20CS505

Hall Ticket Number:

III/IV B.Tech (Regular) DEGREE EXAMINATION

February,2023 | **Computer Science & Engineering**

Fifth Semester | **Enterprise Programming**

Time: Three Hours | **Maximum:** 70 Marks

Answer Question No. 1 Compulsorily. (14X1 = 14 Marks)

Answer ***ANY ONE*** *question from each Unit.* (4X14=56 Marks)

1. a)	Differentiate Web Server and Application server.	CO1	L3	1M
b)	List out various JSF Core tags.	CO1	L1	1M
c)	What is EJB?	CO1	L1	1M
d)	What is the purpose of WebSocket?	CO2	L2	1M
e)	Define RESTful webservice.	CO2	L2	1M
f)	What are the different types of statements in JDBC?	CO2	L3	1M
g)	List out various session tracking techniques.	CO3	L2	1M
h)	Why do we need JSP technology if we already have servlets?	CO3	L2	1M
i)	What is the full form of JSON?	CO3	L1	1M
j)	The WebSocket protocol supports binary and text based messages? (True/False)	CO3	L2	1M
k)	What is the purpose of WebSocket annotations?	CO4	L3	1M
l)	What are the various operations of RESTful webservice resource?	CO4	L2	1M
m)	What are the various life cycle events of a web sockets.	CO4	L2	1M
n)	Write any two advantages of JSP?	CO4	L3	1M
	Unit -I			
2. a)	Discuss in detail about JEE architecture with a neat sketch.	CO1	L1	7M
b)	Explain in detail about JDBC application steps with an example.	CO1	L3	7M
	(OR)			
3. a)	What a Java servlet does? Explain the procedure about creation of Java servlet with an application?	CO1	L2	7M
b)	What is session? Illustrate HTTP session tracking using cookies in a servlets?	CO1	L3	7M
	Unit -II			
4. a)	How do you interpret JSPs in the web container? Generate a servlet code from JSP.	CO2	L3	7M
b)	Demonstrate custom tags with a suitable program along with Expression Language.	CO2	L4	7M
	(OR)			
5. a)	Explain in detail about various JSP action tags.	CO2	L1	7M
b)	Explain various JSP Scripting elements with an example?	CO2	L3	7M
	Unit -III			
6.	What is RESTful Web Service? and explain various RESTful web service concepts	CO3	L4	14M
	(OR)			
7.	Explain in detail about Java web socket Encoders and Decoders.	CO3	L1	14M
	Unit -IV			
8. a)	Explain the flavors of Enterprise Beans.	CO4	L1	7M
b)	Explain in detail EJB life cycle with example.	CO4	L3	7M
	(OR)			
9. a)	Explain in detail about exposing EJBs.	CO4	L2	7M
b)	Explain the run time architecture of JSP with a neat sketch	CO4	L3	7M

1. **a. Differentiate Web Server and Application Server.**

Ans: A Web Server is a computer system which hosts websites. An Application Server is specially designed to run applications and it includes both hardware and software and provides an environment for the programs to run.

b. List out various JSF Core Tags.

Ans: Data Validation Tags, Event Handling Tags, Data Converters Tags.

c. What is EJB?

Ans: "Enterprise Java Bean" is a server-side reusable software component that encapsulates the business logic of an application. The business logic is the code that fulfils the purpose of an application.

d. What is the purpose of Web Socket?

Ans: The WebSocket protocol is a TCP-based protocol that provides a full duplex communication (2-way communication) channel over a single connection. Both the parties communicate and exchange the data at the same time.

e. Define RESTful web service.

Ans: "Web Service" is a piece of software available over the internet and uses a standard XML messaging system. Through a web service we can interact with the other web applications for exchanging data. REST (Representational State Transfer) is an architectural style not a protocol. Web services based on REST architecture are known as "RESTful web services".

f. What are the different types of statements in JDBC?

Ans: i. Register & Load the driver class.

ii. Establish a connection to the database.

iii. Create a statement obj for sending SQL statements to the database.

iv. Prepare & execute an SQL statement.

v. Close the connection.

g. List out various Session tracking techniques?

Ans: HttpSession, Cookies, Hidden form field, URL Rewriting.

h. Why do we need JSP technology? If we already have Servlets?

Ans: Through servlets it is difficult to design complicated webpages as a response to the client. Implementation of JSP is simple and easy to maintain. JSP code looks pretty much similar to HTML code.

i. What is the full form of JSON?

Ans: JSON (JavaScript Object Notation).

j. The WebSocket protocol supports binary and text-based messages? (True/False)

Ans: True.

k. What is the purpose of WebSocket annotations?

Ans: To handle the WebSocket endpoint lifecycle events.

l. What are the various operations of RESTful webservice resource?

Ans: CRUD operations. (Create, Read, Update, Delete)

m. What are the various life cycle events of WebSockets?

Ans: onConnect( ), onText( ), onBinary( ), onClose( ), onError( ).

n. Write any two advantages of JSP?

Ans: i. Implementation of JSP is simple and easy to maintain.

ii. JSP code looks pretty much similar to HTML code.

iii. In order to process client requests, we don't need to write the service() overridden method.

Unit-I

2. a. Discuss in detail about JEE architecture with a neat sketch. **7M**

Ans:

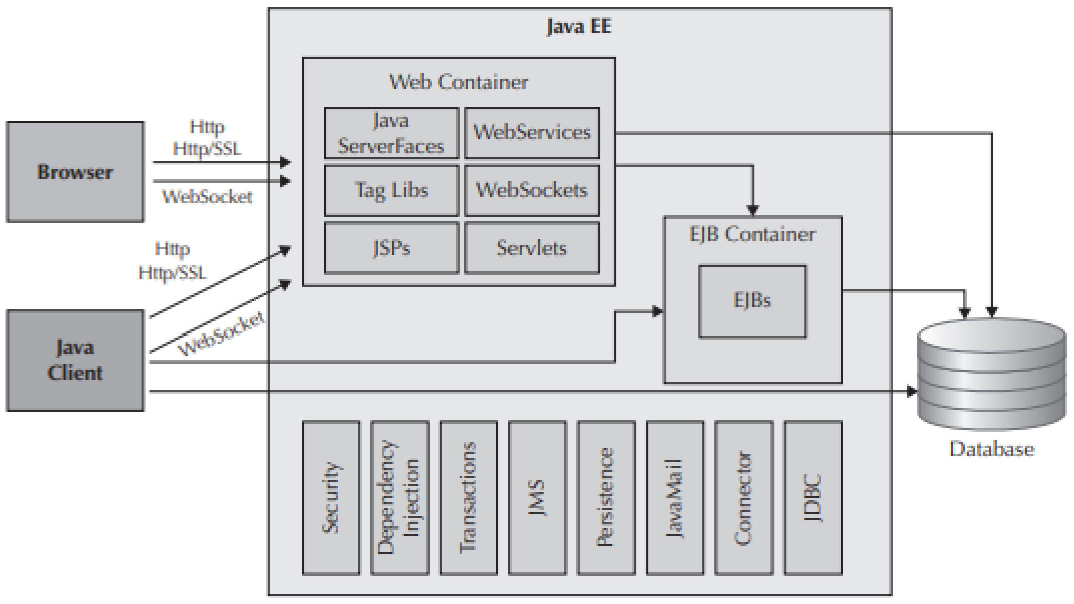


FIGURE 1-1. *Architecture of the Java EE platform*

• It is also known as "Three-tier Architecture" (i.e):

- Client-tier:
  - ▪ It acts as a front end for an application.
  - ▪ It consists of programs that interacts with the user.
  - ▪ Those programs act as a user interface for applications and get the input from the user and then convert into request and forwarded to the middle tier.
  - ▪ The server processes the request and returns the result and data to the client program.
  - ▪ The client program translates the server response into text and presented

to the user.

▪ It holds the logic about how the application is interacted with the user.

▪ Through this tier, client can make a request to the middle tier.

- Server-tier:

  ▪ It consists of various containers for processing the request.

  ▪ Web container is used to deploy web applications.

  ▪ EJB container is used to deploy business applications.

  ▪ It holds the main logic about how the entire application works.

  ▪ It handles the client requests, process it and generate the response.

- Database-tier or EIS:

  ▪ It acts as a backend for an application.

  ▪ It holds the logic about how the data should be stored, fetched and displayed to the user.

