Java EE Assessment - 07

Duration: 90 Minutes **Marks: 50**

Section I [Multiple Choice Questions] : 1 Marks Each

1. Java Platform, Enterprise Edition (Java EE) is a standard for developing and implementing enterprisewide applications:

A. It provides support for multitier applications.
B. It is designed to help improve the process of developing, deploying, and implementing enterprisewide applications.
C. Focus on Single Tie Application
D. End to End Application Development
Answer: A, B, D

---

2. Java Enterprise Edition is a Programming Language

A. True
B. False
Answer: B. False

---

3. JNDI is all about Identifying Components using a Name

A. True
B. False
Answer: A. True

---

4. Which three methods would demarcate a transaction?

A. begin(), commit(), end()
B. begin(), end(), rollback()
C. begin(), end(), explode()
D. begin(), commit(), rollback()
Answer: D. begin(), commit(), rollback()[4][5]

---

5. In JNDI Directory Service is about Identifying Components using their Capability

A. True
B. False
Answer: B. False

---

6. Roles are tied to which of the following?

A. Systems and users
B. Users and privileges
C. Systems and databases
D. Privileges and systems
Answer: B. Users and privileges

---

7. When using a deployment descriptor (web.xml), which directory is that file placed in?

A. htdocs
B. WEB-INF
C. WEB-INIT
D. htdocs/WEB-INIT
Answer: B. WEB-INF

---

8. Which class is used to forward process to another URI?

A. RequestForwarder
B. RequestForward
C. RequestDispatch
D. RequestDispatcher
Answer: D. RequestDispatcher

---

9. Which of the following pieces can a JSP be broken down into?

A. Static data such as HTML
B. JSP directives
C. JSP scripting elements and variables
D. A servlet class
Answer: A, B, C, D (All)

---

10. What does JSTL stand for?

A. Java String Tag Library
B. Java Sting and Text Library

C. Java Server Paged Standard Tag Library
D. Java Standard Text Library
Answer: C. Java Server Paged Standard Tag Library

---

11. Which bracket syntax is used to define an expression?

A. <%! ... %>
B. <%= ... %>
C. <% ... %>
D. <%@ ... %>
Answer: B. <%= ... %>

---

12. Which of the following examples show the proper use of the include directive?

A. <%@ page file="page.html" %>
B. <%@ include src="page.html" %>
C. <%@ include file="page.html" %>
D. <%@ page include="page.html" %>
Answer: C. <%@ include file="page.html" %>

---

13. Select three benefits of JavaServer Faces.

A. Object-oriented approach
B. Enhanced database access
C. Powerful page navigation
D. Advanced type conversion and validation
Answer: A, C, D

---

14. JSF supports internationalization through what kind of file?

A. XML
B. Text
C. JSON
D. Properties
Answer: D. Properties

---

15. What are the three characteristics of a stateless session bean?

A. It retains client-specific information.
B. It does not retain client-specific information.

C. A client may not get the same bean instance.
D. A large number of requests can be handled by the same bean.
Answer: B, C, D

---

16. An entity is a lightweight persistence domain object that represents:

A. A relational database
B. A table in a relational database
C. Entity beans in EJB 2.x specification
D. Persistence data in a file
Answer: B. A table in a relational database

---

17. Where is the location of the datasource defined?

A. web.xml
B. the persistence context
C. the persistence unit defined by web.xml
D. the persistence unit defined by persistence.xml
Answer: D. the persistence unit defined by persistence.xml

---

18. The EntityManager.flush() method:

A. Retrieves the rows in a database table into the entity
B. Reattaches an entity to the persistence context
C. Synchronizes the state of the entity in the EntityManager’s persistence context with the database
Answer: C. Synchronizes the state of the entity in the EntityManager’s persistence context with the database

---

19. As defined in JSR-181, the important annotations used to annotate a Java class as a Web service are:

A. @SOAPBinding
B. @Resource
C. @WebParam
D. @EJB
E. @WebService
Answer: A, C, E

---

20. Which web service technology relies on HTTP methods to define operations?

A. RIA
B. WS-I
C. SOAP
D. REST
Answer: D. REST

---

21. Which of the following web service standards does JAX-WS use primarily?

A. REST
B. SOAP
C. HTTP
D. XML-RPC
Answer: B. SOAP

---

22. What is JMS?

A. Java Management Service, an API for managing Java EE services
B. Java Mobile Service, an API for developing applications for mobile devices
C. Java Macintosh Service, an API for developing applications for Macintosh computers
D. Java Messaging Service, an API for creating, sending, receiving, and reading messages
Answer: D. Java Messaging Service, an API for creating, sending, receiving, and reading messages

