14IT704

Hall Ticket Number:

IV/IV B.Tech (Regular/Supplementary) DEGREE EXAMINATION

Jan/Feb, 2021 — **Information Technology**

Seventh Semester — **Enterprise Programming - II**

Time: Three Hours — **Maximum :** 60 Marks

Answer ALL Questions from PART-A. (1X12 = 12 Marks)
*Answer **ANY FOUR** questions from PART-B.* (4X12=48 Marks)

Part – A

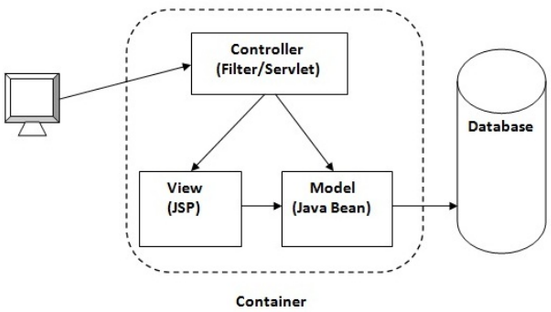
1 Answer all questions (1X12=12 Marks)

a) What is an Enterprise Application?
Enterprise application software (**EAS**), is computer software used to satisfy the needs of an organization rather than individual users. Such organizations include businesses, schools, interest-based user groups, clubs, charities, and governments.

b) Write any two statements in SQL with syntax.
Any two SQL query with syntax.

c) What is Servlets?
Servlet is a web component which is used to develop the web applications.

d) Draw MVC architecture.



e) Define Java Web Socket encoder.
WebSocket provides support for converting between WebSocket messages and custom Java types using encoders and decoders. An encoder takes a Java object and produces a representation that can be transmitted as a WebSocket message.

f) What are the types of Cookies?
Session **cookies,** Permanent **cookies**

g) List the implicit objects of JSP.
Out, request, response, config, application, session, pageContext, page, exception

h) List types of JSP elements.
Scriplet, Page, Declarative, definition

i) State the purpose of an EJBs.
EJB beans are specifically designed to implement the business logic of your application. As such they provide services that are often needed when implementing such logic, such as transactions, injecting of the entity manager (used for JPA, the **Java** Persistence API) and pooling of beans.

j) List the relations of entities.
One to one, one to many, many to one, many to many

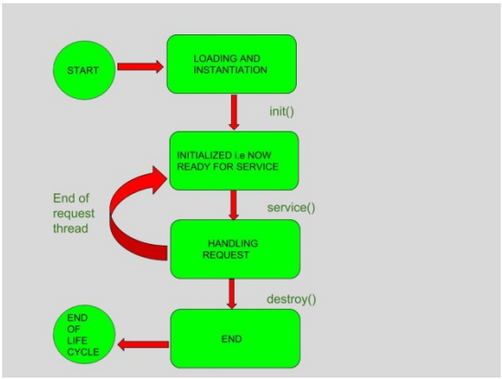
k) Define Web Service
A **web service** is any piece of software that makes itself available over the internet and uses a standardized XML messaging system.

l) What is WSDL?
WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information.

Part – B

2 a) Explain in detail about servlet life cycle. **6 M**

[Description-4, Diagram – 2]



init() method.

service() method.

destroy() method.

<u>init() Method</u>

- The servlet is initialized by calling the **init()** method.
- The init method is called only once.
- It is called only when the servlet is created, and not called for any user requests afterwards.
- public void init() throws ServletException

{ // Initialization code... }

<u>service() Method</u>

- The servlet calls **service()** method to process a client's request.
- main method to perform the actual task.
- service() method to handle requests coming from the client

(browsers) and to write the formatted response back to the client.

- The service() method checks the HTTP request type (GET, POST, PUT, DELETE, etc.) and calls doGet, doPost, doPut, doDelete, etc. methods as appropriate.
- public void service (ServletRequest request, ServletResponse response)

  throws ServletException, IOException

  {

  //body

  }

<u>destroy() Method</u>

- The servlet is terminated by calling the **destroy()** method.
- The destroy() method is called only once at the end of the life cycle of a servlet.
- public void destroy()

{

// Finalization code...