• As a Java EE developer we have to mainly focussed on the middle tier to make the application easier, robust and secure.

• The large box labelled "Java EE" represents "Java EE Platform". It refers to the runtime environment provided by the Java EE application server.

• All the code that we develop runs in that environment.

• That environment is also called as "Java EE Container". It is made up of 2 other containers:

a. Web container:

- To run the web components in a Java EE application.
- The web components are web pages, Java servlets, Web services, etc.
- Different web components can interact with the clients with standard web protocols.

b. Enterprise JavaBeans container:

- To run the application logic part of a Java EE application.
- EJBs are java classes that contain and manipulate the core data structures of a Java EE application.

• The database tier of Java EE application holds all application data.

• The Java EE platform supports a variety of protocols that clients may interact with a Java EE application.

• The browser client connects to the Java EE application using standard web protocols such as HTTP and Web sockets.

• The small boxes in the Java EE container represents a variety of services that the application may choose to use.

b. Explain in detail about JDBC application steps with an example. 7M

Ans:

- It is a standard Java API for database connectivity between Java Programming language and a wide range of databases.
- Before JDBC, ODBC API was the database API to connect and execute queries with the database.
- ODBC API uses ODBC driver which was written in C language which was platform dependent and unsecured.
- Java has defined its own API (JDBC API) that uses a JDBC driver which was written in Java language.
- We can use JDBC API to access tabular data stored in any relational database.
- “Java.sql” package contains classes and interfaces for JDBC API.
- The following are the steps to be followed in a JDBC application:

  Step1: Create a project and a database and connect to it.

  Step2: Add "java DB Driver" to the "Libraries" folder.

  Step3: Import "java.sql.*" package in the java file.

  Step4: Copy "Driver class" of the connected database.

  Step5: Load the driver class using "forName()"

  Class.forName("org.apache.derby.jdbc.ClientDriver");

  "ClassNotFound" exception must be handled.

  Step6: Establish a connection to the database using database URL.

  Connection con=DriverManager.getConnection("jdbc:derby://localhost:1527/mydb", "mydb", "mydb");

  Step7: Create a statement obj for sending SQL statements to the database.

  Statement stmt=con.createStatement();

  Step8: Prepare & execute an SQL statement

  stmt.executeUpdate("insert into student values(1,'xyz',18)");

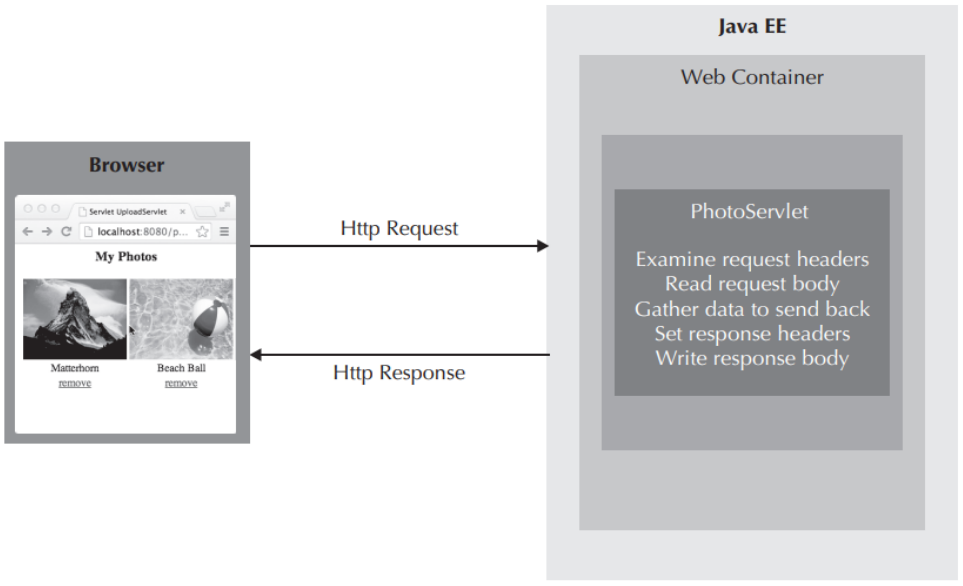
  Step9: Close the connection

  con.close();

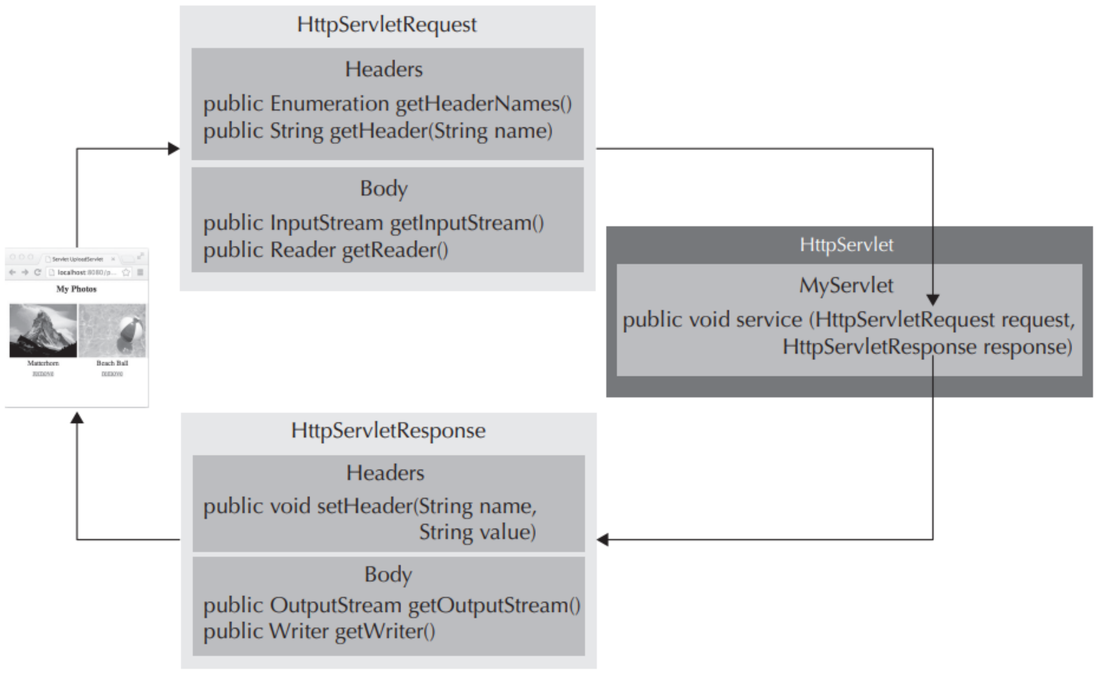
(OR)

3. a. What a Java servlet does? Explain the procedure about creation of Java servlet with an application? 7M

Ans:



- In the above diagram, "PhotoServlet" is running in the web container of a JEE server.
- The "PhotoServlet" will perform various functions like:
    1. **Examine request headers:** What kind of information that the client is requesting and how the client would like to receive it.
    2. **Read request body:** Gathers the information from the request body, if it is POST request. This step doesn't require, if it is GET request.
    3. **Gather data to send back:** Java Servlet can access different components or a database or external systems to gather the application data dynamically.
    4. **Set response headers:** Java Servlets fills out the response headers in order to describe to the client that what kind of information that is held in the response.
    5. **Write response body:** Java Servlets fills out the response body and transmitted back to the client.


