T.Y. BSC. (IT): Sem. V (Elective-I)

Enterprise Java

Mumbai University Examination Paper Solution: Nov-22

Q1. Attempt any three of the following [15]

Q1(a) What is an Enterprise Application? What is the enterprise edition of java? [5]

Ans.

- An enterprise application is computing model in web environment. An enterprise application is basically a business application. Enterprise Java refers to Java applications written for enterprises.
- A way to bring in different elements of enterprise application.
  - Web interface design
  - Transaction processing
  - Meeting non-functional system requirements
  - Availability, reliability, enhance ability, performance, scalability, reusability ,interoperability.
  - Timely development and deployment.
- J2EE defines the standard for developing multitier enterprise applications.
- It stands for Java 2 Platform Enterprise Edition.
- J2EE is a platform-independent, Java-centric environment from Sun/Oracle for developing, building and deploying Web-based enterprise applications online.
- The J2EE platform consists of a set of services, APIs, and protocols that provide the functionality for developing multi-tiered, Web-based applications.
- **It simplifies enterprise applications by:**
  - Standardized modular components.

- Providing a complete set of services to those modular components.
- Automatically handling many details of application behavior without complex programming.

Q1(b) Define java EE containers with the various java container types [5]

Ans.

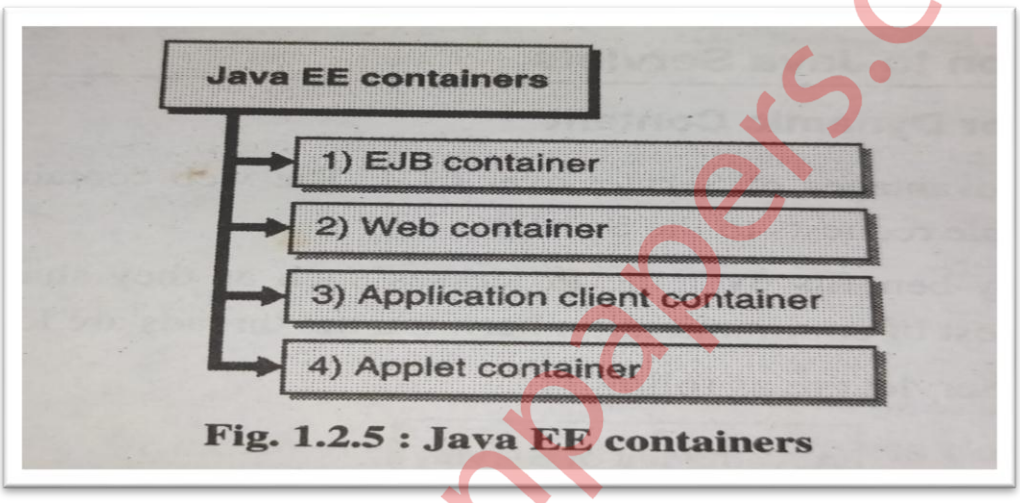


Fig. 1.2.5 : Java EE containers

- Containers are the interface between a component and the low-level, platform-specific functionality that supports the component.
- **Following are the container available:**
  - **EJB container**:

    Manages the execution of enterprise beans for Java EE applications. Enterprise beans and their container run on the Java EE server.
  - **Web container**:

    Manages the execution of web pages, servlets, and some EJB components for Java EE applications. Web components and their container run on the Java EE server.
  - **Application client container:**

Manages the execution of application client components. Application clients and their container run on the client.

- **Applet container:**

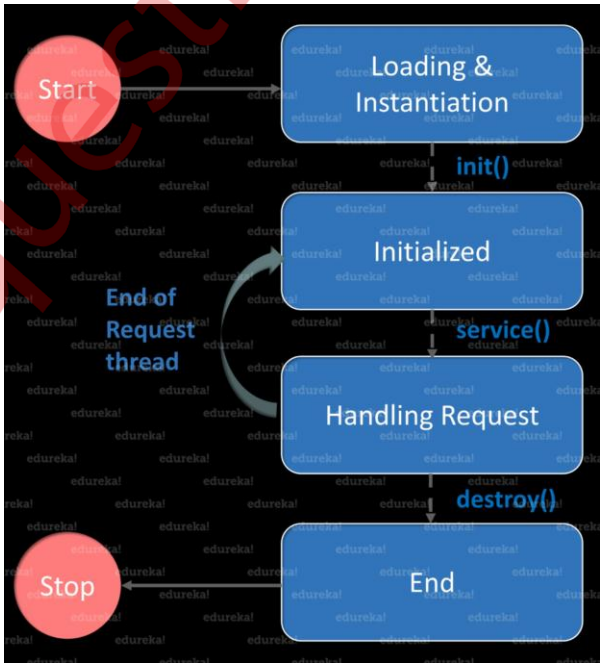
  Manages the execution of applets. Consists of a web browser and a Java Plug-in running on the client together.