}

[Any valid example program]

2 b) Explain Java EE architecture. **6M**

[Description-4, Diagram – 2]

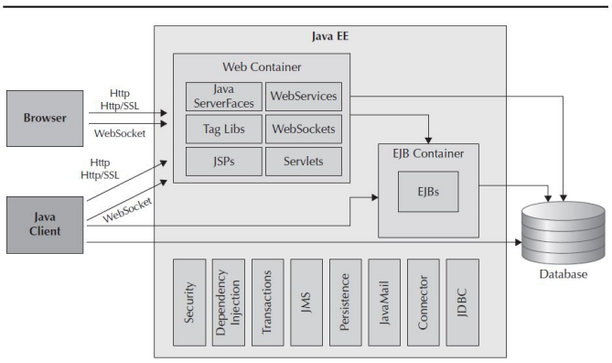


FIGURE 1-1. *Architecture of the Java EE platform*

The Java EE platform supports a wide variety of protocols that clients may use to interact.

- **Browser client:** This connects to the Java EE application using standard web protocols such as HTTP and WebSockets.
- **Java clients:** This connects by Java EE application such as Java desktop application, running on a different application server.

Java EE platform refers to the runtime environment provided by a Java EE application server.

- **Java EE container**, the container can mediate or intercept calls to and from the application code, and insert other kinds of logic that qualify and modify the calls to and from the application code.
- The Java EE container can enforce security rules on the application that is running, for example, a rule such as “only allow access from suresh and ramesh to my application.”
- The Java EE server container is itself made up of two other containers:
- **web container**
- **Enterprise JavaBeans (EJB) container**.
- The **web container** is devoted to running the web components in a Java EE application: the web pages, Java servlets, and other Java EE web components that can interact with clients connecting to the Java EE application with standard web protocols.
- The **EJB container** is devoted to running the application logic part of the Java EE application. Enterprise JavaBeans are Java classes that contain and manipulate the core data structures of the Java EE application.
  These other boxes represent a variety of services that a Java EE application may choose to use.

- The **security** service enables to restrict access to its functions to only a certain set of known users.
- The **dependency injection** service enables to delegate the lifecycle management and discovery of some of its core components.
- The **transaction** service enables to define collections of methods that modify application data in such a way that either all the methods must complete successfully, or the whole set of method executions is rolled back as though nothing has ever happened.
- The **Java Message Service (JMS)** exposes to the ability to reliably send messages to other servers in the deployment environment of the Java EE application server.
- The **Persistence** service enables application data in the form of a Java object to be synchronized with its equivalent form in the tables of a relational database.
- The **JavaMail** service enables to send email, particularly useful in the kind of application that takes some action initiated by and on behalf of a user, and which needs to notify the user at some later time of the outcome of the action.
- The **Java EE Connector Architecture (JCA)**, which provides a framework into which a new service that is not a standard part of the Java EE platform may be added and that can then.

3 a) Write an application to demonstrate servlets. 6 M

[Client side program - 3, Server side program – 3]

```
// Import required java libraries
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

// Extend HttpServlet class
public class HelloWorld extends HttpServlet {

  private String message;

  public void init() throws ServletException {
    // Do required initialization
    message = "Hello World";
  }

  public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

    // Set response content type
    response.setContentType("text/html");

    // Actual logic goes here.
    PrintWriter out = response.getWriter();
    out.println("<h1>" + message + "</h1>");
  }

  public void destroy() {
    // do nothing.
  }
}
```

[Any valid servlet Example]

3 b) Explain in detail about SQL statements in JDBC. 6 M

[List – 2, Description – 4]