HttpServletRequest
Headers
public Enumeration getHeaderNames()
public String getHeader(String name)
Body
public InputStream getInputStream()
public Reader getReader()
HttpServlet
MyServlet
public void service (HttpServletRequest request,
HttpServletResponse response)
HttpServletResponse
Headers
public void setHeader(String name,
String value)
Body
public OutputStream getOutputStream()
public Writer getWriter()
My Photos

- The following are the classes available in Java Servlet API, that are helpful to create a java servlet:
  1. javax.servlet.http.HttpServlet
  2. javax.servlet.http.HttpServletRequest
  3. javax.servlet.http.HttpServletResponse
- Java Servlet is implementing the request/response interaction.
- To do that, we can override one of the following methods of "HttpServlet" class, depending on the type of HTTP request:
  1. public void service (HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException //To handle all HTTP requests.
  2. protected void doGet (HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException //To handle only HTTP GET requests.
  3. protected void doPost (HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException //To handle only HTTP POST requests.
  4. protected void doHead (HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException //To handle only HTTP HEAD requests.
  5. protected void doOptions (HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException //To handle only HTTP OPTIONS requests.
  6. protected void doPut (HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException //To handle only HTTP PUT requests.
  7. protected void doTrace (HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException //To handle only HTTP TRACE requests.
- The following are the methods of "HttpServletRequest" class:
  1. public String getHeader(String name) //To read HTTP request header
  2. public InputStream getInputStream( )

     (or)

     public Reader getReader( ) //To read HTTP request body
- The following are the methods of "HttpServletResponse" class:
  1. public void setHeader(String name, String value) //To set the HTTP response header
  2. public OutputStream getOutputStream( )

     (or)

     public Writer getWriter( ) //To write the body content of HTTP response.

b. What is session? Illustrate HTTP session tracking using cookies in a servlets? 7M

Ans:

- A session means "small interval of time".

- Session Tracking defines a way to maintain the state (data) of a user in a server.
- It is also known as "State Management".
- It is a technique used by the server to maintain the user data. It means the server can remember the user's data for a particular period of time.
- The client sends a request to the server through HTTP protocol.
- HTTP is a stateless protocol, because each request is executed independently without any knowledge of the requests that were executed before.
- Because of stateless protocol, session tracking is used to maintain the state of a user.

  Example: when a client requests (Req1) a page from the server, the server accepts the request and forward it to the servlet for processing and generate a response to the client. When a client requests (Req2) another page from the server, the server will treat it as a new request from a new user. Because of stateless protocol, the server can't able to find that the requests are coming from the same client or from a new client. The server will not remember the data of previous requests. This is called as "State management problem". In order to maintain the state, we have to use session tracking techniques.

  Eg:

```
//In Servlet1
Cookie c=new Cookie("key_name","key_value");
resp.addCookie(c);
//In Servlet2
Cookie c[]=req.getCookies();
for(Cookie c: cookies)
{
        if(c.getName().equals("key_name"))
                var=c.getValue();
}
```

Unit-II

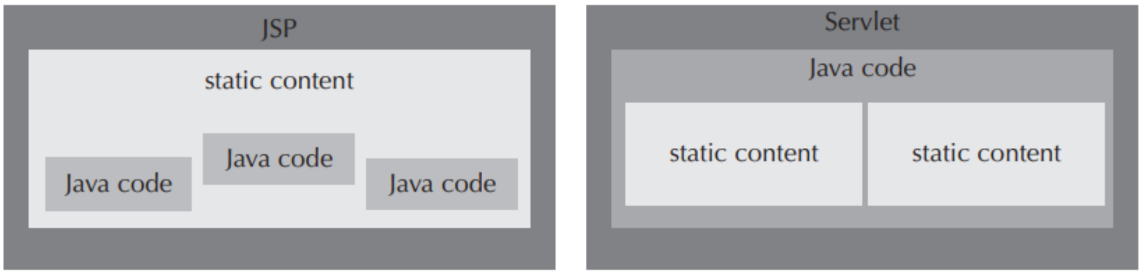
4. a. How do you interpret JSPs in the web container. Generate a servlet code from JSP. 7M

Ans:

- JSP stands for "Java Server Pages".
- Java servlets supports the creation of dynamic web content.
- JEE provides a "Web component model" (JSP, JSF) to create/generate the dynamic web content for all kinds of web browsers.
- JSP is the extension of "Servlet Technology". It provides more functionality than servlets.

- Servlets provides a great foundation for JSP.

- It works similar to a servlet.
- When a servlet receives a client request, it will be processed and a response (in HTML) will be prepared and sent back to the client.
- While generating the response HTML tags are embedded in method calling statements.
- Through servlets it is difficult to design complicated webpages as a response to the client.
- Implementation of JSP is simple and easy to maintain.
- JSP code looks pretty much similar to HTML code.
- A JSP page can include HTML, CSS, JavaScript & Java.



Eg:

//Home.jsp

```
        <%@ page language="java" contentType="text/html"%>
        <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
                "http://www.w3.org/TR/html4/loose.dtd">
        <html>
                <head>
                        <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
                        <title> My first JSP   </title>
                </head>
                <body>
                        <form action="HelloServlet">
                                 Please enter a color <br>
                                <input type="text" name="color"size="20px">
                                <input type="submit" value="submit">
                                </form>
                </body>
        </html>
```

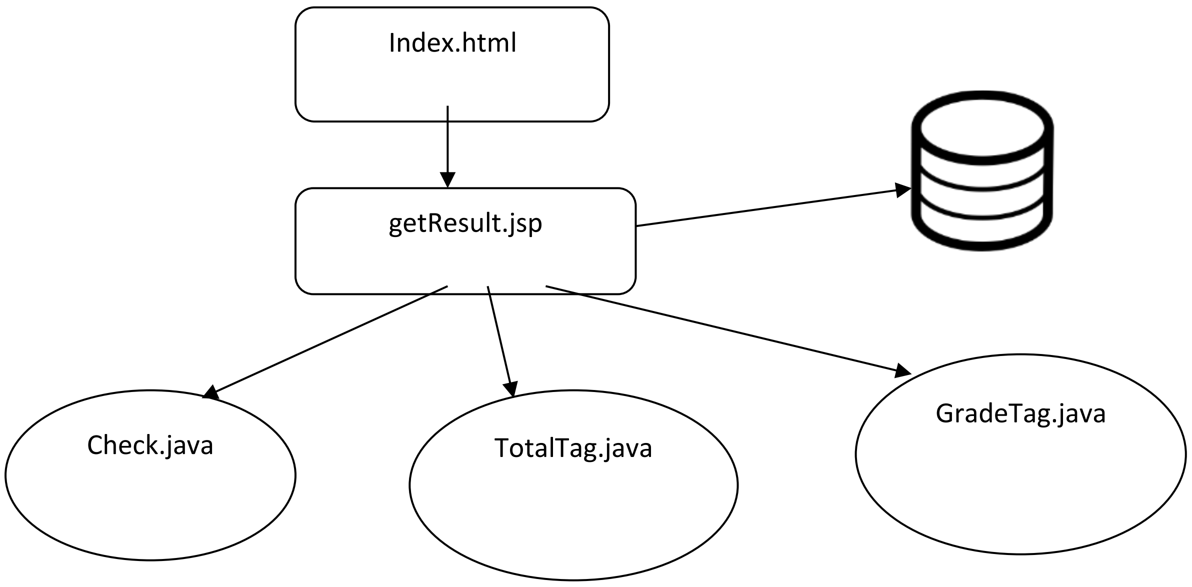
//MyServlet.java

```
        public class MyServlet extends HttpServlet {
                 protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
          {
           // reading the user input
           String color= request.getParameter("color");
        PrintWriter out = response.getWriter();
        out.println (
          "<!DOCTYPE html PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\"
              \"http://www.w3.org/TR/html4/loose.dtd\">\n" +
          "<html> \n" +
            "<body> \n" +
              "<font size=\"12px\" color=\"" + color + "\">" +
                "Hello World" +
              "</font> \n" +
            "</body> \n" +
          "</html>"
        );
      }
    }
```

b. Demonstrate custom tags with a suitable program along with Expression Language. 7M

Ans:

In this application first user enter his/her register number to find the results. when user submit after entering the registration number results is fetch from the data base using JSP standard tags “sql “tags. With the marks getting the grade and displayed to the user’s web page. Grade is displayed in various colors depending up on the total marks getting by the user. The total of all the subjects is sent to the custom tag handler created in tag library created in tag library.