  Containers provide services for:

  - Security
  - Transaction
  - Persistence
  - Concurrency
  - Availability
  - Lifecycle Management

Q1(c) Explain the life cycle of servlet application **[5]**

Ans.



The life cycle of a servlet application consists of four phases:

1**. Initialization**:

When a servlet container initializes a servlet, it creates an instance of the servlet and calls its int() method. This phase is executed only once during the lifetime of a servlet.

2. Request Handling:

Once the servlet is initialized, it is ready to handle client requests. When a client sends a request to the servlet, the servlet container creates a new thread to handle the request and calls the service() method of the servlet. The service() method processes the request and generates a response.

3. Response Generation:

After the service() method has finished processing the request, it generates a response and sends it back to the client.

4. Destroy:

When the servlet container decides to remove a servlet, it calls the destroy() method of the servlet. This phase is executed only once during the lifetime of a servlet. During the destroy() method, the servlet can release any resources it has allocated and perform any cleanup operations.

Q1(d) Write a short note on javax.servlet.servlet package [5]

Ans.

- The javax.servlet package is a part of the Java Servlet API, which is a set of classes and interfaces that define how a Java application can respond to requests from web clients.
- The javax.servlet package contains classes and interfaces that are used to implement servlets, which are Java classes that can dynamically generate web content.

- The most important classes in the javax.servlet package are the ServletRequest and ServletResponse classes, which define the objects that a servlet uses to communicate with clients.
- The ServletRequest class represents the request that a client sends to a servlet, while the ServletResponse class represents the response that a servlet sends back to the client.
- These classes provide methods for accessing information about the request, such as the HTTP headers and parameters, and for generating the response, such as writing data to the response stream.
- Another important class in the javax.servlet package is the ServletContext class, which represents the context in which a servlet runs.
- The ServletContext provides a way for servlets to share information with each other, such as database connections or configuration data.
- The ServletContext also provides methods for accessing resources, such as files or other servlets, that are available in the web application.
- The javax.servlet package also includes classes for handling HTTP sessions, cookies, and filters.
- The HttpSession class provides a way for a servlet to store data that is associated with a specific client session, while the Cookie class provides a way for a servlet to store data in a client's browser.
- The Filter class provides a way for a servlet to intercept requests and responses before they are processed by the servlet.

Q1(e) What is JDBC statement object Explain its three types [5]

Ans.

- JDBC (Java Database Connectivity) is a Java API that provides a standard interface for accessing relational databases. The Statement interface is a sub-interface of the JDBC Statement interface, which is used to execute SQL statements against a database. The Statement object is used to send SQL statements to the database and receive the results.
- **There are three types of JDBC Statement objects:**

  1. Statement:

  This is the simplest type of Statement object, which is used to execute a single SQL statement that doesn't have any parameters. The Statement object is created using the createStatement() method of the Connection interface.

  2. PreparedStatement:

  This type of Statement object is used to execute a precompiled SQL statement that may have one or more parameters. The PreparedStatement object is created using the prepareStatement() method of the Connection interface.

  3. CallableStatement:

  This type of Statement object is used to execute stored procedures that are defined in the database. A stored procedure is a set of SQL statements that are stored in the database and can be executed by a client application. The CallableStatement object is created using the prepareCall() method of the Connection interface.

Q1(f) List and explain each of the four JDBC driver types [5]

Ans.

- **JDBC-ODBC Bridge Driver**

In a Type 1 driver, a JDBC bridge is used to access ODBC drivers installed on each client machine. Using ODBC, requires configuring on your system a Data Source Name (DSN) that represents the target database.

- **JDBC-Native API**

  In a Type 2 driver, JDBC API calls are converted into native C/C++ API calls, which are unique to the database. These drivers are typically provided by the database vendors and used in the same manner as the JDBC-ODBC Bridge. The vendor specific driver must be installed

- **JDBC-Net pure Java**

  In a Type 3 driver, a three-tier approach is used to access databases. The JDBC clients use standard network sockets to communicate with a middleware application server. The socket information is then translated by the middleware application server into the call format required by the DBMS, and forwarded to the database server.

