP expression

```
<%= "Result is" + Result %>
```

Onload

```
<html>
<head>
<script>
  Function loadimage()
      {
        alert ("image is loaded");

      }
</script><body>
  <img src = "Pic.jpg" onload = "loadimage()"
          width = "100" height = "130">

  </body>
   </head>
  ~~</script>~~
```

On select

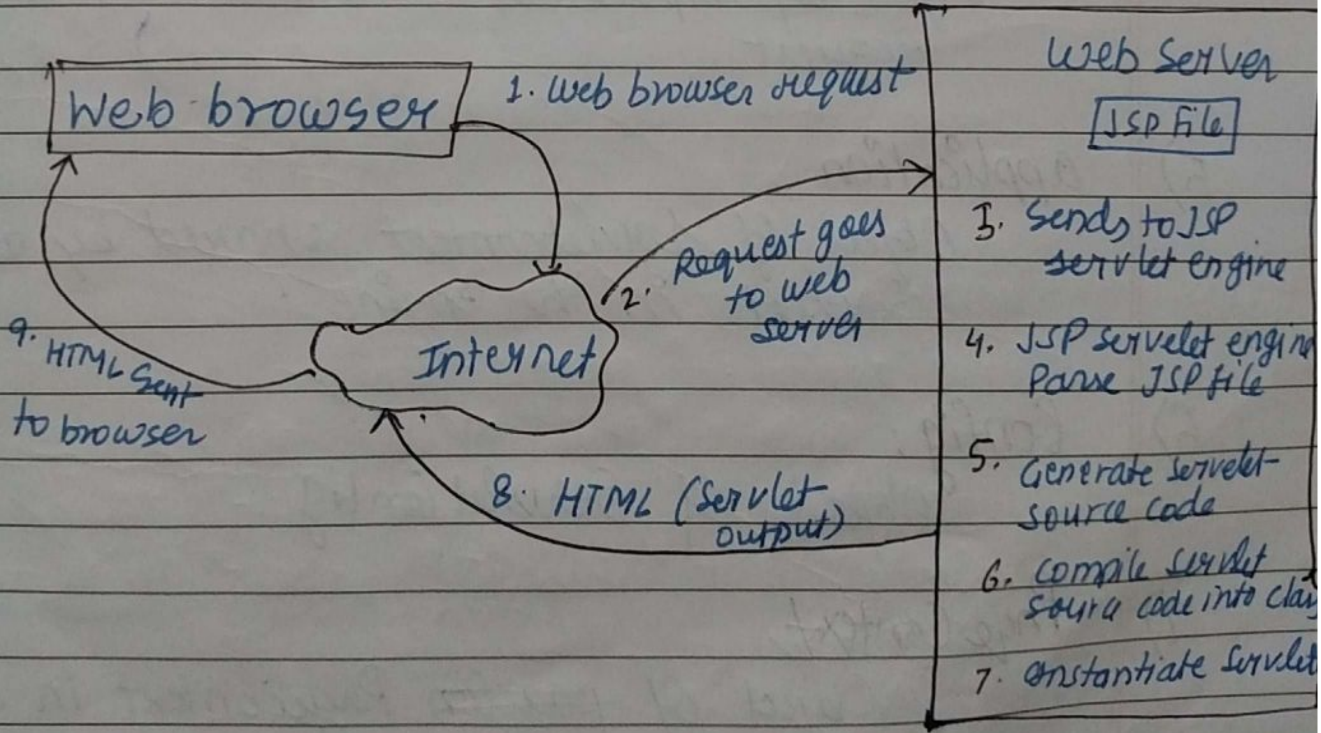
```
<html>
<head>
<script>
  function myalert()
     {


     }
  </script>
  </head>
  <body>
  <input type = "text" id = "myfext" value =
                                     "Some... text"
                  Onselect = " myalert () " >
```

On select

```
<html>
<head>
<script>
    function myselect()
    {
    document . getElementById ("My text") . select();
    }       ↓                ↓
          object           method .
  </script>
  </head>
  <body>
  <button type = "button" onclick= "myselect()">
    </button>
```