Index.html

```
<!DOCTYPE html>
<html>
    <body>
    <center>
        <form>
            <h2><b><u>ResultForm</u></b></h2>
            Enter HallTicket Number<input type="text" name="regdno"/>
            <input type="submit" value="GetResult" formaction="getResult.jsp"/>
        </form>
    </center>
    </body>
</html>
```

GetResult.jsp

```
<%--
    Document   : getResult
    Created on : 30 Jan, 2019, 9:26:33 PM
    Author     : PRAVEENDOPPALAPUDI
--%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>
<%@taglib uri="/WEB-INF/tlds/TotalTag" prefix="mytag" %>
<%@taglib uri="/WEB-INF/tlds/check" prefix="mytag1" %>
<%@taglib uri="/WEB-INF/tlds/GradeTag" prefix="mytag2" %>
<!DOCTYPE html>
<html>
    <body>
        <sql:setDataSource var="con"  driver="org.apache.derby.jdbc.ClientDriver"
url="jdbc:derby://localhost:1527/Student"  user="app" password="app"></sql:setDataSource>
        <sql:query var="res" dataSource="${con}">
            select * from marks where regdno='<%=request.getParameter("regdno")%>'
        </sql:query>




        <c:forEach items="${res.rows}" var="i">
```

```
<center>
    <table>
        <tr>
            <th>RegdNo</th>
            <td>${i.regdno}</td>
        </tr>
        <tr>
            <th>Name</th>
            <td>${i.name}</td>
        </tr>
        <tr>
            <th>FatherName</th>
            <td>${i.fname}</td>
         </tr>
       </table>
</center>
<center>
    <table border='1px' cellpadding='10px'>
        <tr>
            <th>Subjects</th>
            <th>ActualMarks</th>
            <th>FinalMarks</th>
        </tr>
        <tr>
            <td>Sub1</td>
            <td>100</td>
            <td><mytag1:check sub="${i.sub1}"></mytag1:check></td>
        </tr>
        <tr>
            <td>Sub2</td>
            <td>100</td>




            <td><mytag1:check sub="${i.sub2}"></mytag1:check></td>
```

```
                </tr>

                <tr>

                    <td>Sub3</td>

                    <td>100</td>

                    <td><mytag1:check sub="${i.sub3}"></mytag1:check></td>

                </tr>

                <tr>

                    <td>Sub4</td>

                    <td>100</td>

                    <td><mytag1:check sub="${i.sub4}"></mytag1:check></td>

                </tr>

                <tr>

                    <td colspan="2">Total</td>

                    <td><mytag:TotalTag sub1="${i.sub1}" sub2="${i.sub2}" sub3="${i.sub3}"
sub4="${i.sub4}"></mytag:TotalTag></td>

                </tr>

                <tr>

                    <td colspan="2">Grade</td>

                    <td><mytag2:GradeTag sub1="${i.sub1}" sub2="${i.sub2}" sub3="${i.sub3}"
sub4="${i.sub4}"></mytag2:GradeTag></td>

                </tr>

            </table>

        </center>

</c:forEach>

    </body>
</html>
```

TotalTag.java

```
package com;


import java.io.IOException;

import javax.servlet.jsp.JspWriter;

import javax.servlet.jsp.JspException;




import javax.servlet.jsp.tagext.SimpleTagSupport;
```

```
public class TotalTag extends SimpleTagSupport {

    private String sub1;
    private String sub2;
    private String sub3;
    private String sub4;
    String msg=null;

    @Override
    public void doTag() throws JspException, IOException {
        JspWriter out = getJspContext().getOut();

       int i1= Integer.parseInt(sub1);
       int i2= Integer.parseInt(sub2);
       int i3= Integer.parseInt(sub3);
       int i4= Integer.parseInt(sub4);
       int total=i1+i2+i3+i4;
       if(total<=160)
       {
                      out.println("<p style='color:red'>"+total+"F</p>");
}
        public void setSub1(String sub1) {
        this.sub1 = sub1;
    }

    public void setSub2(String sub2) {
        this.sub2 = sub2;
    }

    public void setSub3(String sub3) {
        this.sub3 = sub3;
    }




    public void setSub4(String sub4) {
        this.sub4 = sub4;
    }
    }
```

TotalTag.tdl

```
<?xml version="1.0" encoding="UTF-8"?>
<taglib version="2.1" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-
jsptaglibrary_2_1.xsd">
  <tlib-version>1.0</tlib-version>
  <short-name>totaltag</short-name>
  <uri>/WEB-INF/tlds/TotalTag</uri>

  <tag>
    <name>TotalTag</name>
    <tag-class>com.TotalTag</tag-class>
    <body-content>scriptless</body-content>
    <attribute>
      <name>sub1</name>
      <required>true</required>
      <rtexprvalue>true</rtexprvalue>
      <type>java.lang.String</type>
    </attribute>
    <attribute>
      <name>sub2</name>
      <required>true</required>
      <rtexprvalue>true</rtexprvalue>
      <type>java.lang.String</type>
    </attribute>
    <attribute>
      <name>sub3</name>
      <required>true</required>




      <rtexprvalue>true</rtexprvalue>
      <type>java.lang.String</type>
    </attribute>
    <attribute>
      <name>sub4</name>
      <required>true</required>
      <rtexprvalue>true</rtexprvalue>
      <type>java.lang.String</type>
    </attribute>
  </tag>
</taglib>
```

GradeTag.java

```
package com;

import java.io.IOException;
import javax.servlet.jsp.JspWriter;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.tagext.SimpleTagSupport;



public class GradeTag extends SimpleTagSupport {
    private String sub1;
    private String sub2;

    public void setSub1(String sub1) {
        this.sub1 = sub1;
    }

    public void setSub2(String sub2) {
        this.sub2 = sub2;
    }






    public void setSub3(String sub3) {
        this.sub3 = sub3;
    }
    public void setSub4(String sub4) {
        this.sub4 = sub4;
    }
    private String sub3;
    private String sub4;
    @Override
    public void doTag() throws JspException, IOException {
        JspWriter out = getJspContext().getOut();
        int i1= Integer.parseInt(sub1);
       int i2= Integer.parseInt(sub2);
       int i3= Integer.parseInt(sub3);
       int i4= Integer.parseInt(sub4);
       int total=i1+i2+i3+i4;
       if(total>=350)
       {
            out.println("<p style='color:green'>A</p>");
       }
       else if(total>300 && total<350)
       {
       out.println("<p style='color:yello'>B</p>");
       }
       else if(total>160 && total<300)
       {
       out.println("<p style='color:blue'>C</p>");
       }
       else
       {
       out.println("<p style='color:red'>D</p>");
       }
    }



}
```

GradeTag.tld

```
<?xml version="1.0" encoding="UTF-8"?>
<taglib version="2.1" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-
jsptaglibrary_2_1.xsd">
  <tlib-version>1.0</tlib-version>
  <short-name>gradetag</short-name>
  <uri>/WEB-INF/tlds/GradeTag</uri>
  <!-- A validator verifies that the tags are used correctly at JSP
          translation time. Validator entries look like this:
       <validator>
           <validator-class>com.mycompany.TagLibValidator</validator-class>
           <init-param>
              <param-name>parameter</param-name>
              <param-value>value</param-value>
           </init-param>
       </validator>
    -->
  <tag>
    <name>GradeTag</name>
    <tag-class>com.GradeTag</tag-class>
    <body-content>scriptless</body-content>
    <attribute>
      <name>sub1</name>
      <rtexprvalue>true</rtexprvalue>
      <type>java.lang.String</type>
    </attribute>
    <attribute>
      <name>sub2</name>
      <rtexprvalue>true</rtexprvalue>
      <type>java.lang.String</type>
    </attribute>
    <attribute>
      <name>sub3</name>
      <rtexprvalue>true</rtexprvalue>
      <type>java.lang.String</type>
    </attribute>
    <attribute>
      <name>sub4</name>
      <rtexprvalue>true</rtexprvalue>
      <type>java.lang.String</type>
    </attribute>
  </tag>
</taglib>
```

Check.java

```
package com;
 import java.io.IOException;
import javax.servlet.jsp.JspWriter;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.tagext.JspFragment;
import javax.servlet.jsp.tagext.SimpleTagSupport;
public class check extends SimpleTagSupport {

    private String sub;
 @Override
    public void doTag() throws JspException, IOException {
        JspWriter out = getJspContext().getOut();
        int subject=Integer.parseInt(sub);
        if(subject<40)
        {
            out.println("<p style='color:red'>"+subject+"F");
        }




        else
        {
            out.println(subject+"P");
        }
    }
public void setSub(String sub) {
        this.sub = sub;
    }


}
```

Check.tld

```
<?xml version="1.0" encoding="UTF-8"?>
<taglib version="2.1" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-
jsptaglibrary_2_1.xsd">
  <tlib-version>1.0</tlib-version>
  <short-name>check</short-name>
  <uri>/WEB-INF/tlds/check</uri>


  <tag>
    <name>check</name>
    <tag-class>com.check</tag-class>
    <body-content>scriptless</body-content>
    <attribute>
      <name>sub</name>
      <required>true</required>
      <rtexprvalue>true</rtexprvalue>
      <type>java.lang.String</type>
    </attribute>
  </tag>
</taglib>
```