- **100% Pure Java (Thin Driver)**

  In a Type 4 driver, a pure Java-based driver communicates directly with the vendor's database through socket connection. This is the highest performance driver available for the database and is usually provided by the vendor itself.

Q2. Attempt any three of the following **[15]**

Q2(a) Explain two methods of request dispatch interface **[5]**

Ans.

- The Request Dispatcher interface dispatching the request to another resource it may be html, servlet .
- This interface also be used to include the content of another resource.

- It is one of the way of servlet collaboration.
- **There are two methods defined in the RequestDispatcher interface.**

1. forward(ServletRequest request, ServletResponse response):

This method forwards the request and response objects to another resource for processing. The resource can be a servlet, JSP page, or any other resource that can process the request and generate a response.

The forward() method is typically used when the requested resource cannot handle the request and needs to be forwarded to another resource for processing.

2. include(ServletRequest request, ServletResponse response):

This method includes the response generated by another resource in the response sent to the client.

The resource can be a servlet, JSP page, or any other resource that can generate a response. The include() method is typically used when the requested resource needs to include the output generated by another resource in its own response.

Q2(b)Where are cookies used ? Describe any four important methods of cookie class [5]

Ans.

- Cookies are used to store small amounts of data on the client side, typically in the form of key-value pairs.
- Cookies are sent from the server to the client in the HTTP response headers, and then sent back to the server in subsequent requests in the HTTP request headers.
- Cookies are commonly used for session management, user preferences, and tracking user activity.
- **The Cookie class in Java provides several methods for working with cookies:**

1. Cookie(String name, String value):

This constructor creates a new cookie with the specified name and value.

2. setMaxAge(int expiry):

This method sets the maximum age of the cookie in seconds. A value of 0 or less indicates that the cookie should be deleted immediately. A negative value indicates that the cookie should be deleted when the browser is closed.

3. setPath(String path):

This method sets the path for which the cookie is valid.

The default value is the path of the current request URI.

4. setSecure(boolean secure):

This method sets the secure flag for the cookie. If set to true, the cookie will only be sent over a secure (HTTPS) connection.

Q2(c) What is session ? Explain session management rules. [5]

Ans.

- A session is a way to maintain state information between HTTP requests for a single user or client.
- Sessions are typically used to store user-specific data, such as shopping cart items or login credentials, that needs to persist across multiple requests.
- Session management involves the creation, maintenance, and destruction of sessions.
- **The following are some rules for session management:**

1. **Session ID generation**:

A unique session ID should be generated for each new session. This ID should be difficult to guess or predict to prevent session hijacking attacks.

2. Session ID storage:

The session ID should be stored in a cookie or in the URL. If stored in a cookie, the cookie should be marked as secure and HttpOnly to prevent cross-site scripting (XSS) attacks.

3. Session timeout:

Sessions should have a timeout period to prevent them from being active indefinitely. The timeout period should be long enough to allow users to complete their tasks, but short enough to prevent sessions from being hijacked.

4. Session data storage:

Session data should be stored on the server side, not on the client side. This prevents users from tampering with the session data and allows the server to control the lifetime of the session.

5. Session termination:

Sessions should be terminated when the user logs out or when the session timeout period is reached. This ensures that session data is not left active for longer than necessary.

Q2(d) What is Non-blocking input and output ?How it works? [5]

Ans.

- Non-blocking I/O is a technique that allows a program to perform I/O operations without blocking the execution of the program.
- Non-blocking I/O is useful in situations where the program needs to perform I/O operations on multiple input/output streams simultaneously.
- In non-blocking I/O, the program initiates an I/O operation and then continues executing other code without waiting for the I/O operation to complete.

- When the I/O operation is complete, the program is notified and can then process the data that was read or written.
- Non-blocking I/O works by using a special system call that allows the program to check if an I/O operation can be performed without blocking.
- If an I/O operation can be performed without blocking, the program can initiate the operation and continue executing other code.
- If an I/O operation would block, the program can choose to wait for the operation to complete or continue executing other code.
- Non-blocking I/O is often used in network programming to handle multiple connections simultaneously. By using non-blocking I/O, a program can process data from multiple network connections without blocking on any one connection.
- This can improve the performance of network applications and allow them to handle more connections simultaneously.

