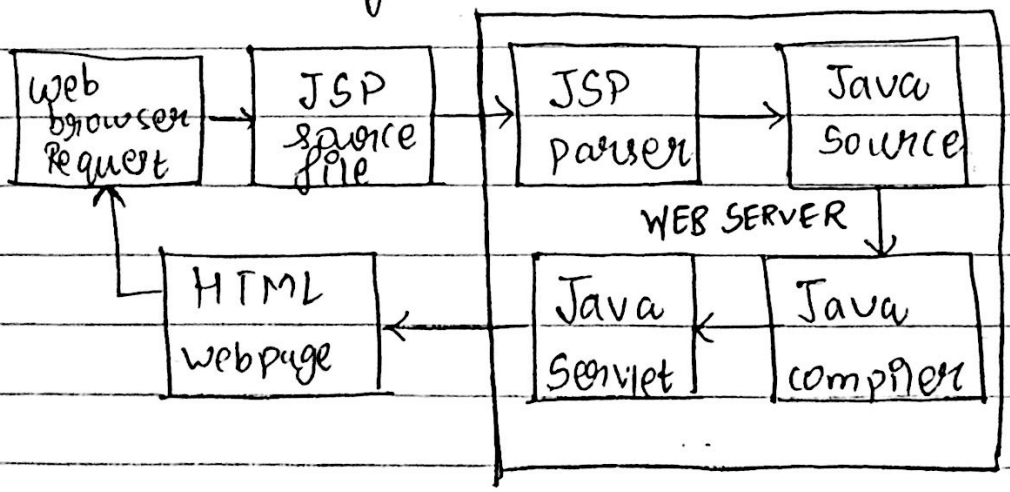
JSP (Java Server Pages)

JSP technology is used to create web applications just like Servlet technology. It can be thought of as an extension to servlet because it provides more functionality than servlet such as expression language. So, JSP page consists of HTML tags + JSP tags.

JSP pages are easier to maintain than servlet because we can separate designing and development. It provides some additional features such as custom tags.



Advantages of JSP over Servlet:

1. Extension to servlet: We can use all features of servlets in JSP. In addition we can use implicit objects, predefined tags and custom tags in JSP.

GOOD WRITE

GOOD WRITE

2. Easy to maintain: JSP can be managed because we can easily separate our business logic with presentation logic.

3. Fast development: No need to recompile and redeploy.

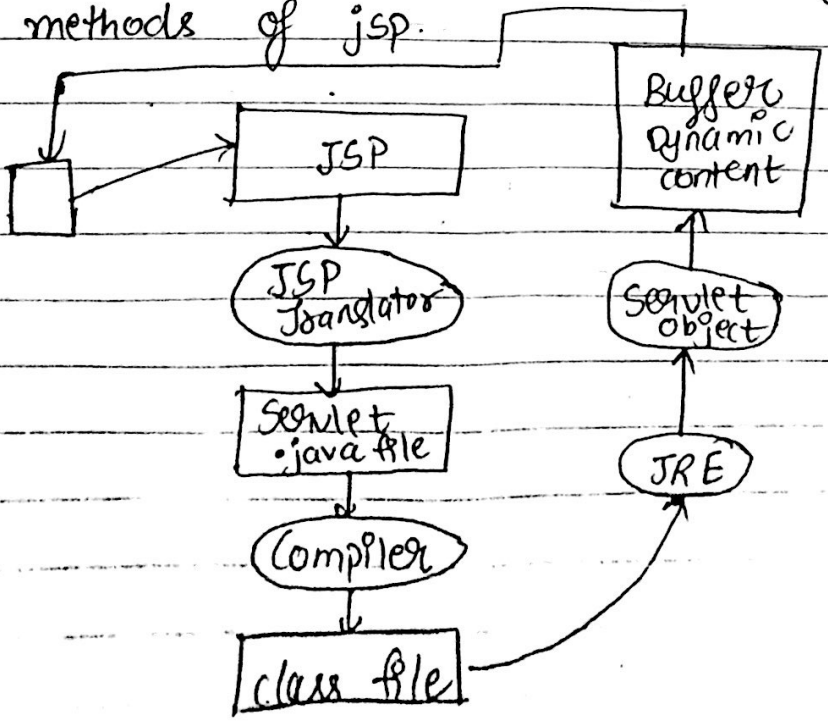
4. Less code than servlet

Life cycle of JSP page:

Phases of JSP page:

1. Translation of JSP page
2. Compilation
3. Class Loading (class file is loaded by class loader).
4. Instantiation (Object of generated servlet is created).
5. Initialization (JSP Init() is invoked by the container).
6. Request Processing (_jspService() method is invoked by container).
7. Destroy (jspDestroy() method is invoked by container).

→ jspInit(), _jspService() and jspDestroy() are life cycle methods of jsp.



JSP page is translated into servlet by the help of JSP translator. JSP translator is a part of web server that is responsible to translate the JSP page into servlet after that servlet page is compiled by the compiler. and gets converted into the class file. Moreover all the processes that happens in the servlet is performed on JSP like initialization, committing response to the browser and destroy.

Creating a simple JSP page:

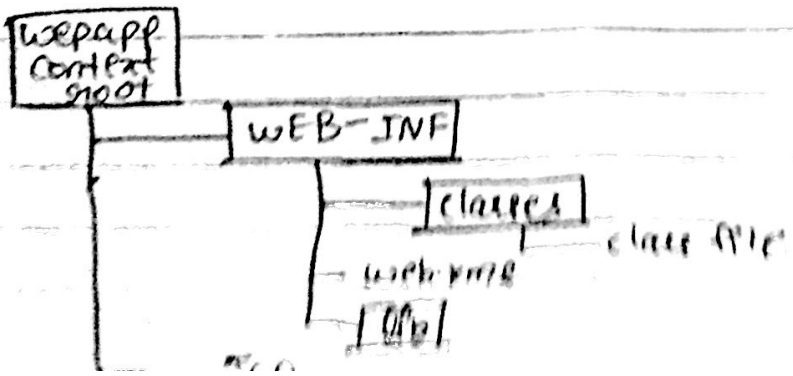
→ Write the HTML code and save it by .jsp extension
→ Put it in a folder and paste the folder in the web-apps directory in apache tomcat to run the JSP page

index.jsp

```
<html>
<body>
<% out.print(2+5); %> → Scriptlet
</body>
</html>
```

Tomcat:

1. Start the server
2. Put the JSP file in a folder and deploy on the server.
3. Visit the browser by URL: http://localhost:portno/contextRoot/jspfile
http://localhost:8888/myapplication/index.jsp.



JSP API
- javax.servlet.jsp
- javax.servlet.jsp.tagext

JSP writer
Page context
JSP Factory
JSP Error

JSP page Component

1. Directives
2. Declarations
3. Scriptlets
4. Expressions
5. Standard Actions
6. Custom Tags

Directives

1. Page Directive

Syntax <%@ page Language = "Java" extends = "<classname>"
import = "<class> or <package>" %>

Attributes: Language = "Java", import = "class
BufferSize = " "
Scope = "REQUEST / PAGE / SESSION / APPLICATION"
And etc.

Page directive is aimed to define certain attribute of a JSP page. For eg:- language of the page in which the page content should be written, which class to be imported so that it can be used within the JSP page.

GOOD WRITE

2. Include Directive

<%@ include file = "<filename>" %>

↳ attribute

This directive is to include the HTML, JSP or Servlet file into a JSP file. This is a static inclusion of the file that is it will be included into the JSP file at the time of compilation and once the JSP file is compiled any changes in the included file will not be reflected.

Declarations

Syntax: <%! Declare all the variables here %>

Scriptlets: <% All your scripts will come here %>

Expressions: <%= expression evaluation & display the variable %>

Include: <jsp:include file="<filename>" />

Forward Directive:

Include <jsp: forward

This will redirect to the different page without notifying the browser.