(OR)

5. **a. Explain in detail about various JSP action tags.** **7M**

Ans:

- "Java Bean" is a server-side reusable software component that encapsulates the business logic of an application.
- Like java, java beans also follow the "write once run anywhere" paradigm.
- The goal of a java bean in JSPs is to separate the dynamic java code from the static web content.
- "Java Beans" are technically a "Java Class".
- "Java Beans" should have the following:
  a. Zero arguments constructor. (Optional)
  b. Getters and Setters.
- To use a java bean within a JSP, use the following syntax:
  <jsp:useBean id="bean_name" class="fully qualified classname">
- We can use a java bean in 2 ways:
  a. Retrieve a property from the java bean using the following syntax:
     <jsp:getProperty name="bean_name" property="property_name">
  b. Set a value on the java bean using the following syntax:
     <jsp:setProperty name="bean_name" property="property_name" value="property_value">

b. Explain various JSP Scripting elements with an example? **7M**

Ans:

1. Declarative Tag:
   <%! Var's or Methods %>
   Eg:

```
<%!
                int a=10,b=20;
                String s="welcome to bec"
                public int add()
                {
                        return (a+b);
                }
                public String rever()
                {
                        StringBuffer sb=new StringBuffer(s);
                    return s.reverse();
                }
%>
```

2. Scriptlet Tag:

It includes the java code which we want to embed in service( ) overridden method.

```
<%      code  %>
```

Eg:

```
<%
          out.print(“Result= ”+add());
          out.print(“Reverse= “+rever());
%>
```

3. Expression Tag:

```
<%=  statement %>
```

Eg:

```
<h1>Result= <%=  add()%></h1> //To call a method
<%= a%> //To print a variable value
```

4. Directive Tag: (@page, @include, @taglib)

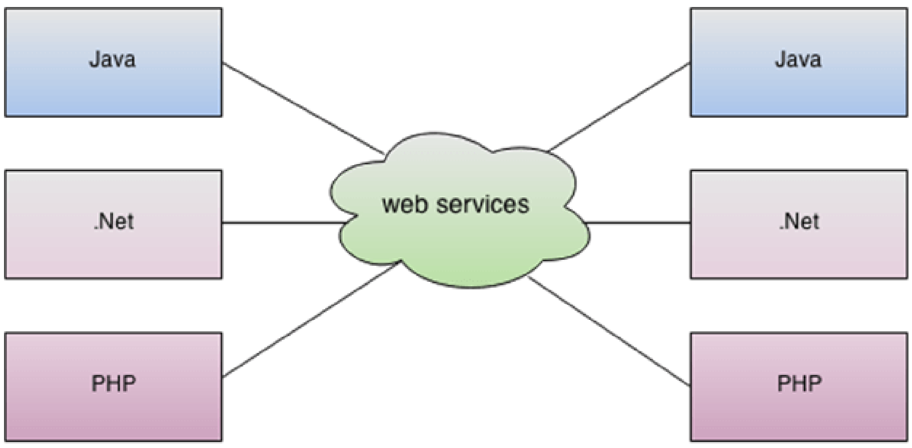
```
<%@page import=”package_name,..” %>
<%@include file=”file_path” %>
<%@taglib prefix=”x” uri=”http://java.sun.com/jsp/jstl/category_name” %>
```

Unit-III

6. What is RESTful Web Service? and explain various RESTful web service concepts? 14M

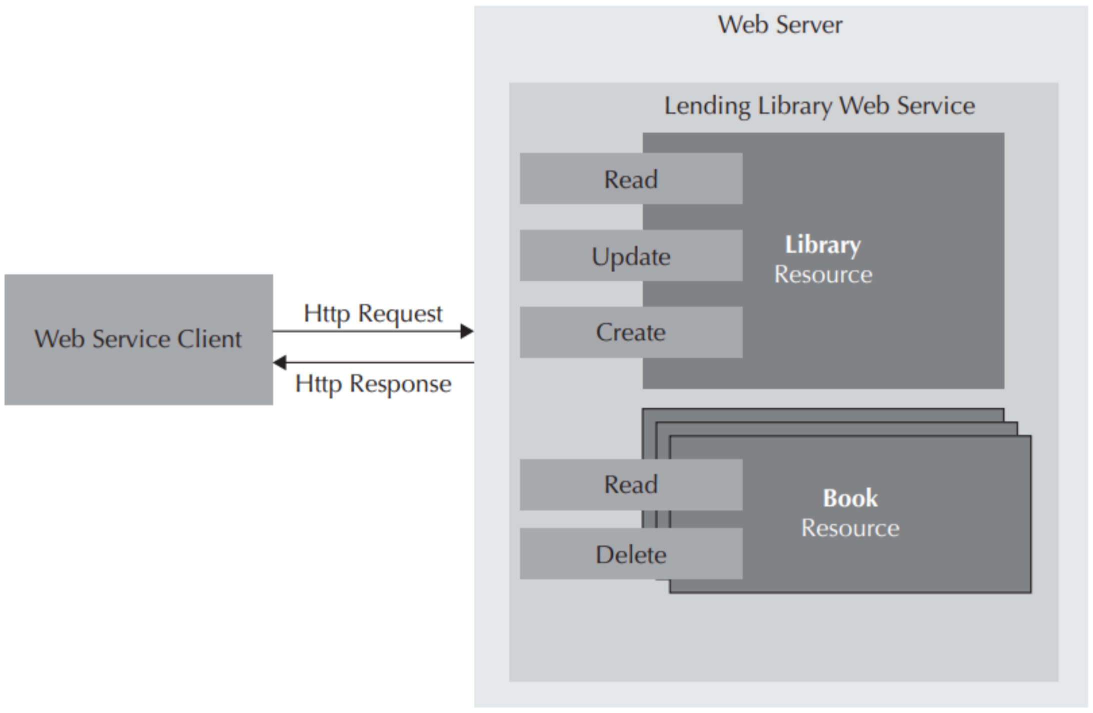
Ans:

- JSP & JSF is mainly focussed on the production of dynamic HTML pages.
- “Web Service” is a piece of software available over the internet and uses a standard XML messaging system. Through a web service we can interact with the other web applications for exchanging data.
- A web service is a method of communication between 2 applications over the network.
- A web service is a collection of open protocols & standards used for exchanging data between applications.



- In the above diagram, different applications are interacted through a web service over the network.

- Applications written in various programming languages and running on various platforms can use a web service to exchange data over the network.
- JEE platform contains a variety of web components that can handle various web services.
- Web services are mainly categorized into 2 types (i.e) SOAP and REST.
- SOAP (Simple Object Access Protocol) is an XML-based protocol for accessing web services. It is a W3C recommendation for communication between 2 applications. It consumes more bandwidth and resources. These APIs uses “Web Services Description Language” for describing the functionalities being offered.
- REST (Representational State Transfer) is an architectural style not a protocol. These web services are fast, will consume less bandwidth and resources. These APIs uses “Web Application Description Language” for describing the functionalities being offered.
- These are used to create APIs for web-based applications.
- REST was introduced by “Roy Fielding” in 2000.
- The protocol used for REST is HTTP.
- Web services based on REST architecture are known as “RESTful web services”.
- RESTful web services are highly scalable, light and maintainable.
- These services uses a model where applications & data are exposed as resources.
- Each resource can be identified by using URIs.
- A resource may allow itself to be read by its client, updated by its client and deleted by its clients and may also allow the creation of new resources.
- Not all web service resources allow all such operations.



- Client interacts with the web service (i.e) Lending Library Web service.
- This service consists of 2 types of web service resources. (i.e) Library resource, Book resource.

- “Library resource” allows browsing of catalogue, updating the catalogue using read and update operations.
- The library contains many “Book resources”.
- Books may be added to the library using create operation.
- Individual book can be examined using read operation on the book resource.
- Books may be removed from the library using delete operation on the book resource.
- All these operations collectively called as “CRUD operations”.
- RESTful web services are categorized using the following properties:

  1. **Uses a URI space to address web service resources:**

     There are no rules about how the URI space is structured. The common practice for URI space is to be hierarchically arranged.