SQL Statements

create

insert

select

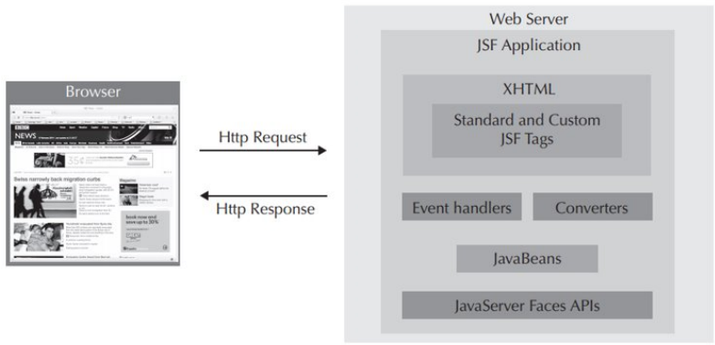
delete

update

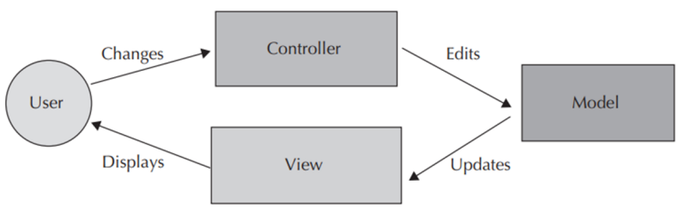
Explanation, syntax and example

[Any 4 SQL satement]

4 a) Explain the architecture of a JSF with an example? 6 M
[Description-4, Diagram – 2]



- A JSF application is made up of a collection of
- **XHTML pages**
- **Java classes**
- **JSF metadata.**
- The **XHTML pages** make up the visual part of the web application.
- It holding the various **JSF tags, markup, and scripting content** that define the UI elements.
- The **Java classes** include JavaBeans that hold various parts of the application data.
- The **Java classes** in the JSF application define the **application data** that is housed in the web application, and also **mediate the way the application data** is bound into the presentation layer of the application.
- Together with Java classes that mediate the data by implementing various types from the JSF APIs, including event handlers, data validators, and data converters.
- The **JSF metadata** defines how the XHTML pages and JavaBeans are treated in the JSF runtime.
- This metadata defines the scope of the JavaBeans, how many instances are used, and their lifecycles, together with navigation rules, and can be used to control how the UI components are rendered.
- Model-View-Controller



- The **user** makes a change to the UI, which is passed to the controller.
- The **controller**'s job in this design pattern is to interpret the changes to the data model that the user is asking for and transform that request, or series of requests, into a form that can be used to edit the data model.
- When the model changes, the controller's job is to make sure that the view is informed of the change.
- The **view**, in turn, has the task of altering its state in such a way that the display information it presents to the user of the application is consistent with the newly edited state of the data model.
- Using this kind of separation of concerns, the data model never needs to interpret all the different ways that a user may ask for a change to the data model, because the controller is taking care of that.
- The data model need never concern itself with how the data model is displayed to the user, because the view is taking care of that.

[Any Example program]

4 b) Explain at least 4 JSP action tags with their uses. 6 M
[List – 2, Description – 4]

There are five types of JSP tags

- Expression tag
- Declaration tag
- Scriptlet tag
- Directive tag
- Comment tag

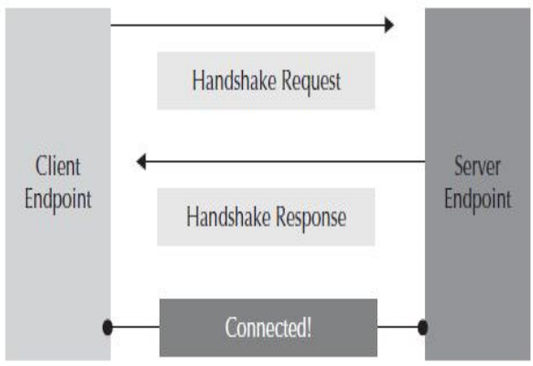
Explanation, syntax and example

[Any 4 tags]

