**J2EE-2017**

**Q-1[A]ONE MARK:**

1]what is thin clients?

a. A thin client is a computer that runs from resources stored on a central server instead of a localized hard drive. Thin clients work by connecting remotely to a server-based computing environment where most applications, sensitive data, and memory, are stored.

2]what is JSP?

a. Jakarta Server Pages (JSP; formerly JavaServer Pages) is a collection of technologies that helps software developers create dynamically generated web pages based on HTML, XML, SOAP, or other document types.

3]full form EJB- Enterprise Java Bean

4]full form CORBA- The Common Object Request Broker Architecture (CORBA)

5]what is XML?

A, The web. xml file is the deployment descriptor file for a web application. It's a J2EE standard XML document that configures the contents of a WAR file. You can override any of the default servlet parameters at the Web application level by specifying corresponding <init-param> elements in the J2EE standard web.

6]full form RMI- Remote Method Invocation

7]what is servlet?

a. Servlets are the Java programs that runs on the Java-enabled web server or application server. They are used to handle the request obtained from the web server, process the request, produce the response, then send response back to the web server.

8]business logic is resides at which tier in 3 tier architecture

a.middle tier

9]the stub is wich side proxy of the remote object?

a. client-side proxy

10]what is the name of Hibernates own query language?

a.

11]which API is the used for distributed transaction application

a. The Java™ Transaction API (JTA)

12]which JDBC driver type can be used in either applet or servlet code?

a. Both Type 3 and Type 4

13]what is a spring?

a. Spring Framework is a Java platform that provides comprehensive infrastructure support for developing Java applications. Spring handles the infrastructure so you can focus on your application.

14]full form IoC- Inversion of Control (IoC)

15]full form AOP- aspect-oriented programming (AOP)

16]list out basic component of struts framework

* a. **JSP** Programs (View Layer Resources)
* FormBean Class(**Java** Class) (Controller Layer Resources)
* Action Servlet(Built-in Controller Servlet) (Controller Layer Resources)
* Action Class (**Java** Class) (Controller Layer Resources)
* web.xml (Deployment Descriptor file of web application)
* Struts Configuration File (XML File) (Controller Layer Resources)

17]list out sequence of layer in hibernate architecture

* a. Java application **layer**.
* **Hibernate** framework **layer**.
* Backhand api **layer**.
* Database **layer**.

18]what is entity bean?

a. Entity bean represents the persistent data stored in the database. It is a server-side component.

19]which provides communication platform between client and components?

a. Java Message Service

20]an object another computer known as-

**Q-2[A]TWO MARK:**

1]scope of JSP variables

a.

2]JDBC driver types

a. Java Database Connectivity (JDBC) is an application programming interface (API) for the programming language Java, which defines how a client may access any kind of tabular data, especially relational database.

**Type-1 driver**

Type-1 driver or JDBC-ODBC bridge driver uses ODBC driver to connect to the database. The JDBC-ODBC bridge driver converts JDBC method calls into the ODBC function calls. Type-1 driver is also called Universal driver because it can be used to connect to any of the databases.

* As a common driver is used in order to interact with different databases, the data transferred through this driver is not so secured.
* The ODBC bridge driver is needed to be installed in individual client machines.
* Type-1 driver isn’t written in java, that’s why it isn’t a portable driver.
* This driver software is built-in with JDK so no need to install separately.
* It is a database independent driver.

**Type-2 driver**

The Native API driver uses the client -side libraries of the database. This driver converts JDBC method calls into native calls of the database API. In order to interact with different database, this driver needs their local API, that’s why data transfer is much more secure as compared to type-1 driver.

* Driver needs to be installed separately in individual client machines
* The Vendor client library needs to be installed on client machine.
* Type-2 driver isn’t written in java, that’s why it isn’t a portable driver
* It is a database dependent driver.

**Type-3 driver**

The Network Protocol driver uses middleware (application server) that converts JDBC calls directly or indirectly into the vendor-specific database protocol. Here all the database connectivity drivers are present in a single server, hence no need of individual client-side installation.

* Type-3 drivers are fully written in Java, hence they are portable drivers.
* No client side library is required because of application server that can perform many tasks like auditing, load balancing, logging etc.
* Network support is required on client machine.
* Maintenance of Network Protocol driver becomes costly because it requires database-specific coding to be done in the middle tier.
* Switch facility to switch over from one database to another database.

**Type-4 driver**

Type-4 driver is also called native protocol driver. This driver interact directly with database. It does not require any native database library, that is why it is also known as Thin Driver.

* Does not require any native library and Middleware server, so no client-side or server-side installation.
* It is fully written in Java language, hence they are portable drivers.

3]explain stub and skeleton

* a. **Stub** − A stub is a representation (proxy) of the remote object at client. It resides in the client system; it acts as a gateway for the client program.
* **Skeleton** − This is the object which resides on the server side. **stub** communicates with this skeleton to pass request to the remote object.