Q2(e) Write a servlet code to download a file. [5]

Ans.

```
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class FileDownloadServlet extends HttpServlet {
```

```
    private static final String FILE_DIRECTORY = "/path/to/your/files/"; // Replace with the actual
path of your files
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        String fileName = request.getParameter("file");
        if (fileName == null || fileName.isEmpty()) {
            response.sendError(HttpServletResponse.SC_BAD_REQUEST, "File name is missing");
            return;
        }
        File file = new File(FILE_DIRECTORY + fileName);
        if (!file.exists()) {
            response.sendError(HttpServletResponse.SC_NOT_FOUND, "File not found");
            return;
        }
        // Set the content type and attachment header.
        response.setContentType("application/octet-stream");
        response.setHeader("Content-Disposition", "attachment; filename=\"" + fileName + "\"");
        // Read the file and write it to the response output stream.
        try (FileInputStream fis = new FileInputStream(file)) {
            byte[] buffer = new byte[4096];
            int bytesRead;
            while ((bytesRead = fis.read(buffer)) != -1) {
```

```
            response.getOutputStream().write(buffer, 0, bytesRead);
        }
    }
}
```

Q2(f) Which things needs to be carefully checked and understood while writing file uploading code in servlet? [5]

Ans.

- When writing file uploading code in a servlet, there are several things that need to be carefully checked and understood to ensure that the code is secure and reliable:

1. Request method:

The file upload request must use the POST method, as other methods like GET or HEAD do not support file uploads.

2. Encoding type:

The encoding type of the request must be set to "multipart/form-data", which allows binary data to be included in the request body.

3. File size:

The maximum file size that can be uploaded should be limited to prevent denial of service attacks and to ensure that the server has enough resources to handle the request.

4. File type:

The file type should be validated to ensure that only allowed file types are uploaded. This can be done by checking the file extension or by checking the MIME type of the file.

5. File name:

The file name should be sanitized to prevent directory traversal attacks. The file name should also be checked for length to prevent buffer overflow attacks.

6. File storage:

The uploaded file should be stored in a secure location on the server, and the file should be given a unique name to prevent conflicts with other files.

7. Error handling:

The servlet should handle errors gracefully and provide informative error messages to the user. This can include checking for file upload errors, file size errors, and file type errors.

8. Security:

The servlet should be protected against cross-site scripting (XSS) and cross-site request forgery (CSRF) attacks.

Q3. Attempt any three of the following [15]

Q3(a) What is the use of java server pages? Give difference between JSP and servlets [5]

Ans.

- **Server pages are used for following:**

1) Extension to Servlet:

JSP technology is the extension to Servlet technology. We can use all the features of the Servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP that makes JSP development easy.

2) Easy to maintain:

The business logic with presentation logic is separated in JSP so we can easily managed web application.In Servlet technology, we mix our business logic with the presentation logic.

3) Fast Development:

No need to recompile and redeploy .If JSP page is modified, we don't need to recompile and redeploy the project. The Servlet code needs to be updated and recompiled if we have to change the look and feel of the application.

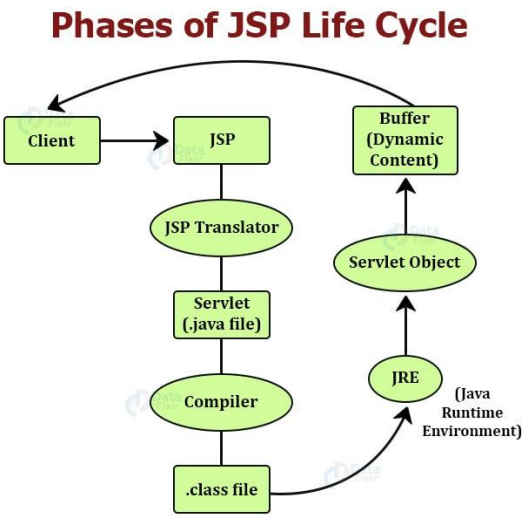
4) Less code than Servlet:

In JSP, we can use many tags such as action tags, JSTL, custom tags, etc. that reduces the code. Moreover, we can use EL, implicit objects, etc.