31/08/16

JSP ARCHITECTURE



Web browser
1. Web browser request
Web Server
JSP File
2. Request goes to web server
Internet
3. Sends to JSP servlet engine
4. JSP servelet engine Parse JSP file
5. Generate servelet source code
6. compile servlet source code into cla
7. instantiate servlet
8. HTML (Servlet output)
9. HTML Sent to browser

IMPLICIT OBJECTS IN JSP

1) Request
   Object of Http Servelet request
   (Request Parameter,
   Http headers, cookies)

2) Response
   Object of Http Response

3) OUT
   Object of printwriter buffered version.
   ~~When~~ JSP writer.

4) SEssion:
   object of httpsession associated with the Request.

5) application
   Object of Servletcontext shared by all Servlets in the engine.

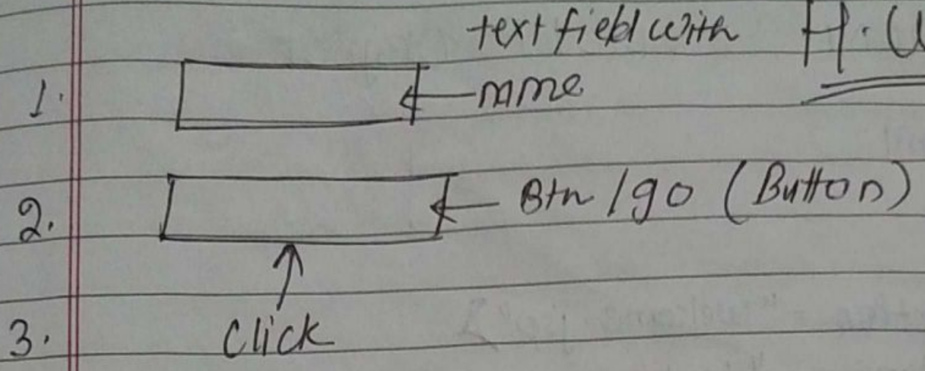
6) Config:
   Object of ServletConfig.

7) Page Context
   • Object of ~~pagecon~~ Pagecontext in JSP for a single point of exces.

8) Page: Variable synonyms for this object.

Scanned by CamScanner

H.W



4. new webpage → welcome amit

```
<html>
<body>
<form action = "welcome.jsp">
 <input type = " text" value = "uname">
  <input type = "submit" value = "go">
    <\form> <\body>

..welcome.jsp.
     <html >
     < body >
     <%. String name = request.get Paramder("Uname")
                                                   ;

          out. print ("welcome"+ name).
        %. >
        </body></html>
```

```
.<html> <body>
<%. String name = request . getParameter ("uname");
                                   new S.B ("name");
                      name1
String Buffer : ~~Class~~ = ~~s[a]. to sb reverse tostring~~
   String s1 = reverse () .to String (name1);
   out. print (" welcome"+s1);
        %. >
```

SESSION IMPLICIT OBJECT

index. html

```
<html>
<body>
<form action = "welcome.jsp">
<input type = "text" name = "uname">
<input type = "submit" value = "go">
</form> </body> </html>
```

welcome. jsp

```
<html>
<body>
<%. String name = request.getParameter ("uname
   out.print ("Welcome" + name);
<a href = "second.jsp> click to go for second
                         jsp page </a>
   Session. Set Attribute ("user", name);
   %>

   </body> </html>
```

Second. jsp

```
<html>
<body>
<%.
    String name = (String) session.getAttribute
                                  ("user");
     out.print ("Hello" + name);
     %> </body> <html>
```

ERROR HANDLING & DEBUGGING

Error handling for jsp pages can be performed in various ways.

a) Error handling from within the page:

For JSP page that require more intricate error handling & recovery, a page can be written to directly handle error from the data bean.
The JSP page can either catch exceptions thrown by the databean or it can check for error codes set within each data bean, depending on how the data bean was activated.
The JSP page can then take an appropriate recovery action based on the error recieved.

b) Error JSP page at the page level

A JSP page can specify its own default from an exception occuring within it through the JSP error tag.
This enables a JSP page to specify its own handling often error.
A JSP page that doesnot contain a JSP error tag will have an error for through to the application level. Error

Error Han

JSP page.
In the page level error JSP page, it must call the JSP helper class to roll back. the current transaction.

