SERVLET

1.Introduction:

The servlet can be defined in many ways based on context:

- **Servlet** technology is used to create a web application which resides at server side and generates a dynamic web page/dynamic response. To develop web application using servlet, programmer needs more java knowledge or java stuff.
- Servlet is API that provides many interfaces and classes including documentation.
- servlet is program which resides at server side and generates a dynamic response(webpage).
- Servlet is a Thread based technology, if we deploy it at server then container will create a separate thread instead of the process for every request from the client. Due to this Thread based technology at server side server side application performance will be increased.

2.Java servlets and common GateWay Interface Programming:-Originally server-side program handled client requests by using Common Gateway Interface(CGI), which was written in perl,c,c++ or other programming languages.

Although java servlet technology replaces CGI.

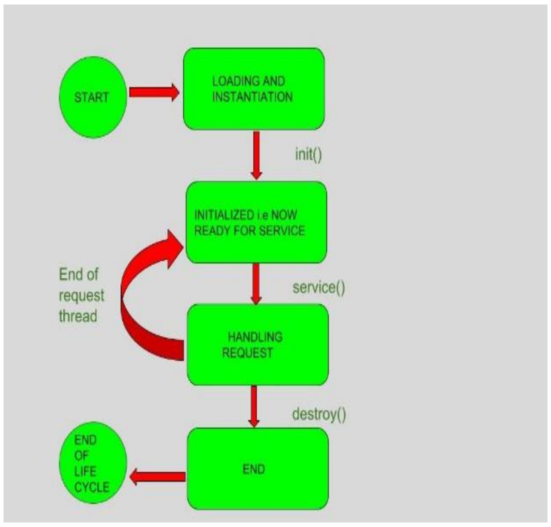
2.1)Benefits of Using java servlet:-

CGI	servlet
1. A new OS process begins each time a request is made to CGI Program.	1.Each request begins a thread to Java servlet.
2. The CGI program is loaded every time when request is made to CGI program.	2.Servlet is loaded only one time into JVM.
3. CGI can't handle cookies	3.Servlet can handle cookies.
4. CGI can not track session	4.Servlet can track sessions.

5. CGI can not read and set Http Headers	5.Servlet can read and set http headers.

3.Anatomy of Java Servlet:- Servlet Container managing the life cycle of servlet. The Servlet life cycle mainly goes through four stages,

- Loading a Servlet.
- Initializing the Servlet.
- Request handling.
- Destroying the Servlet.



A) Loading a Servlet:- The Servlet container performs two operations in this stage .

Loading : Loads the Servlet class byte code into memory. To load the servlet class, Container invokes Class.forname("servlet name").
Instantiation : Creates an instance of the Servlet class. To create a new instance of the Servlet, the container uses the no-argument constructor.

B)Initializing the Servlet:- After the Servlet is instantiated successfully, the Servlet container initializes the instantiated Servlet object. The container initializes the Servlet object by invoking The **Servlet.init(ServletConfig)** method which accepts ServletConfig object reference as parameter.The Servlet container invokes the **Servlet.init(ServletConfig)** method only once.
Before invoking this method,The servlet container create object to servlet config.

C) Request Handling/Request Processing:-After initialization, the Servlet instance is ready to serve the client requests. The Servlet container performs the following operations when the Servlet instance is located to service a request :

- It creates the **ServletRequest** and **ServletResponse** objects. In this case, if this is a HTTP request,then the Web container creates **HttpServletRequest** and **HttpServletResponse** objects which are subtypes of the **ServletRequest** and **ServletResponse** objects respectively.
- After creating the request and response objects it invokes the Servlet.service(ServletRequest, ServletResponse) method by passing the request and response objects.

The **service()** method while processing the request may throw the **ServletException** or **UnavailableException** or **IOException**.

The service() examines the http request type and then calls appropriate request method such as doGet() or doPost().

D)Destroying the Servlet:- The destroy() method is the last method to be called right before the java servlet terminates, such as when an instance of java servlet is removed from memory. You can override the destroy() method with statements that releases resources, such as closing a db connection.

3.1) Deployment Descriptor:- The deployment descriptor is a file located in WEB-INF directory that controls behaviour of servlet and JSP's. This file is called web.xml file which contains XML header,DOCTYPE and web-app element. The web-app element should contain servlet element with 3 sub elements . There are servlet-name,servlet-class & servlet-mapping.