Custom Tags:

taglib

Syntax <%@ taglib uri = "<tag library Uri>" prefix = "<tag prefix>" %>

a) uri = "<relative path of the tag library uri>

b) prefix = "<tag prefix>"

prefix is alias name for tag library name.

JSP Objects

Objects	of kind
out	JSP writer
Request	HTTP Servlet Request
Response	HTTP servlet Response
Session	HTTP session
Application	Servlet context
Config	Servlet config
Page	Object
Page context	Responsible for generating all other implicit objects.

Out: object is instantiated implicitly from JSP writer class and can be used for displaying anything on the browser.

eg: out.println("JSP");

Request: It is also an implicit object of class HTTP servlet Request and using this object request parameters can be accessed. In case of retrieval of parameters from another form.

eg:- Request.getParameter("Name")

Response: It is also an implicit object of class HTTP servlet response and using this object response parameters can be modified or set.

eg:- Response.setBufferSize("100")

Session: object is of class HTTP session and used to maintain the session information. It stores the information in name-value pair.

GOOD WRITE

2. Include Directive

<%@ include file="<filename>" %>

↳ attribute

This directive is to include the HTML, JSP or Servlet file into a JSP file. This is a static inclusion of the file that is it will be included into the JSP file at the time of compilation and once the JSP file is compiled any changes in the included file will not be reflected.

Declarations

Syntax: <%! Declare all the variables here %>

Scriptlets: <% All your scripts will come here %>

Expressions: <%= expression evaluation & display the variable %>

dc: <jsp:include file="<filename>" />

Foward Directive:

Include <jsp: foward

This will redirect to the different page without notifying the browser.

Custom Tags:

taglib

Syntax <%@ taglib uri="<tag library Uri>" prefix="<tag prefix>" %>

a) uri="<relative path of the tag library uri>

b) prefix="<tag prefix>"

prefix is alias name for tag library name.

GOOD WRITE

eg. Session.setValue("Sname", "ABC");
Session.setValue("age","2");

Application: Object belongs to the class servlet context and used to maintain certain information throughout the scope of the application.
eg:- Set the value of server using this object:
Application.setValue("ServerName", "www.myServer.com")

Steps to develop first JSP application:

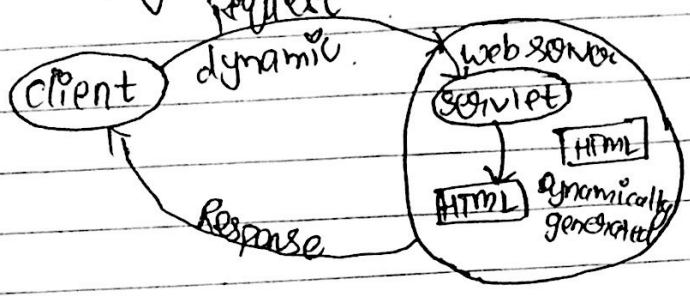
1. Install the Apache Tomcat server.
2. Go to Webapps make a new directory where you want to save all JSP programs.
3. Write a JSP program

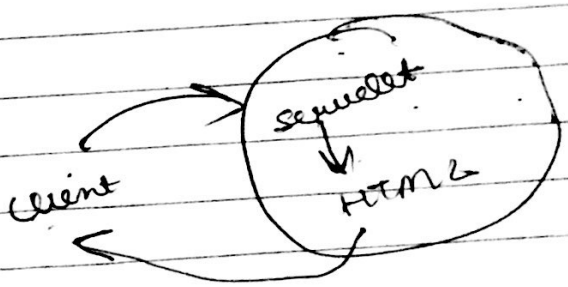
```
<html>
<body>
<H1> ———— </H1>
<font color=red size=7> ———— </font><BR>
<% for(int i=1; i<=4; i++)
    { out.write("Hello <br>");
    }
%>
</body>
</html>
```

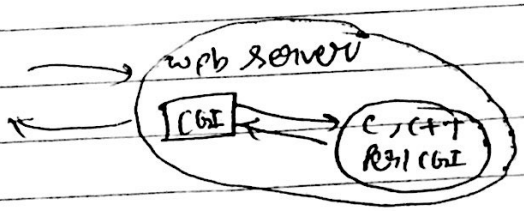
4. Open the IE and write following URL:
http://localhost:8080/myJSP/First.JSP

Servlets

Servlets are Java programs that can be deployed on Java enabled web server to enhance and extend the functionality of the web server.
Servlets can also be used to add dynamic content to web pages.