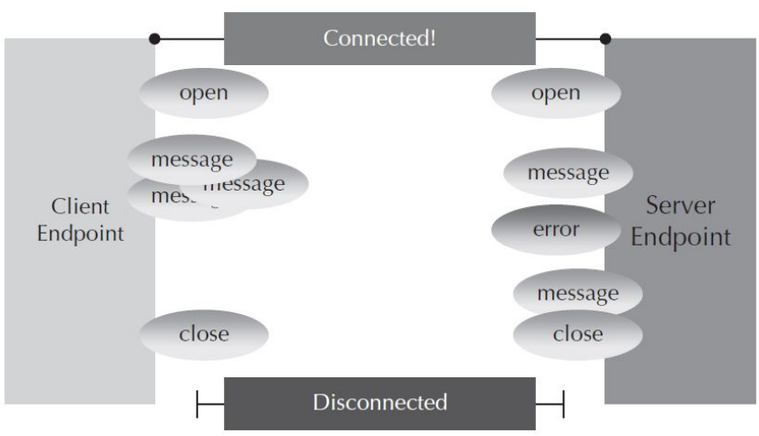
5 a) Explain about Java Web socket architecture? 6 M

[Description-3, Diagram – 3]

- In the WebSocket protocol, a client and a server are mostly the same as each other.
- The only antisymmetric in the protocol is in the initial phase of the connection being established, where it matters who initiated the connection.
- It is somewhat like a phone call. To make the phone call happen.
- WebSocket client is almost always a browser or a rich client running on a laptop, smartphone, or desktop computer, and the WebSocket server
- First, the client initiates a connection request.
- This occurs when the client sends a specially formulated HTTP request to the web server.
- What identifies this as a WebSocket opening handshake request over any common or garden-variety HTTP request is the use of the ***Connection : Upgrade and Upgrade : websocket*** headers, and the most important information is the request URI, /myChat.
- The web server decides whether it supports WebSockets at all (all Java EE web containers do).
- If so, whether there is an endpoint at the request URI of the handshake request that meets the requirements of the request.
- If all is well, the WebSocket-enabled web server responds with an equally specially formulated HTTP response called a WebSocket opening handshake response

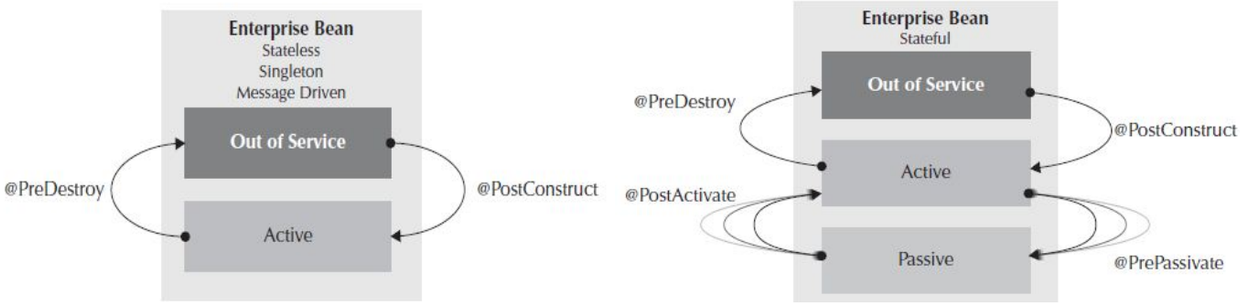


- This response confirms that the server will accept the incoming TCP connection request from the client and may impose restrictions on how the connection may be used.
- Once the client has processed the response and is happy to accept any such restrictions, the TCP connection is created.
- Each end of the connection may proceed to send messages to the other.
- Once the connection is established, a number of things can occur:
- ***Either end of the connection may send a message to the other.*** This may occur at any time that the connection is open. Messages in the WebSocket protocol have two flavors: **text and binary**.
- ***An error may be generated on the connection.*** In this case, assuming the error did not cause the connection to break, both ends of the connection are informed. Such non terminal errors may occur, for example, if one party in the conversation sends a badly formed message.
- ***The connection is voluntarily closed.*** This means that either end of the connection decides that the conversation is over and so closes the connection. Before the connection is closed, the other end of the connection is informed of this.