     Eg: The library resource were mapped to “/library”, then the books would be mapped to “/library/books/15”.

  2. **Uses HTTP methods for operations:**

     RESTful web services use various HTTP methods to perform operations on resources. The following are various HTTP methods which are commonly used:

     a. HTTP GET: Used to perform read operations on a resource.

     b. HTTP POST: Used to create a new resource.

     c. HTTP DELETE: Used to remove a resource.

     d. HTTP PUT: Used to update an existing resource (or) to create a new resource.

     Eg: HTTP GET /library -> To perform the read operation which gives a list of all books in the library.

     HTTP PUT /library/books/80 -> To update the library’s book 80.

     HTTP DELETE /library/books/50 -> To delete the 50 from the library.

  3. **These resources are stateless:**

     The resources don’t hold the state across multiple requests. When a request comes from the new client, the resources doesn’t remember anything.

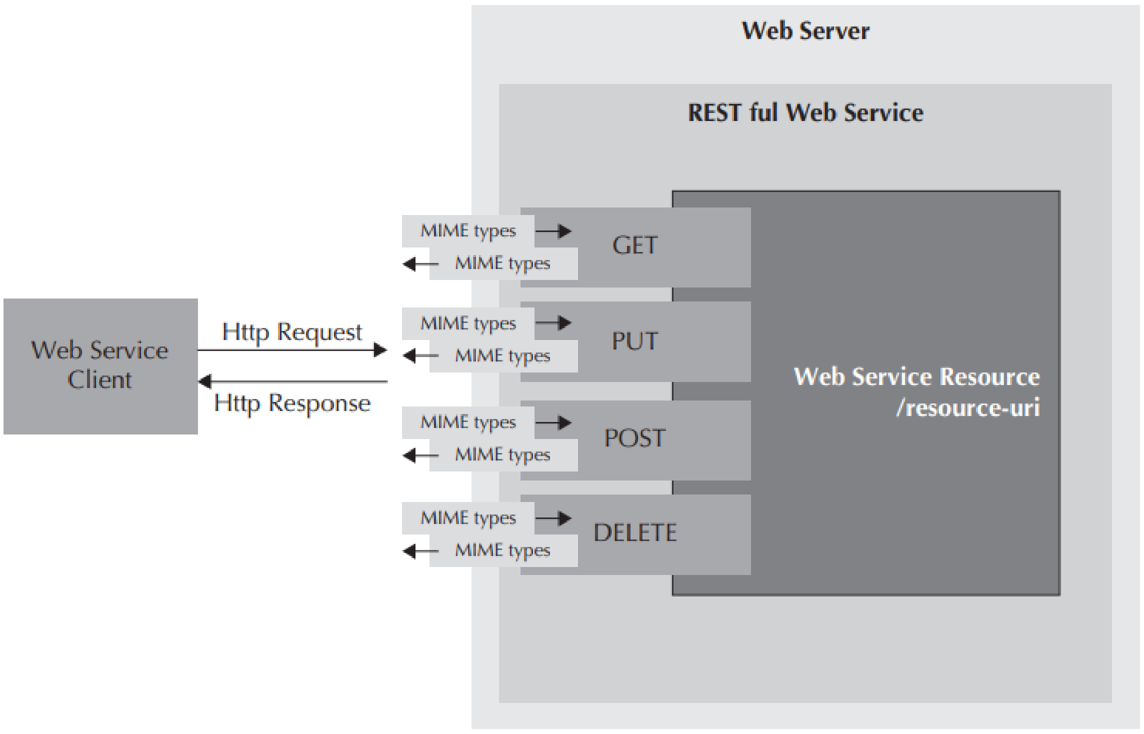
  4. **Uses familiar formats for structured data:**

     - These services use MIME (Multipurpose Internet Mail Extension) type to identify the request information and process it to generates response.
     - MIME is an extension of SMTP (simple Mail Transfer Protocol), it lets the users to exchange different kinds of information through email.
     - On the request side, client uses the “HTTP Accept header”, to indicate the format of data it is sending to the web service resource.

- On the response side, the web service resource uses “Content-type header” to indicate the MIME type of response it is giving.
- RESTful web services can consume and produce any MIME type.
- Most web services consume and produce structured data as XML or as JSON (JavaScript Object Notation) or both.

Eg: HTTP GET /library/book/90, mime-type=”application/xml”

```
<book>
        <title>C Programming</title>
        <author>Balaguru Swamy</author>
</book>
```



(OR)

7. Explain in detail about Java web socket Encoders and Decoders. **14M**

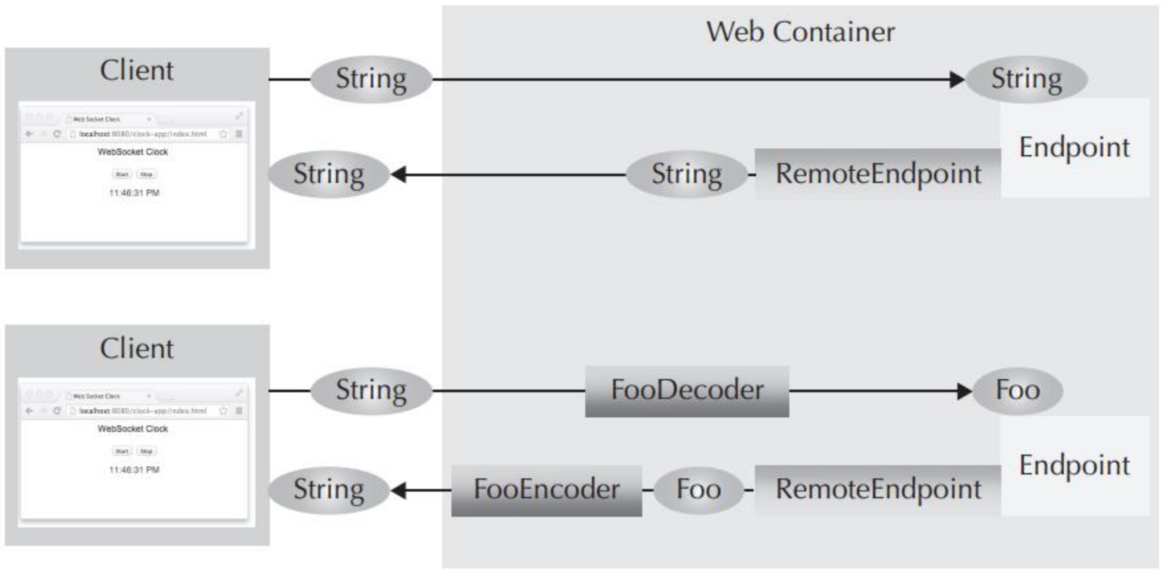
Ans:

- WebSocket protocol supports 2 formats (i.e) text & binary.
- Simple applications can exchange simple information between client and server.
- But if an application has anything more complicated to send or receive over a WebSocket connection, it will follow a structure to put the information.
- Encoders and Decoders are used to transmit complex structures between client and server.

- A WebSocket can use a "Decoder" to transform a text message into a Java object and then handle it in the @OnMessage method.
- A WebSocket can use an "Encoder" to convert the object into text and send it back to the client.
- Java WebSocket API attempts to convert incoming messages into any Java primitive type or its equivalent class.

  Eg:

  ```
          @OnMessage
      public void onMessage(int x)
                  (or)
      @OnMessage
      public void onMessage(Boolean b)
  ```
- When sending messages, the RemoteEndpoint contains a general-purpose method "sendObject( )", into which we can pass any primitive or its equivalent class, and the implementation converts the value into string.
- A WebSocket Decoder implementation used to convert the incoming message into object type.
- A WebSocket Encoder implementation used to convert object type into message and it will be forwarded to the client.



- In the above figure, one endpoint exchanging strings with the client and the other endpoint uses encoder and decoder for converting text messages into objects and vice versa.
- Java WebSocket API is having 2 interfaces (i.e) javax.websocket.Encoder and javax.websocket.Decoder.
- To implement the interface Decoder.Text<T> or Decoder.Binary<T>, we must implement the following method to convert incoming messages into object.

```
public Class_name decode(String s) throws DecodeException
```

This method will be called for each incoming message.

- To implement the interface Encoder.Text<T> or Encoder.Binary<T>, we must implement the following method to convert object into incoming messages.

  public String encode(Class_name x) throws EncodeException

This method will be called for each outgoing message.