It provides dynamic content like getting the results of a database query and returning it to the client.
Process and store data submitted by HTML.

Disadvantages of CGI over servlets:
Common Gateway Interface
1. Platform dependency.
2. For each request a process is created.
→ transferring info b/w www server & cgi prog

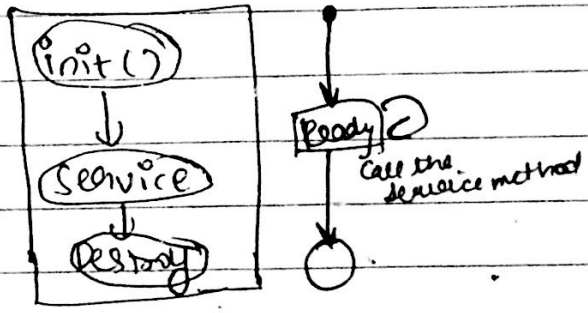
Advantages of servlets:
1. Platform Independence.
2. Performance: Due to interpreted nature of Java programs written in Java are slow but the java servlets run very fast. For any program initialization takes significant amount of time but in case of servlets, initialization takes place very first time it receives a request and remains in memory till time's out or server GOOD WRITE shut downs. After servlet is loaded to handle

a new request it simply creates a new thread and gives service method of the servlet.

3. Extensibility
4. Safe and secure
5. Portable
6. Persistant
7. Efficient

Life cycle of servlet

1) servlet is loaded
   ↳ instance is created
2) init invoked
3) service invoked
4) destroy method



Type of HTTP requests:
GET, POST, Head, Put, Delete Trace

GET: Conventionally Get is used for static resources.

POST:- request is used for dynamic resources.

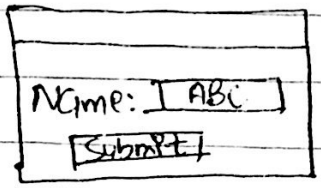
	GET	POST
1)	Request parameters are appended to the URL and send as a part of request header in case of get. Name and values of all request parameters are shown in address bar of the browser.	Request parameters are sent as part of request body. Their name and values are not shown in the address bar.
2)	Maximum data that can be sent as request parameters is determined by the size of header or which has a fixed size	Unlimited data can be sent as request parameters.

GOOD WRITE

How to make a servlet?

```
<html>
<head> First web App </title> </head>
<body>
<form method = "post" action = "First servlet">
<input type = "submit" value = "Submit">
</form>
</body>
</html>
```

GET/

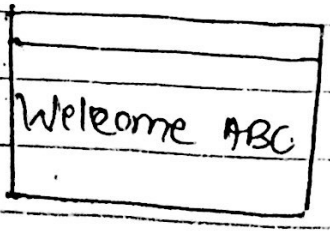


FirstServlet.java

```
import java.servlet.*;
import java.servlet.http.*;
import java.io.*;
class FirstServlet extends HttpServlet
{
        doGET/
    Public void doPost (HttpServletRequest request,
                        HttpServletResponse response) throws
                                ServletException, IOException
    {
        response.setContentType ("text/html");
        PrintWriter out = response.getWriter();
        String name = request.getParameter("txtname");
        out.println ("welcome" + name);
    }
}
```

public void doPost

response.set



response
PrintWriter out = response.getWriter();

① Java scriptlet <html> language statements, variables, methods declaration, expressions
<%. code %>
</html>