---

23. Which of the following does a publish/subscribe architecture have?

A. A topic and a single subscriber
B. A message and multiple copies
C. A topic and multiple messages
D. A topic and multiple subscribers
Answer: D. A topic and multiple subscribers

---

24. Which two lifecycle events are generated by message-driven beans?

A. PostConstruct
B. UnConstruct
C. PreDestroy
D. postDestroy
Answer: A. PostConstruct, C. PreDestroy

---

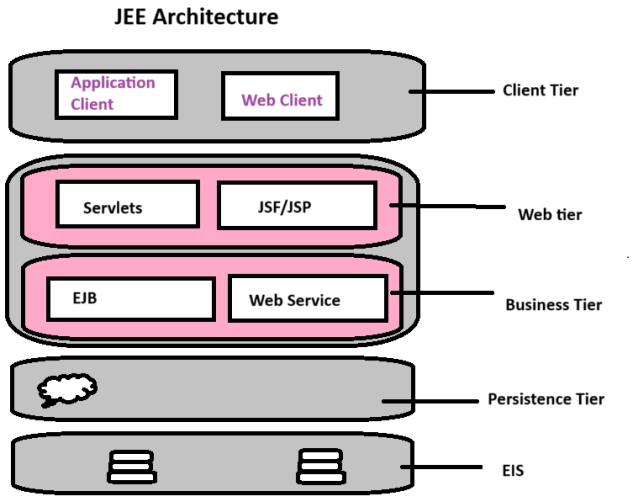
25. Atomicity can be defined as:

A. Row in a table
B. A single SQL statement
C. A single object in a container
D. A number of transactions that succeed or fail as a unit
Answer: D. A number of transactions that succeed or fail as a unit

Section II [Descriptive Questions] : 5 Marks Each

1. Explain the Java EE Architecture With diagrammatic explanation on the different Layers and Its Components.

Answer:



1. Client Tier
This is the front-end layer where users interact with the application. It includes:
Application Clients: Standalone Java applications that access business logic directly.
Web Clients: Typically, web browsers that request and display web pages.
Other Clients: Mobile apps or other remote clients.
The client tier sends requests to the server and displays responses, but it contains minimal business logic.

2. Web (Presentation) Tier
This tier handles user interface and interaction logic. It processes client requests and generates dynamic content. Key components include:
Servlets: Java classes that handle HTTP requests and responses.
JavaServer Pages (JSP): Used to create dynamic web content by embedding Java code in HTML.
JavaServer Faces (JSF): A component-based UI framework for building web interfaces.
Web Services: Provide standardized interfaces for communication with other applications.
This tier validates user input, manages sessions, and forwards requests to the business tier.

3. Business (Logic) Tier
This tier contains the core application logic and business rules. Components here include:
Enterprise JavaBeans (EJBs): Stateless or stateful session beans that implement business processes and transactions.
Message-Driven Beans (MDBs): Handle asynchronous messaging.
Web Services: Also, can be part of business logic for integration.
This tier processes data enforces business rules, and coordinates between the web tier and data tier.

4. Persistence Layer
This layer manages data persistence and retrieval. It uses:
Java Persistence API (JPA): For object-relational mapping and managing entity lifecycle.
JDBC (Java Database Connectivity): Low-level API to connect and execute SQL queries on databases.
This layer abstracts database operations and ensures data integrity and consistency.

5. Enterprise Information System (EIS) Tier
The backend tier responsible for data storage and integration with external systems. It includes Databases like Relational (Oracle, MySQL) or NoSQL databases.
This tier ensures reliable data storage and access, supporting the business tier with persistent data

2. Create Simple Web Application Containing Serviets and Business Component Show the Interaction Between them…

Answer:

```
@WebServlet("/validateEmail")
public class CustomerServlet extends HttpServlet {

    private CustomerManagementService customerService;

    @Override
```

```
    public void init() throws ServletException {
        // Initialize the business service
        customerService = new CustomerManagementService();
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        String email = request.getParameter("email");
        boolean isValid = customerService.isValidEmail(email);

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        out.println("<html><body>");
        if (email == null || email.isEmpty()) {
            out.println("<h3>Please provide an email address.</h3>");
        } else if (isValid) {
            out.println("<h3>Email '" + email + "' is valid.</h3>");
        } else {
            out.println("<h3>Email '" + email + "' is invalid.</h3>");
        }
        out.println("</body></html>");
    }
}


public class CustomerManagementService {
    public boolean isValidEmail(String email) {
        if (email == null || email.isEmpty()) {
            return false;
        }
        // Simple validation logic
        return email.contains("@") && email.contains(".");
    }
}
```