5 b) Draw and explain the life cycle of EJB. 6 M

[Description-4, Diagram – 2]



- Singleton and stateless session beans together with message-driven beans have the same lifecycle.
- There are two states: Out of Service and Active.
- When beans are being brought into service, they start in the Out of Service state.
- When the container needs to bring a bean into service, it invokes the constructor, injects any dependencies (@EJB annotation) and then brings the instance into the Active state.
- The bean instance can accept and service incoming calls from clients.
- When the container has no more need for the bean instance, it brings the instance out of service.
- cleans up any supporting resources, and releases the bean instance for garbage collection.
- Stateless session beans and message-driven bean instances are required to exist only for the duration of a client call.
- Enterprise Bean containers may bring them into service only as a client makes a call to them.
- Enterprise Bean containers may destroy such bean instances as soon as the client call is serviced.
- singleton session beans are created only once prior to any client calls being serviced, and then destroyed only once all the client calls have been completed, when the container shuts down.
- You may intercept all these state changes on your Enterprise Beans, from Out of Service to Active, and from Active to Out of Service.
- Java method to be called when the state change occurs and mark it with one of the state change annotations.
- when it is moving from the Out of Service to the Active state, you use the @PostConstruct annotation.
- Note: the bean instance has been constructed and any dependency injection has already occurred.
- when it is moved from the Active state to the Out of Service state, you use the @PreDestroy annotation.
- Stateful session beans have a lifecycle that includes these same states, Out of Service, Active and Passive state.
- use @PostConstruct and @PreDestroy annotations to ask the container to call you when it transitions your Enterprise Beans between these states.
- The Passive state is one in which the Enterprise Bean instance is taken out of service temporarily.
- The container will not pass any client calls to the Enterprise Bean while it is in this state.
- When a call comes in for a bean in the Passive state, the container brings it back into the Active state in order to respond to the call.
- The container may bring the same Stateful session bean instance in and out of the Passive state many times.
- The reason for this extra state in the case of Stateful session beans is, the container must create a new instance for each and every client that uses it.
- I.e., stateful session beans with large numbers of clients, the container needs to support correspondingly large numbers of instances.
- using the @PrePassivate and @PostActivate events to annotate methods so the container calls them during the transitions.
- Final note on singleton beans: the Enterprise Bean container has a choice as to when to instantiate.
- It can do it at any time between the deployment of the application containing the singleton bean and the moment immediately prior to a client call needing a response.
- The class-level @Startup annotation on a singleton bean to direct the container to instantiate the bean upon application deployment, rather than allow the container to leave it to the last minute.

6 a) List and explain the types of EJB. 6 M

[List – 1, Description – 4, Diagram-1]

There are **three** types of EJB:

1. Session beans
2. Entity Bean
3. Message-driven beans

1. Session beans

- Session bean encapsulates business logic only, it can be invoked by local, remote and webservice client.
- It can be used for calculations, database access etc.
- The life cycle of session bean is maintained by the application server (EJB Container).
- These are not persistent because they do not survive a server crash or a network failure.

A) Stateful Session Bean
B) Stateless Session Bean
C) Singleton Session Bean

A) Stateful Session Bean:

- It maintains state of a client across multiple requests.
- Stateful session bean can be used to access various method calls by storing the information in an instance variable.
- in which it is necessary to maintain state, such as instance variable values or transactional state, between method invocations.
- For example, shopping site, the items chosen by a customer must be stored as data is an example of stateful session bean.
- A stateless session bean implementation class is marked **@Statefull.**

B) Stateless Session Bean

- It doesn't maintain state of a client between multiple method calls.
- Stateless session bean can be used in situations where information is not required to used across call methods.
- Stateless session beans do not share state or identity between method invocations.
- They are useful mainly in middle-tier application servers that provide a pool of beans to process frequent and brief requests.
- A stateless session bean implementation class is marked **@Stateless.**

C) Singleton Session Bean

- One instance per application, it is shared between clients and supports concurrent access.
- A stateless session bean implementation class is marked **@Singleton.**