Unit-IV

8. a. Explain the flavors of Enterprise Beans. **7M**

Ans:

There are 3 types of EJBs:

1. **Session beans:**
   - These are general purpose & commonly used kind of EJB, accessible through RMI/IIOP, web service protocols and local java method invocations.
   - These are further classified into 3 types:
     a. Stateless Session beans:
        - It can't hold the client's state between calls.
        - This bean is indicated with @Stateless annotaion.
        - The EJB container will maintain a pool of EJB instances.
        - The client requests will be handled by different instances of the EJBs.
        - An instance can be shared by multiple clients.
        - As soon as the request is served, the client will be released.

     b. Stateful Session beans:
        - It can hold the client's state between calls.
        - This bean is indicated with @Stateful annotaion.
        - The EJB container will maintain a single stateful session bean instance for each client.
        - That instance will not be shared with the other clients.
        - When the communication between the client and the bean ends, the bean will get terminated.
        - This client's state will be stored in the instance variables of EJB implementaion class.
        - These type of beans are the most commonly used kind of EJB.
        - These are used in the applications that collect user data over a sequence of interactions.
        - The downside of these beans are, they are not so scalable like stateless beans.

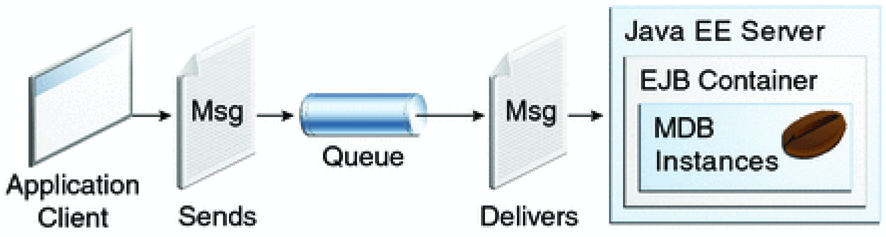
     c. Singleton Session beans:

- These are similar to Stateless session beans, but only one instance of Singleton session bean is created.
- That instance doesn't terminates until the application is terminated.
- This bean is indicated with @Singleton annotaion.
- The created instance is shared between multiple clients and it can be concurently accessed.

2. **Entity beans.**

   An entity bean is a remote object that manages persistent data, performs complex business logic, potentially uses several dependent Java objects, and can be uniquely identified by a primary key.

3. **Message-driven beans:**



- These are special kind of EJBs, clients can send messages using JMS (Java Message Service).
- This bean is indicated with @MessageDriven annotation.
- These beans are stateless. Any bean instance can process the client's messages.
- Messages are received through a listener interface (i.e) "MessageListener" (javax.messaging), with a method (i.e) onMessage( ).

  void onMessage(Message msg)

  Where "Message", represents the incoming message.
- In session beans, methods are called on EJB.
- In message-driven beans, messages are pushed into a queue, which delivers to the bean asynchronously.
- These beans are useful in applications, where clients need not to wait around for a task to complete.

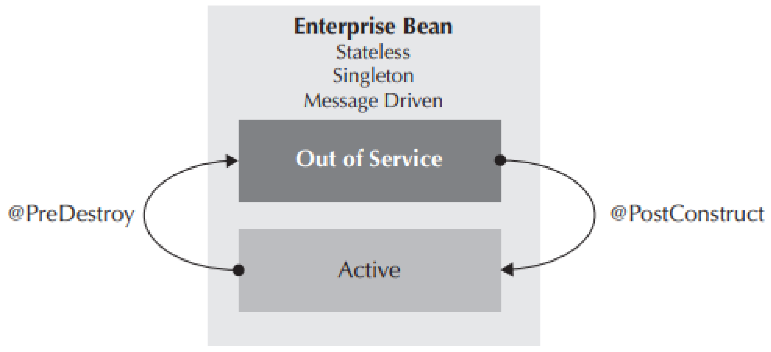
b. Explain in detail EJB life cycle with example. **7M**

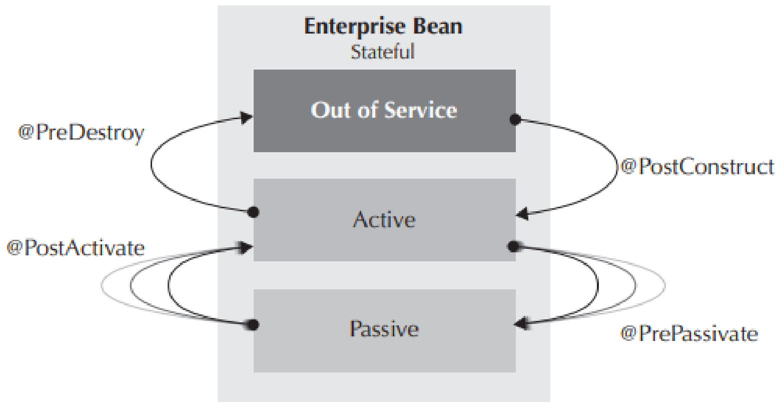
Ans:

- The EJB container manages the lifecycle of EJBs.
- Stateless, Singleton and Message-driven beans have the same lifecycle.
- In those beans there are 2 states (i.e) "Out of Service" & "Active".

- Initially, there are no instances present inside the EJB container.

- As soon as a client makes a request, then EJB instance will be created by the container.
- Once the bean instance is ready, then it goes into active state. In active state, the bean instance can accept and service incoming calls from clients.
- When the container doesn't need of bean instance, it will goes to the out of service and it will be destroyed.
- EJB container may destroy the instances as soon as the client requests is serviced.
- In the case of singleton session bean, instances will be created only once prior to any client request and then destroyed only once when all the clients requests are completed.
- To move the bean from out of service to the active state, we should use @PostConstruct annotaion.
- To move the bean from active to the out of service state, we should use @PostDestroy annotaion.
- In statefull session beans, there are 3 states (i.e) "Out of Service", "Active" & "Passive".
- In the passive state, the bean is temporarily out of service. The container will not pass any client request to the bean.
- When a request comes in, the container bring the bean back to active state.
- The container may bring the bean in and out of the passive state many times.
- @PrePassivate and @PostActivate annotaions are used in between active and passive states.





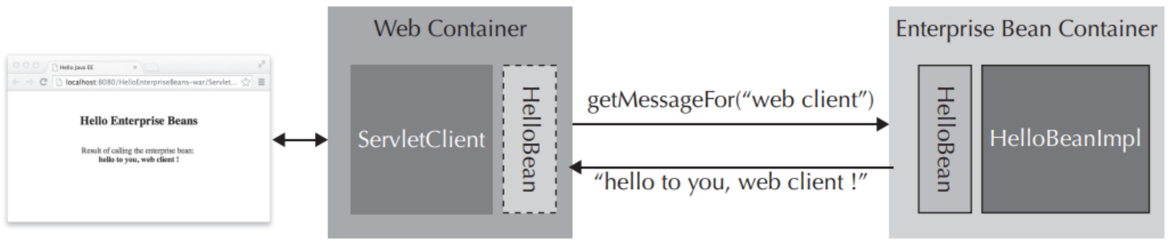
(OR)

9. a. Explain in detail about exposing EJBs. **7M**

Ans:

- "Java Bean" is a server-side reusable software component that encapsulates the business logic of an application.
- The business logic is the code that fulfils the purpose of an application.
- EJB provides an architecture to develop and deploy component-based enterprise applications.
- EJB is used to develop scalable, robust and secured enterprise applications in java.
- EJB is a specification provided by Sun Microsystems to develop secured, robust and scalable distributed applications.
- An EJB application can be deployed on any application server.
- EJB is a server-side component, it is required to be deployed on the server.
- "Enterprise Java Beans" are instantiated by the EJB container when each time a new client wishes to use it. It (Stateful Session Bean) is useful in the application to hold its application state for each of its connected clients.

  Eg: If the application needs to model a Shopping cart and to hold user data while purchasing a product through online.
- EJBs will expose the available methods are to the clients in 2 ways:
  1. Remotely
  2. Locally
- When an EJB expose its methods remotely, we have to use @Remote annotaion in the interface.
- When an EJB expose its methods remotely, we have to use @Local annotaion in the interface.