Servlet-name element contains name used to access the java servlet.

Servlet-class is class name of java servlet.

We write the mapping details inside the servlet-mapping tag.

Example:-

```
<web-app>

        <servlet>

              <servlet-name>MyJavaServlet</servlet-name>

              <servlet-class>MyServletClass</servlet-class>

              <servlet-mapping>

                     <servlet-name>MyJavaServlet</servlet-name>

                     <url-pattern>/MyServlet</servlet-name>

             </servlet-mapping>

       </servlet>

</web-app>
```

4. Simple Java Servlet:-The java Servlet is java class that reads request sent from client and resonds by sending information to the client.

```
Import java.io.*;
Import javax.servlet.*;
Import javax.servlet.http.*;
Public class js1 extends Httpservlet{

        Public    void    doGet(HttpservletRequest    request,HttpservletResponse
response) throws servletException, IOException

           {

        Response.setContenttype("text/html");



         PrintWriter out=response.getWriter();

        Out.println("<html>\n"+

                          "<head><title>Java Servlet</title>\n"+

                         "<body>\n"+
                    "<p>My basic java Servlet</p>\n"+
                   " </body>\n  "+
                    "</html>");
          }
     }
```

The java class must extend the Httpservlet abstract class.

The doGet() method is used to interact with reques sent using METHOD='GET' attribute from HTML form.The doGet() method takes two arguments.

The first argument is HttpservletRequest object. It contains incoming information. The incoming data includes explicit data and implicit data. Explicit data is supplied by user. Implicit data is Http information such as request headers.

The second argument is HttpServletResponse object. It contains outgoing data. The outgoing data includes explicit data and implicit data. Explicit data is generated by servlet. Implicit data is Http information such as response headers. The implicit data is generated by servlet/webserver.

The setContentType() method is used to set value for the ContentType Http header information. ContentType identifies type of document that is being sent explicitly.

The Outgoing data is sent using PrintWriter object using println() method. The println() method is used to send explicit data in the format of simple webpage/xml page , if the client is a browser.

5.Reading Data from Client:- The data sent by client is read into java servlet by calling the

getParameter() method of HttpservletRequest object.

Syntax of getParameter() method:

Public String getParameter(String name);

This methods returns String object. The string object contains the value of parameter, if client assigns value to parameter. An empty string object is returned if client did not assign value to parame

ter. Also null is returned if the parameter is not received from client.

Example:-

Index.html

```
  <Form Action="/servlet/myservlets.js2">
        Enter email Address:
   <input type="text" name="email">
  </form>
```

Sample.java

```
   Import java.io.*;
   Import javax.servlet.*;
   Import javax.servlet.http.*;
   Public class  sample extends Httpservlet{
        Public void doGet(HttpservletRequest request,HttpservletResponse
response) throws servletException, IOException
          {
        Response.setContenttype("text/html");
        PrintWriter out=response.getWriter();
        String email;
        Email=request.getParameter(“email”);

        Out.println(“<html>\n”+

                           “<head><title>Java Servlet</title>\n”+

                          “<body>\n”+

                           “<p>My Email Address:”+email+”</p>\n”+

                          “ </body>\n  “+


                           “</html>”);

          }

    }
```

6) Reading HTTP Request Headers:- A request from client contains two components. These are data(explicit data)sent by the client, such email address . The other component is set of HTTP request headers. This request header values are set by client(browser). These values are said to be implicit data.

HttpRequestHeader	Description
1. Accept-Charset used by browser that	Identifies character sets that can be made the request.
2. Accept-Language are used by browser.	specifies the preferred language that
3. Cookie	returns cookies to server.
4. Content-Length are transmitted using	Contains the size of data in bytes that POST method.
5. Accept can be handled by	Identifies the MIME type data that

Browser. Ex:- image/jpeg,text/html,text/plain....etc.

The HttpRequest Header value is read into java servlet by calling the
getHeader() method of HttpServletRequest object.

Syntax of getHeader() method:

Public String getHeader("HttpRequestHeader-name");

It returns String. It contains value or null. If the set of HTTP request headers does not have specified HTTP request header.