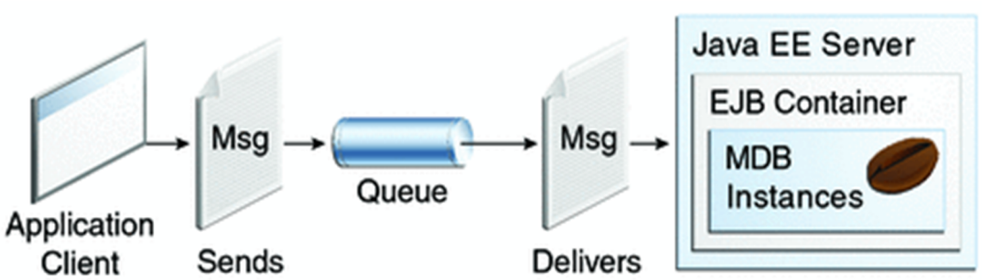
2. Entity Bean

- Entity bean represents the persistent data stored in the database. It is a server-side component.
- There was two types of entity beans: **bean managed persistence** (BMP) and container managed persistence (CMP).
- it is deprecated and replaced by JPA (Java Persistence API) that is covered in the hibernate tutorial.

3. Message-driven beans

- Message-Driven Beans (MDB) provide an easier method to implement asychronous communication than using straight JMS.
- MDBs were created to receive asynchronous JMS messages.
- The container handles much of the setup required for JMS queues and topics. It sends all messages to the interested MDB.

- They are marked using the @MessageDriven annotation



6 b) Write an Enterprise application to demonstrate multithreading in Enterprise Beans 6 M

[Program - 6]

A non-concurrent singleton bean

```
import javax.ejb.Singleton;
@Singleton
public class VotingBeanSingleThreaded {
private String contestant;
private long voterId;
public String getLastVote() {
return "Voter: " + voterId + " vote for " + contestant;
}
public void doVote(long voterId, String contestant) {
// process vote in voting system
this.voterId = voterId;
this.contestant = contestant;
}
}
```

(Or)

A partially concurrent singleton bean

```
import javax.ejb.Singleton;
import javax.ejb.ConcurrencyManagement;
import javax.ejb.ConcurrencyManagementType;
import javax.ejb.Lock;
import javax.ejb.LockType;
@Singleton
@ConcurrencyManagement(ConcurrencyManagementType.CONTAINER)
public class VotingBeanConcurrentReadAccess {
private String contestant;
private long voterId;
@Lock(LockType.READ)
public String getLastVote() {
return "Voter: " + voterId + " vote for " + contestant;
}
@Lock(LockType.WRITE)
public void doVote(long voterId, String contestant) {
// process vote in voting system
this.voterId = voterId;
this.contestant = contestant;
}
}
```

[Any valid example program]

7 a) **Write an application to demonstrate the JSP of login and registration pages. [Login – 3, Registration – 3]** **6 M**

login.jsp

```
<%@ include file="index.jsp" %>
<hr/>
<h3>Login Form</h3>
<%
String profile_msg=(String)request.getAttribute("profile_msg");
if(profile_msg!=null){
out.print(profile_msg);
}
String login_msg=(String)request.getAttribute("login_msg");
if(login_msg!=null){
out.print(login_msg);
}
 %>
 <br/>
<form action="loginprocess.jsp" method="post">
Email:<input type="text" name="email"/><br/><br/>
Password:<input type="password" name="password"/><br/><br/>
<input type="submit" value="login"/>"
</form>
```