3. Explain MVC Type-I Architecture with suitable example in JSP [JSP Usebean] and Breif About Custom Tags in JSP

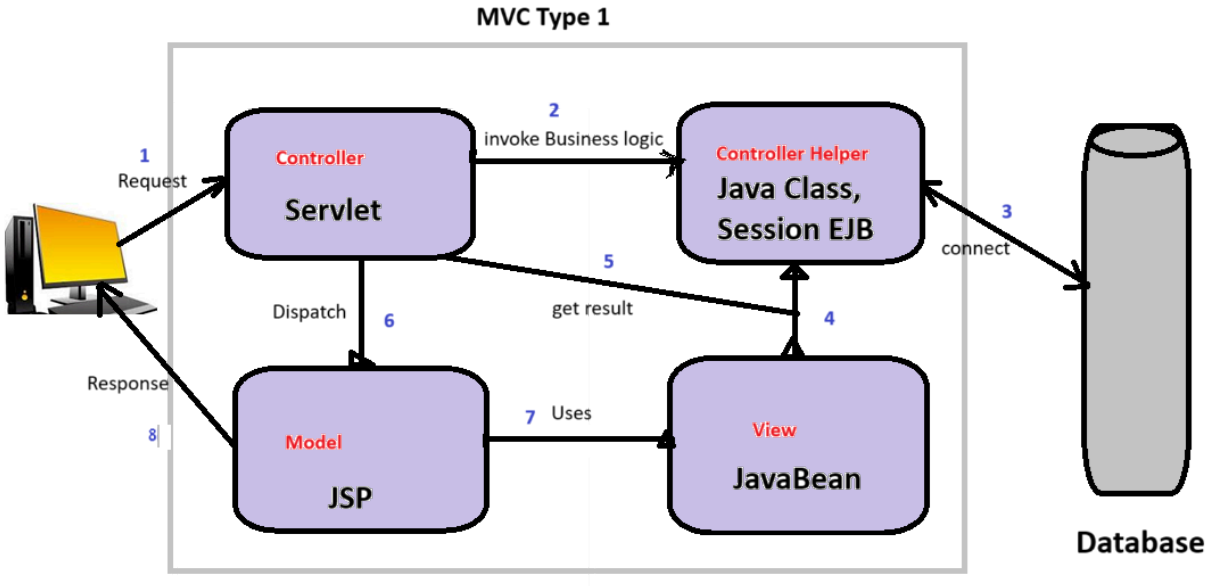
Answer:

MVC (Model-View-Controller) Type-I architecture is a design pattern that separates an application into three interconnected layers:

Model: Represents the business logic and data (e.g., JavaBeans or POJOs).
View: The presentation layer, typically JSP pages, responsible for displaying data.
Controller: Manages the flow of the application, handling user requests and responses.

In Type-I MVC, the JSP acts both as the Controller and View. The browser sends a request to the JSP, which processes the request, interacts with the business component (JavaBean), and generates the response.

Steps:
1. Browser Request: User submits a form or sends a request to a JSP page.
2. JSP Page (Controller + View): Receives the request. Uses <jsp:useBean> tag to instantiate or access a JavaBean (business component). Calls methods on the bean to process data or retrieve information. Uses JSP scriptlets or expression language to display results.
3. JavaBean (Model): Contains business logic or data storage.
4. Data Store: (Optional) JavaBean may interact with a database or other datastore.
5. Response: JSP generates HTML output sent back to the browser.



Example Using JSP <jsp:useBean>
1. JavaBean (Business Component):

java
```
// File: UserBean.java
public class UserBean {
    private String username;
    private String email;

    // Getters and setters
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }

    // Business logic method
    public String validateUser() {
        if (username != null && !username.isEmpty() && email != null && email.contains("@")) {
            return "Valid user: " + username;
        } else {
            return "Invalid user data";
        }
    }
}
```
2. JSP Page (Controller + View):

text
```
<%@ page language="java" contentType="text/html; charset=UTF-8" %>
<html>
<head><title>User Validation</title></head>
<body>

<jsp:useBean id="user" class="UserBean" scope="request" />
<jsp:setProperty name="user" property="*" />

<h2>User Validation Result</h2>
```

```
<p>
   <%= user.validateUser() %>
</p>

<form method="post" action="userValidation.jsp">
   Username: <input type="text" name="username" /><br/>
   Email: <input type="text" name="email" /><br/>
   <input type="submit" value="Validate" />
</form>

</body>
</html>
```

Explanation:
<jsp:useBean> creates or accesses the UserBean instance.
<jsp:setProperty> automatically sets bean properties from request parameters.
The JSP calls validateUser() method to perform business logic and display the result.
The form submits back to the same JSP, continuing the cycle.