Example:

Sample1.java

```
Import java.io.*;
Import javax.servlet.*;
Import javax.servlet.http.*;
Public class  sample1 extends Httpservlet{

    Public void doGet(HttpservletRequest request,HttpservletResponse response) throws servletException, IOException

      {

    Response.setContentype("text/html");

    PrintWriter out=response.getWriter();

    String a,b,c;

    a=request.getHeader("Accept");

    b=request.getHeader("Accept-Language");
```

```
        c=request.getHeader("Connection");
        Out.println("<html>\n"+
                          "<head><title>Java Servlet</title>\n"+
                         "<body>\n"+
                          "<p>Accept:"+a+"</p>\n"+
                         "<p>Accept-Language:"+a+"</p>\n"+
                        "<p>Connection:"+a+"</p>\n"+
                         " </body>\n  "+
                          "</html>");
          }
    }
```

7)Senging Data to client and writing Http Respone Header:- HttpServletResponse object contains outgoing data. The outgoing data includes explicit data and implicit data. Explicit data is generated by servlet. Implicit data is Http information such as response headers. The implicit data is generated by servlet/webserver.

The setContentType() method is used to set value for the ContentType Http header information. ContentType identifies type of document that is being sent explicitly.

The Outgoing data is sent using PrintWriter object using println() method. The println() method is

used to send explicit data in the format of simple webpage/xml page , if the client is a browser.

Response Object Format:

Http version statuscode Message
Set of response Headers
Blank line
Document.

Http Response Header contains a status line, response headers, and blank line followed by Document.

Example to Response Header:

Http /1.1 200 OK
Content-type:text/plain
Blank line
My response

The java servlet can write status code in response header by calling the setStatus() method of HttpservletResponse object.

Syntax: void setStatus(int status code);

This method automatically inserts status code and short message associated with the status code into HTTP response header.

statusCode	ShortMessage
100	continue
200	OK
302	FOUND
400	BAD REQUEST
404	NOTFOUND

There is another method that developer use in java servlet to write HTTP response header. That method is sendError().

Syntax: public void sendError(int statuscode,String shortMessage);

This method is used to notify the client that an error has occurred.

There are number of HTTP response headers that can be set by a java servlet. Each HTTP response header is set from within java servlet by calling appropriate method of the HTTPservletResponse before the java servlet writes response document.

HTTP Response Header	Description
Allow	specifies the request method(GET,POST) supported by server.
Content-language	Indicates the language of document.
Content-length	indicates number of bytes in the message.
Content-type	Indicates the MIME type of response document.
Set-Cookie	Identifies the cookie for the page.

<u>8. Working with Cookies</u>:- A cookie is small of data that is saved on the client machine and can be created by and referenced by Java servlet using java servlet cookie API.

It is a class . It is defined in javax.servlet.http package.

Constructors:-
----

1. Cookie():- creates cookie.
2. Cookie(String name,String value) :- creates cookie with given name and value.

Methods:-
----

1. Public string getName():- It retruns the name from cookie object.
2. Public String getValue():- It return the value from cookie object.
3. Public void setVersion(int no):- It sets the version to cookie.
4. Public int getVerison():- it returns version which was set to cookie.

Q) How is cookie added to response object?

A) Response object has following method to add cookie.
Public void addCookie(Cookie c)

Q) How to get all cookies from resquest object?

A) It has following method to get all cookies from request object.
Public Cookie[] getCookies()

Example:-

One.java

```
 Import java.io.*;
Import javax.servlet.*;
Import javax.servlet.http.*;
Public class  One extends Httpservlet{

      Public void doGet(HttpservletRequest request,HttpservletResponse
response) throws servletException, IOException

         {

      Cookie myCookie=new Cookie("userId","123");

       Response.addCookie(myCookie);

      Response.setContentype("text/html");

      PrintWriter out=response.getWriter();

       Out.println("<html>\n"+

                    "<head><title>Java Servlet</title>\n"+

                   "<body>\n"+

                    "<p>Cookie Written</p>\n"+

                  " </body>\n  "+

                    "</html>");

         }

 }
```

Two.java

```
 Import java.io.*;
 Import javax.servlet.*;
 Import javax.servlet.http.*;
 Public class  Two  extends Httpservlet{

        Public void doGet(HttpservletRequest request,HttpservletResponse
response) throws servletException, IOException

          {

        Response.setContenttype("text/html");
        PrintWriter out=response.getWriter();
        Out.println("<html>\n"+
                        "<head><title>Java Servlet</title>\n"+
                       "<body>\n"+
                        "<h3>Cookie</h3>\n"+
                         "<p>"+
                          Cookie[] cookies=request.getCookies();
                          If (cookies == null)
                          {
                            Out.println("No cookies");
                          }
                         Else
                         {
                           Cookie Mycookie;
                              For(int i=0;i<cookie.length();i++)
                               {
                                    MyCookie=cookies[i];

Out.println(mycookies.getName()+"="+Mycookie.getValue());
                                 }
                           }
                         }
                   }

                      " </body>\n  "+
                       "</html>");
          }
```

```
}
```