JSP	SERVLET
JSP is an interface on top of servlet and it is "translated" into java code.	Servlet is server-side program.
JSP run slower.	Servlet run faster.
In MVC architecture JSP acts as view.	In MVC architecture act as controller.
It is easy to code in JSP	Coding is little more difficult in servlet
JSP compiled into a java servlet before execution.	Servlet execute inside a web server

Q3(b) Explain life cycle of JSP page **[5]**

Ans.



- **The life cycle of a JSP page is as follows:**

1. Translation:

When a JSP page is accessed for the first time, it is translated into a servlet by the JSP container. The JSP container creates a servlet class that extends the HttpServlet class and contains the code for the JSP page.

2. Compilation:

The servlet class is then compiled into bytecode by the Java compiler.

3. Initialization:

The JSP container creates an instance of the servlet class and calls its init() method. This method is used to perform any initialization tasks that need to be done before the servlet can handle requests.

4. Request handling:

When a client sends a request to the JSP page, the JSP container calls the service() method of the servlet class. This method handles the request and generates the response.

5. Destruction:

When the JSP container shuts down or the JSP page is removed, the container calls the destroy() method of the servlet class. This method is used to perform any cleanup tasks that need to be done before the servlet is destroyed.

6. Recompile:

If the JSP page is modified, the JSP container recompiles it into a servlet and replaces the old servlet with the new one. This allows the JSP page to be updated without restarting the server.

Q3(c) What are directives in JSP? Explain its types? [5]

Ans.

- Directives in JSP are special instructions that are used to provide information to the JSP container about how to translate a JSP file into a servlet. There are three types of directives in JSP:

1. Page directive:

The page directive is used to provide information about the JSP page to the JSP container. It is placed at the top of the JSP file and begins with the <%@ symbol. The page directive can be used to specify attributes such as the content type, language, error page, and session behavior.

2. Include directive:

The include directive is used to include the contents of another file in the JSP page. It is placed in the body of the JSP file and begins with the <%@ symbol. The include directive can be used to include static files such as HTML or text files.

3. Taglib directive:

The taglib directive is used to define a custom tag library that can be used in the JSP page. It is placed at the top of the JSP file and begins with the <%@ symbol. The taglib directive specifies the URI of the tag library and the location of the tag library descriptor file.

- Overall, directives in JSP are used to provide important information to the JSP container and to include external files or custom tag libraries in the JSP page.

Q3(d) Give an explanation of the jsp:useBean action tag's attributes and usage [5]

Ans.

- The jsp:useBean action tag is used to create an instance of a JavaBean and store it as a variable in the page context, session context, or request context
- . **The attributes of the jsp:useBean tag are as follows:**

1. id:

This attribute is used to specify the name of the variable that will store the instance of the JavaBean.

2. class:

This attribute is used to specify the fully qualified class name of the JavaBean.

3. scope:

This attribute is used to specify the scope of the variable that will store the instance of the JavaBean. The possible values for this attribute are page, request, session, and application.

4. type:

This attribute is used to specify the data type of the variable that will store the instance of the JavaBean. This is an optional attribute and is only used when the type of the variable cannot be inferred from the class attribute.

- The jsp:useBean tag is typically used in conjunction with the jsp:setProperty and jsp:getProperty tags to set and get the properties of the JavaBean.
- For example, the following code creates an instance of the Customer JavaBean and sets its properties:

  <jsp:useBean id="customer" class="com.example.Customer" scope="session"/>

  <jsp:setProperty name="customer" property="firstName" value="John"/>

  <jsp:setProperty name="customer" property="lastName" value="Doe"/>
- The customer variable is then stored in the session context and can be accessed by other JSP pages or servlets.

Q3(e) Write short note on <JSP:plugin> Action [5]

Ans.

- The <jsp:plugin> action tag is used to embed applets or other types of plugins in a JSP page.
- The <jsp:plugin> tag has several attributes that can be used to specify the parameters and properties of the plugin, including the codebase, code, type, width, height, and archive attributes.
- For example, the following code embeds an applet in a JSP page using the <jsp:plugin> tag:

  <jsp:plugin type="applet" code="com.example.Applet" width="400" height="300">

```
        <jsp:params>
            <jsp:param name="param1" value="value1"/>
            <jsp:param name="param2" value="value2"/>
        </jsp:params>
    </jsp:plugin>
```

- This code embeds an applet with the class name "com.example.Applet" and a width and height of 400 and 300 pixels, respectively.
- The <jsp:params> tag is used to specify the parameters that will be passed to the applet.
- The <jsp:plugin> tag is a powerful tool for embedding plugins in a JSP page, but it should be used with caution as plugins can pose security risks.
- It is important to ensure that plugins are up-to-date and that they are only used when necessary.