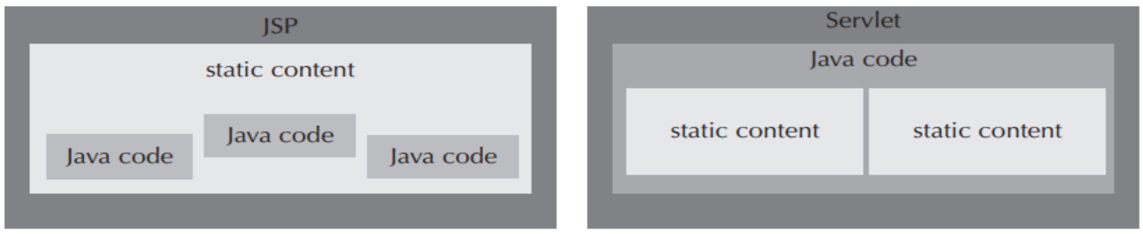

Hello Enterprise Beans
Result of calling the enterprise bean:
hello to you, web client !
Web Container
ServletClient
HelloBean
getMessageFor("web client")
"hello to you, web client !"
Enterprise Bean Container
HelloBean
HelloBeanImpl

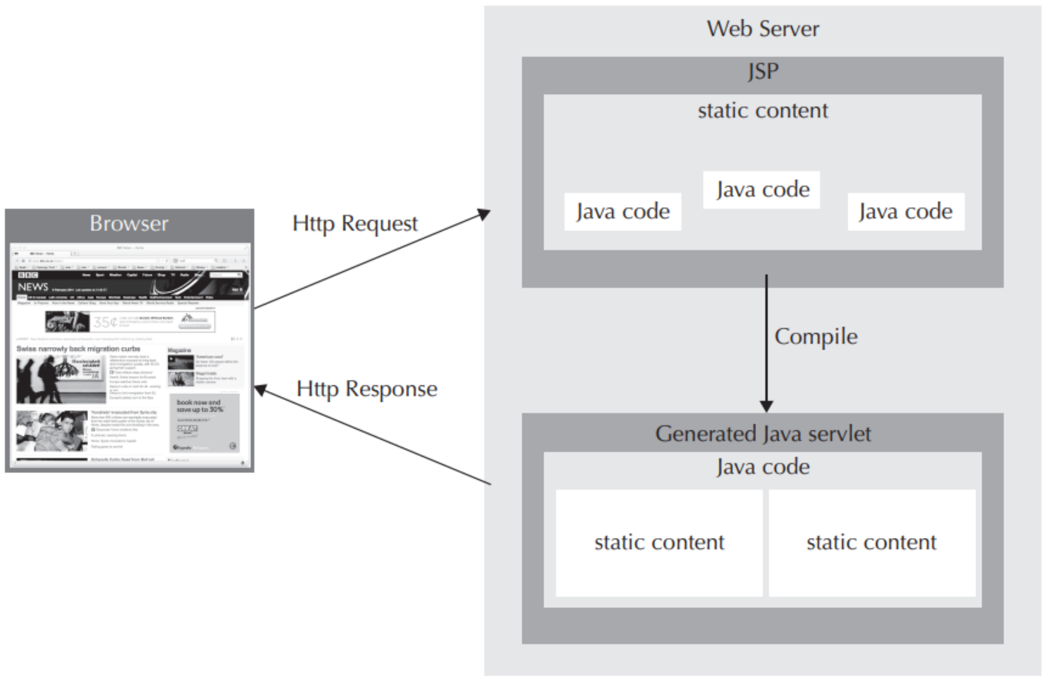
- For the created EJB there should be 1 interface and 1 implementation class.
- In the EJB interface, we have to include abstract methods and add a class-level annotations (i.e) @Remote or @Local.
- The created EJB interface will be used by an EJB to publish its methods to all the clients.
- Next create an EJB to behave like an implementaion class for the EJB interface.
- Implement all the abstract methods of EJB interface in the implementation class.
- In the servlet, create an instance to the EJB interface through @EJB annotation.

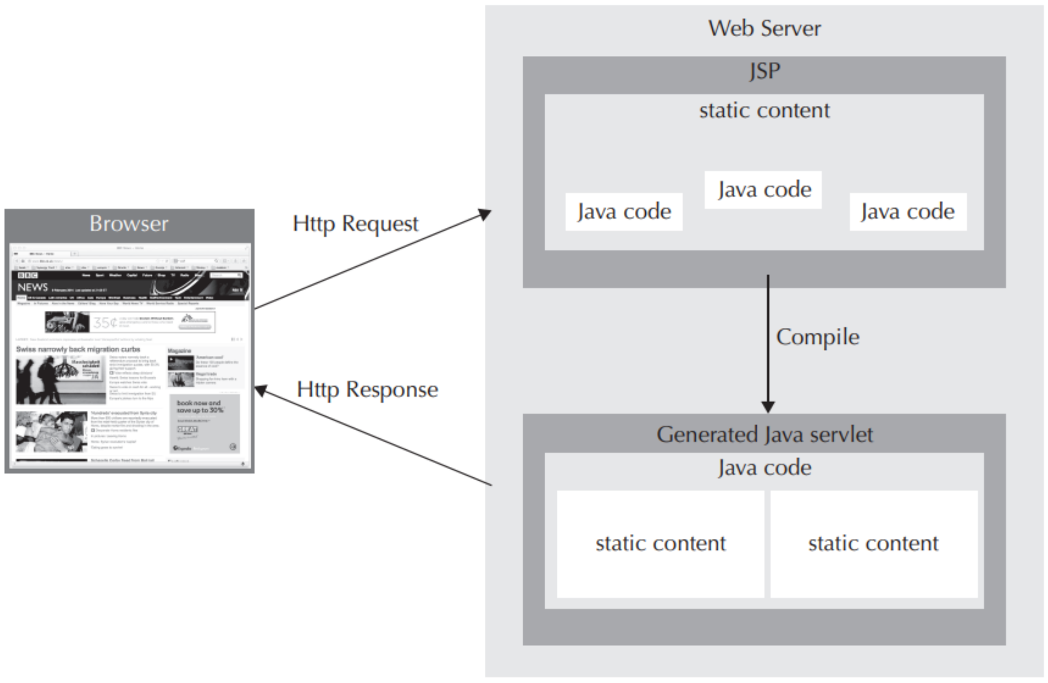
b. Explain the run time architecture of JSP with a neat sketch. **7M**

Ans:

- JSP stands for “Java Server Pages”.
- Java servlets supports the creation of dynamic web content.
- JEE provides a “Web component model” (JSP, JSF) to create/generate the dynamic web content for all kinds of web browsers.
- JSP is the extension of “Servlet Technology”. It provides more functionality than servlets.
- Servlets provides a great foundation for JSP.
- It works similar to a servlet.
- When a servlet receives a client request, it will be processed and a response (in HTML) will be prepared and sent back to the client.
- While generating the response HTML tags are embedded in method calling statements.
- Through servlets it is difficult to design complicated webpages as a response to the client.
- Implementation of JSP is simple and easy to maintain.
- JSP code looks pretty much similar to HTML code.
- A JSP page can include HTML, CSS, JavaScript & Java.
- The following are the various disadvantages of servlets:
  1. Static html content is generated by a servlet.
  2. Designing of response is difficult.
  3. For every request we have to write service () overridden method inside each servlet.
  4. Each of the client request will be handled by anyone of the servlets.
  5. Whenever modification is made in the static content (presentation logic), then a servlet needs to recompiled and redeployed.
- In JSP, in order to process the client requests, we don’t need to write the service ( ) overridden method.
- In JSP, in order to process the client requests, we can embed the java code through a “scriptlet tag” (<% code %>).
- Once a JSP page is created, we can send another request through it without overriding service ( ) method.
- JSP offers JSTL library which provides both predefined tags and custom tags.
- JSP never really deals the HTTP request or the HTTP response.
- Once we deploy the JSP to the JEE web container, the web container reads the deployed JSP.
- From that information it will generate the java code for a servlet that fulfils the objectives of JSP.







Prepared By:	**Department**	**Signature**
J. Madhan Kumar	Computer Science & Engineering	
Taught By:	**Department**	**Signature**
1. J. Madhan Kumar	Computer Science & Engineering	
2. S.N. Chandra Sekhar	Computer Science & Engineering	

H.O.D., C.S.E