9.Tracking Sessions:- A session is created each time a client requests service from java servlet. The java servlet processes the request and responds accordingly, after which the session is terminated.Many times the same client follows with another request to the same java servlet, and the java servlet requires information regarding the previous session to process the request. However, HTTP is stateless protocol, meaning that there is not any holdover from previous session.

The servlet is capable of tracking sessions by using HTtpSession API. In HttpSession Session Tracking Mechanism, to create HttpSession object we will use either of the following methods. These are available in request object.

1. HttpServletRequest.getSession(true);
2. HttpServletRequest.getSession(false);

If the getSession() takes Boolean true as an argument,get session() method returns the current session as session and returns a session object.If current session does not exist, the getSession() method creates as new session and returns a session object.

If the getsession() takes Boolean false as argument,getSession() method returns current session as session object. If the current session does not exist, the getSession() method returns null.

To set an attribute on to the HttpSession object we have to use the following method.

-> public void setAttribute(String name, Object value);

To get a particular attribute value from HttpSession object we have to use the following method. ->public Object getAttribute(String name);

Example:-

Index.html

```
<!DOCTYPE html>
```

```
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
 <form method="GET" action="Servlet">
      Book Name: <input type="text" name="uname"/><br>
      Quantity: <input type="number" name="pwd"/><br>
 <input type="submit" value="OK"/>
  </form>
</body>
</html>
```

Servlet.java

```
package com.raos;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import java.util.*;
import java.io.*;
/**
 * Servlet implementation class Servlet
 */
@WebServlet("/Servlet")
public class Servlet extends HttpServlet {
	private static final long serialVersionUID = 1L;

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
		response.setContentType("text/html");
```

```
                PrintWriter pw=response.getWriter();
                String a;
                String b;
                a=request.getParameter("uname");
                b=request.getParameter("pwd");
                HttpSession s1=request.getSession(true);
                s1.setAttribute(a,b);
                pw.println("<html><body>");
                pw.println("<h2>Session Created</h2>");
                pw.print("<form action='Servlet2'>");
                pw.print("<input type='submit' value='Send Another Request to 
Server'>");
                pw.print("</form>");
                pw.println("</body></html>");
        }

}
```

Servlet2.java

```
package com.raos;
import java.io.IOException;
import java.io.*;
import javax.servlet.http.Cookie;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
 * Servlet implementation class Servlet2
 */
@WebServlet("/Servlet2")
public class Servlet2 extends HttpServlet {
        private static final long serialVersionUID = 1L;
        protected void doGet(HttpServletRequest request, HttpServletResponse 
response) throws ServletException, IOException {
        response.setContentType("text/plain");
        PrintWriter out = response.getWriter();
        HttpSession s2=request.getSession(true);
        out.println("Data Which has been received from session");
        out.print("uname: ");
        out.println(s2.getAttribute("uname"));
        out.close();
        }

}
```

Java Server Pages

1.JSP:-
The JSP is a server side technology. It is used to design web applications in order to generate dynamic response.

For developing single page web applications, we use only JSP technology.

For developing medium and large scale web applications, we use both Servlet and JSP technology.

1.1. Differences:

1) a) Servlet is server side technology.

b)JSP is extension to Servlet. We can use all features of servlet in JSP. In addition to , we can use implicit objects, predefined tags,custom tags and extension languages in jsp.

2) a) In servlet technology, The business logic and presentation logic is mixed. Therefore maintaince of that code is difficult.

b) In JSP technology, we separate business logic with presentation logic. Therefore code can be easily managed.

3) a) If we modify servlet ,then servlet is to be recompiled and redeployed.

b) If we modify jsp, then jsp is not to be recompiled and redeployed because JSP pages are auto-compiled and auto-reloaded.

4) a) If programmer wants to design web application using servlet, Programmer needs a more java knowledge.

b)JSP technology reduces the usage of java code in web-application development.