Q3(f) What exactly is JSTL? Describe Xpath in detail. [5]

Ans.

- JSTL stands for JavaServer Pages Standard Tag Library, which is a collection of JSP tags that encapsulate core functionality common to many JSP applications.
- JSTL provides a set of tags that can be used to perform common tasks such as looping over collections, formatting data, and conditional processing.
- JSTL is designed to simplify JSP development by providing a standard set of tags that can be used across different JSP containers.
- XPath is a language used for selecting nodes from an XML document.

- XPath is used to navigate through elements and attributes in an XML document and is used by many XML-related technologies, including XSLT, XQuery, and XML Schema. XPath uses a path-like syntax to identify nodes in an XML document.
- XPath expressions are used to select nodes in an XML document.
- XPath expressions can be used to select nodes based on their name, value, or position in the document.
- XPath expressions are evaluated against an XML document and return a set of nodes that match the expression.
- XPath expressions can use a variety of operators and functions to select nodes.

Q4. Attempt any three of the following **[15]**

Q4(a) What is EJB? Explain its advantages **[5]**

Ans.

- EJB stands for Enterprise JavaBeans, which is a server-side component architecture used to build distributed, transactional, and scalable applications in Java.
- EJB provides a set of APIs for building business logic components that can be deployed on an application server and accessed by client applications. Some of the advantages of EJB include:

1. Scalability:

EJB provides a scalable architecture that can handle large volumes of transactions and users.

2. Transaction management:

EJB provides built-in support for transaction management, which simplifies the development of transactional applications.

3. Security:

EJB provides a robust security model that can be used to secure applications against unauthorized access.

4. Persistence:

EJB provides support for persistence, which allows data to be stored and retrieved from a database.

5. Reusability:

EJB components can be easily reused in different applications, which reduces development time and improves code quality.

6. Portability:

EJB components can be deployed on any application server that supports the EJB specification, which provides platform independence.

Q4(b) Write a detail note on the type of Session Bean [5]

Ans.

- Session Beans are a type of Enterprise Java Beans (EJBs) that are used to manage the business logic of an application.
- There are three types of Session Beans: Stateful Session Beans, Stateless Session Beans, and Singleton Session Beans.

1. Stateful Session Beans:

Stateful Session Beans are used to maintain the conversational state of a client. When a client interacts with a Stateful Session Bean, the bean maintains the state of the conversation between the client and the server.

This allows the client to continue the conversation with the server across multiple requests.

2. Stateless Session Beans:

Stateless Session Beans are used to perform a specific task or operation. When a client interacts with a Stateless Session Bean, the bean performs the requested operation and returns the result to the client

3. Singleton Session Beans:

Singleton Session Beans are used to maintain a single instance of a bean across multiple clients. When a client interacts with a Singleton Session Bean, the bean performs the requested operation and returns the result to the client.

Q4(c) Write a short note on Remote and Local Interface. [5]

Ans

- Remote and Local Interfaces are used in Enterprise Java Beans (EJBs) to define the methods that can be accessed by the clients of the bean.

1. Remote Interface:

- A Remote Interface is used to define the methods that can be accessed by clients that are located on a different machine or JVM (Java Virtual Machine) from the EJB.
- The methods defined in the Remote Interface are accessed through RMI (Remote Method Invocation).
- Remote Interfaces are useful in scenarios where the client and server are located on different machines, such as in a client-server application.

2. Local Interface:

- A Local Interface is used to define the methods that can be accessed by clients that are located on the same machine or JVM as the EJB.
- The methods defined in the Local Interface are accessed through direct method calls.
- Local Interfaces are useful in scenarios where the client and server are located on the same machine, such as in a web application.

Q4(d) Describe message driven beans characteristics [5]

Ans

- Message Driven Beans (MDBs) are a type of Enterprise Java Beans (EJBs) that are used to asynchronously process messages from a messaging system.
- **Here are some characteristics of MDBs:**

1. Asynchronous Processing:

MDBs are designed to process messages asynchronously. When a message is received, the container invokes the onMessage() method of the MDB to process the message. This allows the application to continue processing other requests while the message is being processed.