Registration page

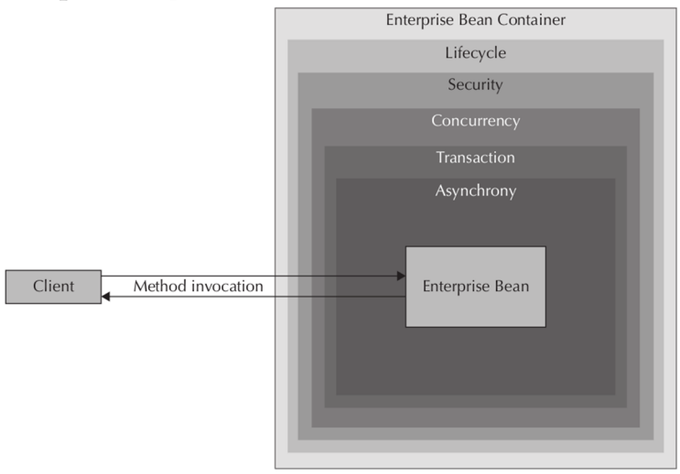
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Guru Registration Form</title>
</head>
<body>
<h1>Guru Register Form</h1>
<form action="guru_register" method="post">
<table style="with: 50%">
<tr>
<td>First Name</td>
<td><input type="text" name="first_name" /></td>
</tr>
<tr>
<td>Last Name</td>
<td><input type="text" name="last_name" /></td>
</tr>
<tr>
<td>UserName</td>
<td><input type="text" name="username" /></td>
</tr>
<tr>
<td>Password</td>
<td><input type="password" name="password" /></td>
</tr>
<tr>
<td>Address</td>
<td><input type="text" name="address" /></td>
</tr>
<tr>
<td>Contact No</td>
```