2.JSP Tags:- The JSP program consists of combination of HTML tags and JSP tags. JSP tags define java code that is to be executed before the output of jsp program is sent to the browser.
A JSP tag begins with <% ,which is followed by java code, and ends with %>
There are 5 types of JSP tags that you will use in JSP program. These are

1. Comment tag:- A comment tag opens with <%-- and closes with --%>, and is followed by comment.

   Ex:-<%-- This is comment --%>
2. Declaration Statement tags:- A declaration statement tag opens with <%! And is followed by java declaration statement(s) that define variables, objects and methods that are available to other components of JSP program. The declaration tag close with %>.

```
Example:-
  <html>
 <head>
        <title> JSP Program</title>
</head>
<body>
   <%!   Int age;//variable declaration
         Float salary=20.5;
         String ename=new String("suku");//created the objects.
         Boolean sum(int a,int b)// Method declaration
           {
                Return a+b;
           }
    %>
 <body>
</html>
```

3. **Directive Tags:-** A directive tag opens with <%@ and commands the JSP virtual engine to perform a specific task. The declarative tag closes with %>.There are 3 common directive tags.

   a)import tag:- It is used to import java packages into JSP program.

   Example:-<% page import=" import java.sql.*";%>

   b)include tag:- It inserts specified file into JSP program replacing the include tag.

   Example:- <% include file="abc.html"%>

   c)taglib:- It specifies a file that contains a tag library.

   Example:- <%taglib url="mytag.tld"%>

d) Expression tags:- An expression tag opens with <%= and is used for an expression statement whose result replaces the expression tag when JSP virtual engine resolves JSP tags. An expression tags close with %>.

Example:-

```
<html>
  <head>
          <title> JSP Program</title>
 </head>
 <body>
    <%!    Int age;//variable declaration
           Float salary=20.5;
           String ename=new String(“suku”);//created the objects.
           Boolean sum(int a,int b)// Method declaration
             {
                   Return a+b;
             }
     %>
 <p>Salary is:<%=salary%></p>
<p>Sum is:<%=sum(10,20) %></p>
  <body>
 </html>
```

e)Scriptlet tags:- A scriplet tag opens with <% and contains commonly used java control statements and loops. A scriplet tag closes with %>

Example:-

```
<html>
  <head>
          <title> JSP Program</title>
 </head>
 <body>
    <%!    Int age=19;
     %>
   <% if (age >=18){%>
         <p> Major</p>
     <%} else {%>
```

```
                    <p> Minor</p>
                 <%}%>
               <body>
              </html>
```

<u>3.Requesting String:-</u> The browser generates a user request string whenever the submit button is selected. The user request string consists of the URL and query String both are separated by question mark(?).

The URL is divided into four parts beginning with the protocol. The protocol defines rule that are used to transfer the request string from browser to JSP program. The protocol are Http,https..etc.Next is host and port combination. The host is IP address or name of server that contains a JSP program. The port number is port that host monitors. Following the host and port is the virtual path of JSP program. The server maps the virtual path to the physical path.

The query string consists of name and value pairs. The pairs are separated by '&'.

Example:-

http://localhost:8080/webprojectname/resoursename(0r)Urlpattern?
Uname="suku"&town="nlr"

<u>4.Cookie</u>:- A cookie is small of data that is saved on the client machine and can be created by and referenced by Java servlet using java servlet cookie API.

It is a class . It is defined in javax.servlet.http package.

Constructors:-

1.Cookie():- creates cookie.
2.Cookie(String name,String value) :- creates cookie with given name and value.
Methods:-

1.Public string getName():- It retruns the name from cookie object.
2.Public String getValue():- It return the value from cookie object.

3.Public void setVersion(int no):- It sets the version to cookie.
4.Public int getVerison():- it returns version which was set to cookie.