Error JSP page at the application level.
An application under websphere can specify a default error JSP page when an exception from with in any of it servelets or JSP pages occurs.

The application level error JSP page can be used as a store level error handler. In a application level error JSP page, a call must be made to the servelet ~~han~~ helper class to rollpack to current transaction.

8/9 index.jsp

```
<form action = " process . jsp">
<input type = "text" name = "n1"/>
<input type = " text" ~~Aand~~ name= " n2"/>
<input type =" Submit" value=" divide" />
 </form>
```

process.jsp

```
<%@ page errorPage = "error" %>

<%
    String num1 = request.getParameter ("n1");
    String num2 = request.getParameter("n2");
    int a = Integer.parseInt (num1);
    int b = Integer.ParseInt (num2);
        int c = a/b;

    out.print ("division of number is "+c);
%>
```

error.jsp

```
<%@ page isErrorPage = "True" %>
  <h3> An exception has occured! </h3>
```

[illegible]

SHARING DATA B/W JSP PAGES

Any real application consist of more than a single page, multiple pages often need acess to the same information & server side resources. When multiple pages crosses the same request.

eg. One page that retrives the data the user asked for and another that displace it, their must be the way to pass data from one page to another. In an application in which the user is asked to provide information in multiple steps, such as an online shopping application, there must be the way to collect the information received with each request and access to the complete set when the user is ready.

Other information & resources need to be share among multiple pages, & all users.

eg: Information about currently logged in users, Database connection pool objects, & Cache objects to avoid frequent database lookups.

Passing Control & data b/w Pages

One of the most fundamental feature of JSP Technology is that it allows for separation of request processing, Buisness logic & presentation, using the MVC (MODEL VIEW CONTROLLER MODEL). As you may recall, the roles of model, view, & controller can be assign to different side of server component

JSP pages are used for both the controller & view roles & the model role is plaed by either or bean or a JSP

Scanned by CamScanner

page.

The different aspects of JSP application can be categorized as:

- Display the form for user i/P (PRESENTATION)
- Validate the input (REQUEST PROCESSING & BUISNESS LOGIC)
- Display the result of the validation (PRESENTATION)

SHARING SESSION & APPLICATION DATA

The request scope makes data available to multiple pages processing the same request. But in many cases, Data must be shared over multiple request.

Imagine a travel agency application, its important to rember the dates & destination enter to book the flight so that the coustomer does not have to renter the information when its time to make hotel & rental car reservation.

This type of information, available only to request. From the same user, can we share through the session scope.

Some information is needed by multiple pages
Independent of ·to the current user is JSP
Supports access to this type of shared
information through the application scope.

Information saved in the application scope
by one page can later be accessed by
another page, even if the two pages were
requested by different user.

eg of information typically share through
the application scope are delta base
Collection cool objects, information about
Currently logged in users, at Cache
objects that avoids unnecessary data base
Queries for data that is same for
all users.

$$\begin{tikzcd}
\text{Init} & \text{Service} \arrow[dl] \arrow[dr] & \text{Destroy} \\
\text{Doget} & & \text{Dopost}
\end{tikzcd}$$

architect by   reference by
Tomcat Server - (Sun microsystem)   James Duncan Davidson

Apache Tomcat is an open source web server and servlet container develobed by Apache software foundation (ASF). Tomcat implements several Java EE specifications including java servlet, Java Server Pages (JSP), Java EL, and Websocket, and provides a "pure Java" web server environment for java code to run it.

eg: Tomcat 4.x was released with

Catalina (a servlet container)
Coyota (an HTTP connector) 1.1 Webserver
Jasper (a JSP engine)
Cluster (Manage large application)
High availability (Facilitate schedulling of System upgrdes without affecting live environmnt)
Web application (Manage user or system based web application & manage session or applications accross network)

Scanned by CamScanner