```
<td><input type="text" name="contact" /></td>
</tr></table>
<input type="submit" value="Submit" /></form>
</body>
</html>
```

[Any valid programs]

7 b) **Briefly discuss the Enterprise Java Bean (EJB) architecture.** **6 M**

[Diagram – 3, Description – 3]



- It performs:
    1. life cycle management
    2. security
    3. Concurrency
    4. transaction management
    5. Asynchrony
- Managing the lifecycle and number of instances of the Enterprise Bean, from instantiation through various phases to destruction.
- Gating calls to Enterprise Beans to allow them to declare which users are allowed access to their methods.
- Managing the concurrency of access to methods exposed by the Enterprise Bean, allowing the Enterprise Bean to determine whether the Enterprise Bean container should allow multiple threads to access its methods at the same time, or whether such threads should wait in line to access methods one at a time.
- Allowing multiple Enterprise Bean method calls to be handled atomically within a single transaction, guaranteeing the integrity of common data being handled by more than one method at a time.
- Allowing Enterprise Beans to implement asynchronous methods, wherein the component's method can return immediately while the container thread waits on the actual process of the method to complete.

These features of the Enterprise Bean component model is used when,

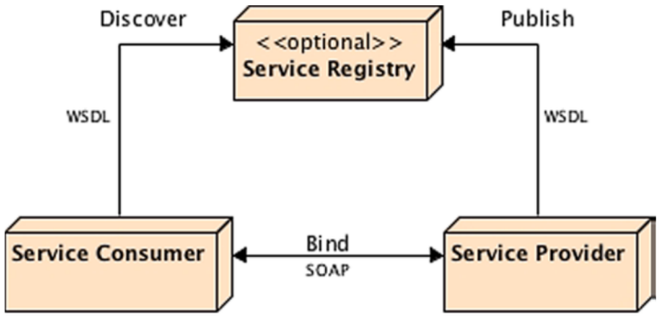
- ❖ Your application needs to scale to a large number of users.
- ❖ Your application is managing relatively complex data that requires transactional support.
- ❖ Your application has a variety of different clients.

8 a) **Demonstrate SOAP web services.** **6 M**

[Diagram – 2, Description – 4]

SOAP is an XML-based protocol for accessing web services over HTTP. It has some specification which could be used across all applications.

The SOAP web service can optionally register its interface into a registry (Universal Description Discovery and Integration, or UDDI) so a consumer can discover it. Once the consumer knows the interface of the service and the message format, it can send a request to the service provider and receive a response.

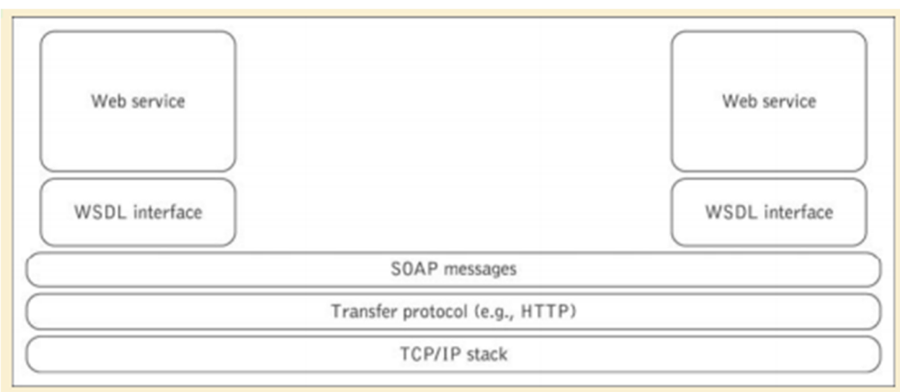


SOAP web services depend on several technologies and protocols to transport and to transform data from a consumer to a service provider in a standard way. The ones that you will come across more often are the following:
Extensible Markup Lan • guage (XML) is the basic foundation on which SOAP web services are built and defined (SOAP, WSDL, and UDDI).
• Web Services Description Language (WSDL) defines the protocol, interface, message types, and interactions between the consumer and the provider.
• Simple Object Access Protocol (SOAP) is a message-encoding protocol based on XML technologies, defining an envelope for web services communication.
• Messages are exchanged using a transport protocol. Although Hypertext Transfer Protocol (HTTP) is the most widely adopted transport protocol, others such as SMTP or JMS can also be used.
• Universal Description Discovery, and Integration (UDDI) is an optional service registry and discovery mechanism, similar to the Yellow Pages; it can be used for storing and categorizing SOAP web services interfaces (WSDL).
With these standard technologies, SOAP web services provide almost unlimited potential. Clients can call a service, which can be mapped to any program and accommodate any data type and structure to exchange messages through XML.

8 b) **Is SOAP as a light weight protocol, Justify it?** **6 M**

[Diagram – 2, Description – 4]

SOAP is a lightweight protocol that allows applications to pass messages and data back and forth between disparate systems. • By lightweight we mean that the SOAP protocol possesses only two fundamental properties. It can: – send and receive HTTP (or other) transport protocol packets, and – process XML messages. • This can be contrasted with the heavyweight protocols such as ORPC protocols.



[Explanation]

9 a) Explain WSDL. Write different WSDL elements and attributes **6 M**

[Definition - 1, Elements – 2.5, Attributes – 2.5]

WSDL stands for Web Service Description Language. WSDL is an XML based document that provides technical details about the web service. Some of the useful information in WSDL document are: method name, port types, service end point, binding, method parameters etc

Some of the different tags in WSDL xml are:

xsd:import namespace and schemaLocation: provides WSDL URL and unique namespace for web service.
message: for method arguments
part: for method argument name and type
portType: service name, there can be multiple services in a wsdl document.
operation: contains method name
soap:address for endpoint URL.

9 b) What are the advantages and disadvantages of SOAP Web Services? **6 M**

[Advantages – 3, Disadvantage - 3]

SOAP web services have all the advantages that web services has, some of the additional advantages are:

- WSDL document provides contract and technical details of the web services for client applications without exposing the underlying implementation technologies.
- SOAP uses XML data for payload as well as contract, so it can be easily read by any technology.
- SOAP protocol is universally accepted, so it's an industry standard approach with many easily available open source implementations.

Some of the disadvantages of SOAP protocol are:

- Only XML can be used, JSON and other lightweight formats are not supported.
- SOAP is based on the contract, so there is a tight coupling between client and server applications.
- SOAP is slow because payload is large for a simple string message, since it uses XML format.
- Anytime there is change in the server side contract, client stub classes need to be generated again.
- Can't be tested easily in browser

HOD,IT Dept.

Scheme Prepared by:
Mr. Shaik.Mabasha,
Dept. of IT, BEC.

Signature of Subject Teachers:

S.No	Name of the Faculty	Name of the college	Signature
1	K Sai Prasanth	BEC, IT Dept.	
2	B Krishnaih	BEC, IT Dept.	