Q) How is cookie added to response object?
A) Response object has following method to add cookie.
    Public void addCookie(Cookie c)
Q) How to get all cookies from resquest object?
A) It has following method to get all cookies from request object.
    Public Cookie[] getCookies()

One.jsp

```
            <html>
           <head>
                  <title> JSP Program</title>
          </head>
          <body>
            <%!   String Mycookiename='uname';
                  String Mycookievalue='rock';
                  Response.addcookies(new
Cookie(mycookiename,mycookievalue));
             %>
             <body>
          </html>
```

Two.jsp

```
          <html>
           <head>
                  <title> JSP Program</title>
          </head>
          <body>
            <%!   String Mycookiename='uname';
                  String Mycookievalue;
                  String cname,cvalue;
                  Cookie[] cookies=request.getCookies();
            %>
             <%  for(int i=0;i<cookies.length;i++){%>
                  <p>Cookie Name: <% =cookies[i].getName() %></p>
                  <p>Cookie Value: <% =cookies[i].getValue() %></p>
```

```
                    <%{%>
                    <body>
          </html>
```

5.Session Objects:- A session is created each time a client requests service from JSP. The JSP processes the request and responds accordingly, after which the session is terminated.Many times the same client follows with another request to the same JSP, and the JSP requires information regarding the previous session to process the request. However, HTTP is stateless protocol, meaning that there is not any holdover from previous session.

The Session object is also used to store information called attributes. An attribute can be login information, preferences, or even purchases placed in electronic shopping cart.

To set an attribute on to the HttpSession object we have to use the following method.

-> session. setAttribute(String name, Object value);

To get a particular attribute value from HttpSession object we have to use the following method. -> Object session. getAttribute(String name);

Note:- In JSP, programmer does not need to write statement for creating the HttpSession object.

Example:-

One.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
   <%
         session.setAttribute("scooter",2000);
```

```
    %>
    <h2>Session Created and Data added to Session Object</h2>
    <form  action="two.jsp">
      <input type="submit" value="send Another Request"/>
    </form>

</body>
</html>
```

Two.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>

<%!
   String b="scooter";
%>
<%
  String a=session.getAttribute(b).toString();
%>
<h3>Item Name:Scooter</h3>
<h3> Cost:<%=a%></h3>
</body>
</html>
```

JAVA &XML

1.Generating XML document:- A J2EE application can generate an XML document based on business rules that are encoded into one or more J2EE components. Typically, these J2EE components are either java servlet,JSP,Enterprise Java Beans or java Message Service component.

The java servlet that generates an XML document. The xml document contains a template for book. Only <Titlepage> tag contains text. There are ,of course, many more tags and directives used i
n real-world XML document used for a book, and all those tags contain text.

```
Import java.io.*;
Import javax.servlet.*;
Import javax.servlet.http.*;
Public class  Two  extends Httpservlet{

      Public void doGet(HttpservletRequest request,HttpservletResponse
response) throws servletException, IOException

        {

      Response.setContenttype("text/html");
      PrintWriter out=response.getWriter();
      Out.println("<?xml version=\"1.0\ " ?>");
      Out.println("<book> ");
     Out.println("<titlepage> ");
     Out.println(" J2EE, The complete Reference");
     Out.println("</titlepage >");
     Out.println(" <copywritepage>");
     Out.println(" </copy writepage>");
     Out.println(" <Table of contents>");
     Out.println("</tableofcontents ");
     Out.println(" <preface>");
     Out.println(" </preface>");
     Out.println(" <chapter>");
     Out.println(" </chapter>");
     Out.println(" </book>");

        }
  }
```

2.Parsing XML:- An XML document contains information and tags that identify information. Any application that reads an XML document must parse information in the document in order to access information contained in the document.

Writing an application that parses a document is challenging and time-consuming. However, there are parsers available that are designed to parse an XML document and are accessible through an application programming interface(API). The parser API consists of Java classes that provide easy access to elements of XML document. There are two types of XML parsers.

2.1)DOM
2.2)SAX.

2.1)DOM(Document Object Model):- The DOM parser reads entire XML document and creates tree structure in memory for xml document.

Advantages:-
----
1. The DOM API is very simple to use.
2. It supports read and write operation on XML document.
3. It is preferred when random access to widely separated parts of document is required.

Disadvantage:-
----
1. It is memory inefficient.(consumes more memory because whole XML document needs to loaded into memory)
2. It is comparativey slower than other parser.

2.2)SAX(simple API for XML):- A SAX parser implements SAX API. This API is event based API. The SAX

parser uses an event strategy. A j2EE component that interacts with a SAX parser to manipulate an
XML document must contain event-handler classes that listen for specific events sent by SAX parser as the SAX parser reads the XML document.

Advantages:-

----

1) It is simple and memory efficient.
2) It is very fast and works for huge documents.

Disadvantages:-

----

1) The J2EE component does not have control of parsing once the SAX parser is invoked.