② JSP declaration
<%! declaration %> declare

③ Comments
<%-- comment --%>

④ JSP expression
Scripting exp. i.e converted to string.
<%= expression %>

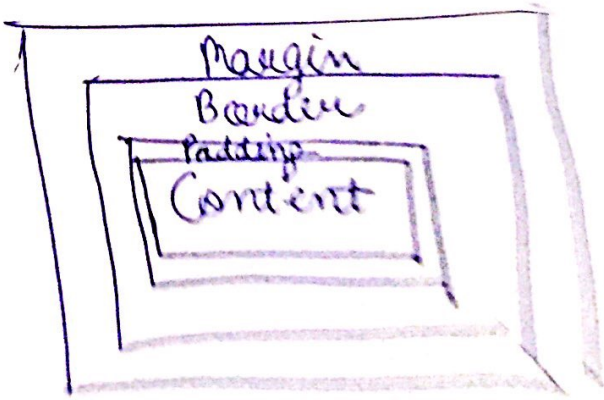
⑤ JSP directives → affect overall structure of servelet class
<%@ directive attribute = "value" %>

CSS selectors

① Element selector
<style type = "text/css">
② Id selector
③ class selector
④ Grouping selector
⑤ Universal selector

CSS box model

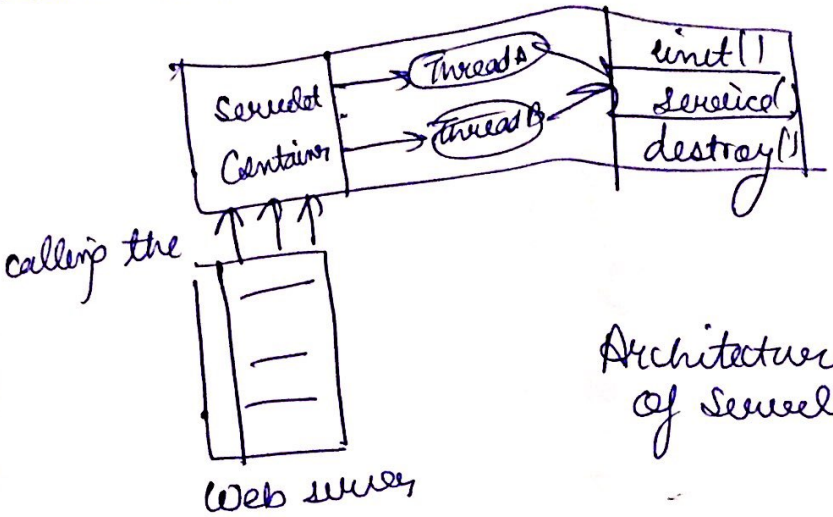
design & layout of HTML element



box model wraps around every HTML document

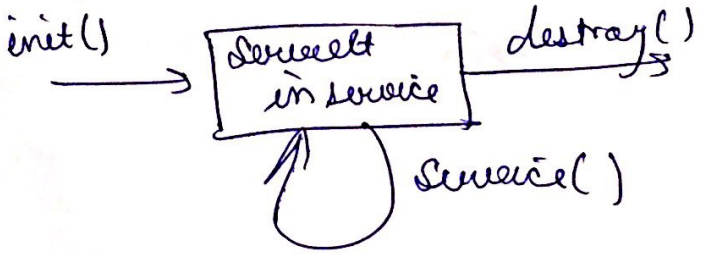
* For every req. in CGI a diff process is created for Servlet for one session if there are multiple req, a single process is created & inside it diff threads are created

Servelet container calls destroy & init automatically



Architecture of Servelet

Life cycle of Servelet



public void doGet(HTTPServeletRequest req, HTTPServeletResponse res) throws ServeletException, IOException)

JSP life cycle

1. JSP compilation
   - (i) Parsing JSP (checking syntax of JSP, verifying all tags & packet)
   - (ii) Turning JSP to Servelet
   - (iii) Compiling Servelet
2. JSP Initialization
3. JSP Execution.
4. JSP cleanup