2. Message Driven:

MDBs are designed to process messages from a messaging system. A messaging system is a platform-independent communication mechanism that allows applications to exchange messages. MDBs can be used with messaging systems such as Java Message Service (JMS) and Advanced Message Queuing Protocol (AMQP).

3. No Client Dependency:

MDBs do not have a client dependency. This means that MDBs can be invoked by any client that sends messages to the messaging system. This allows the application to be more flexible and scalable.

4. Stateless:

MDBs are stateless. This means that the container can reuse the same instance of the MDB to process multiple messages. This reduces the overhead of creating and destroying instances of the MDB.

5. Configurable:

MDBs are configurable. The container allows you to configure the number of instances of the MDB that can be created to process messages. This allows you to optimize the performance of the application.

Q4(e)What is a naming service? [5]

Ans

- A naming service provides a way to overcome these difficulties by providing a centralized directory of services that clients can use to locate the services they need.
- The naming service maps the names of services to their network addresses.
- Clients can use the name of a service to look up its network address in the naming service's directory.
- Once the client has the network address, it can use it to connect to the service and use it.
- The naming service acts as an intermediary between the client and the service, allowing the client to access the service without having to know its physical location.
- Naming services provide several benefits to distributed systems.

- They make it easier for clients to locate and use services, even when the services are located on different machines or have different network addresses.
- They also make it easier to add or remove services from the system, because clients can use the naming service to locate the services they need, rather than having to know their physical location.

Q4(f) Write a note on DataSource Resource Definition in java EE? [5]

Ans

- In Java EE, a DataSource Resource Definition is a way to define a connection pool for a database.
- A DataSource is a Java object that provides a connection to a database.
- By defining a DataSource Resource Definition, you can configure a connection pool that can be used by your Java EE application.
- To define a DataSource Resource Definition, you need to create a deployment descriptor file (web.xml or ejb-jar.xml) for your application.
- In this file, you define the DataSource Resource Definition using XML syntax.
- The definition includes the name of the DataSource, the type of database driver to use, and the connection properties for the database.
- Once you have defined the DataSource Resource Definition, you can use it in your Java EE application to get a connection to the database.
- You can do this by looking up the DataSource using the Java Naming and Directory Interface (JNDI).
- Once you have the DataSource, you can use it to get a connection to the database.

- Using a DataSource Resource Definition has several benefits. First, it allows you to configure a connection pool for your database, which can improve performance by reducing the overhead of creating and destroying connections.
- Second, it allows you to manage the connections to the database from a central location, which can simplify administration.
- Finally, it provides a standard way to access the database from your Java EE application, which can make your application more portable and easier to maintain.

Q5. Attempt any three of the following **[15]**

Q5(a) Explain following:

1.JOIN condition Using ON 2.Entity Listeners Using CID **[5]**

Ans.

1. JOIN condition using ON:

- In SQL, a JOIN operation is used to combine rows from two or more tables based on a related column between them.
- The JOIN condition is used to specify the relationship between the tables. The ON keyword is used to specify the condition that must be met for the JOIN to occur.
- The ON keyword is used in the following syntax:

```
SELECT column_name(s)
FROM table1
JOIN table2
ON table1.column_name = table2.column_name;
```

- Here, the JOIN keyword is used to combine rows from table1 and table2 based on a related column, and the ON keyword is used to specify the condition that must be met for the JOIN to occur.

2. Entity Listeners using CID:

- In Java Persistence API (JPA), an Entity Listener is a callback method that is invoked when a specific event occurs on an entity, such as when the entity is loaded, persisted, or updated.
- The CID (Callback Interface Definition) is a feature of JPA that allows you to define your own callback methods in an interface, which can be implemented by your entity classes.
- To use Entity Listeners with CID, you need to define an interface that contains the callback methods you want to use.
- The interface should extend the javax.persistence.Callback interface, which defines the standard callback methods.
- You can then annotate your entity classes with the @EntityListeners annotation, specifying the interface you defined.

Q5(b) Where does java persistence API fit In? [5]

Ans.

- Java Persistence API (JPA) is a specification for object-relational mapping (ORM) in Java.
- It provides a set of interfaces and annotations that allow Java developers to map Java objects to relational database tables and vice versa.