Brief About Custom Tags in JSP
1. Custom tags are user-defined JSP tags that encapsulate reusable functionality, making JSP pages cleaner and easier to maintain. They are packaged in tag libraries and used like standard HTML tags. Custom tags help separate presentation from logic by moving complex code out of JSP scriptlets into reusable components.
2. They are packaged in tag libraries (.tld files) and can be used like standard HTML tags.
3. Custom tags can perform complex tasks such as formatting, iteration, conditional processing, or database access without embedding Java code directly in JSP.
Example: JSTL (Java Standard Tag Library) provides common tags for iteration (<c:forEach>), conditionals (<c:if>), and more.

Benefits of Custom Tags:
Promote separation of concerns by moving logic out of JSP.
Improve readability and maintainability of JSP pages.
Allow non-Java developers to work on JSP pages using simple tag syntax.

4. Explain JMS Architecture and demonstrate a usage of Queues and Topics. What is MDB ?

Answer:

A) JMS Architecture:
Java Message Service (JMS) is a Java EE API that enables asynchronous, reliable, and loosely coupled communication between distributed components through messaging. The JMS architecture consists of:

1) JMS Provider: The messaging middleware that manages message delivery.

2) JMS Clients: Applications that produce and consume messages.

3) Administered Objects: Preconfigured objects like ConnectionFactory (to create connections) and Destination (queues or topics).

4) Messages: Data packets exchanged between clients.

JMS supports two messaging models:
I) Point-to-Point (Queues):
Messages are sent to a queue and consumed by exactly one receiver. This model ensures one-to-one communication where each message is processed once.
Components: Queue, Message Producer, Message Consumer.

II) Publish/Subscribe (Topics):
Messages are published to a topic and delivered to multiple subscribers. This enables one-to-many communication, where all subscribers receive a copy of each message.
Components: Topic, Publisher, Subscribers.

Usage of Queues and Topics
Queue Example (Point-to-Point):
* A producer sends an order message to a queue.
* One consumer receives and processes the order message.

Topic Example (Publish/Subscribe):
* A news publisher sends breaking news messages to a topic.
* Multiple subscribers (e.g., news apps) receive the news simultaneously.

What is MDB (Message-Driven Bean)?
A Message-Driven Bean (MDB) is a server-side component in Java EE that acts as an asynchronous message listener for JMS messages. MDBs automatically consume messages from queues or topics without the need for explicit polling. They simplify asynchronous processing by handling message reception and business logic invocation within the EJB container.
MDBs are stateless and support concurrent message processing.
They integrate seamlessly with JMS, allowing scalable and reliable message-driven applications.

5. What is Packaging and Deploying in Java EE and Do Explain the Different Ways of Deployment Options.

Answer:

Packaging in Java EE is the process of assembling application components and resources into standard archive files that can be deployed on Java EE-compliant servers. Deployment is the act of installing these packaged applications onto the server so they can run and be accessed by users.

A) Packaging in Java EE
Java EE applications are packaged into three main types of archives:

1) JAR (Java Archive):
Used mainly for EJB modules containing enterprise beans and related classes. It has a .jar extension.
2) WAR (Web Archive):
Contains web components like Servlets, JSPs, HTML, JavaScript, and web deployment descriptors. It has a .war extension.
3) EAR (Enterprise Archive):
A higher-level archive that bundles one or more JAR and WAR modules together, along with deployment descriptors. It has a .ear extension and represents the entire enterprise application.
Each archive includes deployment descriptors (XML files) that specify configuration details like security, transaction management, and resource references. These descriptors can be overridden or supplemented by annotations in the source code.

B) Deploying in Java EE
Deployment involves copying the packaged archive (JAR, WAR, or EAR) to the application server and configuring it for execution. The server reads the deployment descriptors and annotations to set up resources, security, and lifecycle management.

Different Ways of Deployment
1) Standalone Module Deployment:
    Deploy a single module (EJB JAR or Web WAR) independently.
    Useful for modular development or updates.
2) EAR Deployment:
    Deploy the entire enterprise application as an EAR file containing multiple modules.
    Ensures all components are deployed together with consistent configuration.
3) Partial Deployment / Incremental Deployment
    Deploy or update only parts of the application, such as a single EJB module or web module, without redeploying the entire EAR.
    Supported by some servers and IDEs for faster development cycles.
4) Using IDE Tools:
    Integrated Development Environments like NetBeans or Eclipse provide automated packaging and deployment to registered servers.
    Simplifies build and deploy processes.
5) Command-Line or Admin Console Deployment:
    Manually deploy archives using server-specific command-line tools or web-based admin consoles Useful for production environments and scripting.