- JPA is part of the Java EE (Enterprise Edition) platform, which provides a set of APIs and services for developing and deploying enterprise applications.
- JPA is used in conjunction with other Java EE technologies, such as Servlets, JavaServer Pages (JSP), Enterprise JavaBeans (EJB), and Java Transaction API (JTA), to build scalable, transactional, and distributed applications.
- JPA provides a standard way to interact with relational databases, which makes it easier for developers to switch between different persistence providers (such as Hibernate, EclipseLink, or OpenJPA) without changing their application code.
- JPA also provides a set of features for managing transactions, caching, and concurrency control, which can help developers to write efficient and robust applications.

Q5(c) Why is there need for Object relational mapping? [5]

Ans.

- Object-relational mapping (ORM) is used to map data between an object-oriented programming language (such as Java or Python) and a relational database management system (RDBMS).
- ORM provides a way to represent a database table as a class in an object-oriented language, and vice versa.
- ORM is needed for several reasons.
- First, it allows developers to work with objects in their programming language, rather than dealing with low-level SQL queries.
- This makes it easier to write and maintain code, as developers can use the object-oriented features of their language to encapsulate data and behavior.

- .Third, ORM can help to improve performance by providing caching and lazy loading mechanisms.
- ORM frameworks can cache frequently accessed data in memory, reducing the number of database queries required.
- Additionally, ORM can use lazy loading to only load data from the database when it is needed, reducing the amount of data that needs to be transferred from the database to the application.

Q5(d) Write short note on functions in JPQL and downcasting in JPQL [5]

Ans.

- Functions in JPQL (Java Persistence Query Language) are used to perform various operations on the data retrieved from the database.
- JPQL provides a set of built-in functions that can be used to perform common operations such as string manipulation, mathematical calculations, date and time operations, and more.
- Some common JPQL functions include CONCAT, SUBSTRING, TRIM, LOWER, UPPER, LENGTH, ABS, SQRT, and CURRENT_DATE.
- These functions can be used in JPQL queries to transform or manipulate data before it is returned to the application.
- **Downcasting in JPQL:**
  - Downcasting in JPQL is used to convert an object of a superclass to an object of a subclass.

- This is useful when you want to retrieve only the objects of a particular subclass from the database, but the database stores objects of both the superclass and the subclass.
- To perform downcasting in JPQL, you can use the TYPE operator.
- The TYPE operator returns the type of an entity or a relationship attribute.
- You can use the TYPE operator in a JPQL query to retrieve only the objects of a particular subclass.

Q5(e) What is Hibernate? Explain the features of Hibernate [5]

Ans.

- Hibernate is an open-source object-relational mapping (ORM) framework for the Java programming language.
- It provides a way to map Java objects to relational database tables, and vice versa. Hibernate is widely used in enterprise applications to simplify database programming and improve performance.
- **Some of the key features of Hibernate include:**

1. Object-relational mapping:

Hibernate provides a way to map Java objects to relational database tables, and vice versa. This allows developers to work with objects in their programming language, rather than dealing with low-level SQL queries.

2. Database independence:

Hibernate provides a database-independent way to work with the database. This means that you can write your application code once, and it will work with any database that Hibernate supports.

3. Caching:

Hibernate provides a caching mechanism that can be used to improve performance. Hibernate can cache frequently accessed data in memory, reducing the number of database queries required.

4. Lazy loading:

Hibernate supports lazy loading, which means that it only loads data from the database when it is needed. This can help to reduce the amount of data that needs to be transferred from the database to the application.

5. Transactions:

Hibernate provides a way to manage database transactions in a consistent and reliable way. This helps to ensure that the data in the database remains consistent and accurate.

Q5(f) Explain the components of Hibernate configuration [5]

Ans.

The main components of Hibernate configuration are:

1. SessionFactory:

The SessionFactory is responsible for creating and managing sessions. It is a thread-safe object that should be created once per application. The SessionFactory is used to create new sessions, which are used to interact with the database.

2. Database connection details:

Hibernate needs to know how to connect to the database. This includes details such as the JDBC driver class name, the database URL, the username, and the password.

3. Mapping files:

Hibernate uses mapping files to map Java objects to database tables. These mapping files define how each Java class is mapped to a database table, and how each property of the Java class is mapped to a column in the database table.

4. Hibernate configuration properties:

Hibernate has many configuration properties that can be used to customize its behavior. These properties can be used to configure things such as the cache provider, the transaction manager, and the dialect used to generate SQL statements.

5. Hibernate query language (HQL) configuration:

HQL is a powerful query language that is used to retrieve data from the database. Hibernate configuration includes settings for HQL, such as the dialect used to generate SQL statements.