4]explain ResultSetMetaData

a. Java ResultSetMetaData Interface

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| The metadata means data about data i.e. we can get further information from the data.  If you have to get metadata of a table like total number of column, column name, column type etc. , ResultSetMetaData interface is useful because it provides methods to get metadata from the ResultSet object.  Commonly used methods of ResultSetMetaData interface   |  |  | | --- | --- | | **Method** | **Description** | | public int getColumnCount()throws SQLException | it returns the total number of columns in the ResultSet object. | | public String getColumnName(int index)throws SQLException | it returns the column name of the specified column index. | | public String getColumnTypeName(int index)throws SQLException | it returns the column type name for the specified index. | | public String getTableName(int index)throws SQLException | it returns the table name for the specified column index. | |

5]what is java bean

a. A JavaBean is a Java class that should follow the following conventions:

* It should have a no-arg constructor.
* It should be Serializable.
* It should provide methods to set and get the values of the properties, known as getter and setter methods.

Why use JavaBean?

According to Java white paper, it is a reusable software component. A bean encapsulates many objects into one object so that we can access this object from multiple places. Moreover, it provides easy maintenance.

How to access the JavaBean class?

To access the JavaBean class, we should use getter and setter methods.

6]explain session beans

a. 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).

Types of Session Bean:-

There are 3 types of session bean.

1) Stateless Session Bean: It doesn't maintain state of a client between multiple method calls.

2) Stateful Session Bean: It maintains state of a client across multiple requests.

3) Singleton Session Bean: One instance per application, it is shared between clients and supports concurrent access.

**Q-2[B]THREE MARK:**

1]explain type of web container

a. Web Container is a java application that controls servlet. Servlet does not have a main() method, So they require a container to load them.It is used to host web applications.

2]describe validate() and reset() method

a. Validate() Method:  
- This method is used to validate the properties after they are explored by the application.  
- Validate method is Called before FormBean is handed to Action.  
- This method returns a collection of ActionError.  
Syntax :public ActionErrors validate(ActionMapping mapping,HttpServletRequest request)Reset() Method:  
- This method is called by the Struts Framework with each request that uses the defined ActionForm.  
- Used to reset all the data from the ActionForm.Syntax :public void reset() {}

Reset() Method: this method is called by the Struts Framework with each request that uses the defined ActionForm.

* Used to reset all the data from the ActionForm
* Syntax of Reset() Method:  
  public void reset() {}

3]explain different types of beans

a. EJB is an acronym for enterprise java bean. It is a specification provided by Sun Microsystems to develop secured, robust and scalable distributed applications.

There are 3 types of enterprise bean in java.

* Session Bean

Session bean contains business logic that can be invoked by local, remote or webservice client.

* Message Driven Bean

Like Session Bean, it contains the business logic but it is invoked by passing message.

* Entity Bean

It encapsulates the state that can be persisted in the database. It is deprecated. Now, it is replaced with JPA (Java Persistent API).

4explain databse metadta

a. DatabaseMetaData interface provides methods to get meta data of a database such as database product name, database product version, driver name, name of total number of tables, name of total number of views etc.

Commonly used methods of DatabaseMetaData interface:-

public String getDriverName()throws SQLException: it returns the name of the JDBC driver.

public String getDriverVersion()throws SQLException: it returns the version number of the JDBC driver.

public String getUserName()throws SQLException: it returns the username of the database.

public String getDatabaseProductName()throws SQLException: it returns the product name of the database.

public String getDatabaseProductVersion()throws SQLException: it returns the product version of the database.

public ResultSet getTables(String catalog, String schemaPattern, String tableNamePattern, String[] types)throws SQLException: it returns the description of the tables of the specified catalog. The table type can be TABLE, VIEW, ALIAS, SYSTEM TABLE, SYNONYM etc.

How to get the object of DatabaseMetaData:

The getMetaData() method of Connection interface returns the object of DatabaseMetaData. Syntax:

public DatabaseMetaData getMetaData()throws SQLException

5]life cycle of servlet

a. The web container maintains the life cycle of a servlet instance. Let's see the life cycle of the servlet:

1. Servlet class is loaded.
2. Servlet instance is created.
3. init method is invoked.
4. service method is invoked.
5. destroy method is invoked.



As displayed in the above diagram, there are three states of a servlet: new, ready and end. The servlet is in new state if servlet instance is created. After invoking the init() method, Servlet comes in the ready state. In the ready state, servlet performs all the tasks. When the web container invokes the destroy() method, it shifts to the end state.

1) Servlet class is loaded

The classloader is responsible to load the servlet class. The servlet class is loaded when the first request for the servlet is received by the web container.

2) Servlet instance is created

The web container creates the instance of a servlet after loading the servlet class. The servlet instance is created only once in the servlet life cycle.

3) init method is invoked

The web container calls the init method only once after creating the servlet instance. The init method is used to initialize the servlet. It is the life cycle method of the javax.servlet.Servlet interface. Syntax of the init method is given below:

public void init(ServletConfig config) throws ServletException

4) service method is invoked

The web container calls the service method each time when request for the servlet is received. If servlet is not initialized, it follows the first three steps as described above then calls the service method. If servlet is initialized, it calls the service method. Notice that servlet is initialized only once. The syntax of the service method of the Servlet interface is given below:

public void service(ServletRequest request, ServletResponse response)

throws ServletException, IOException

5) destroy method is invoked

The web container calls the destroy method before removing the servlet instance from the service. It gives the servlet an opportunity to clean up any resource for example memory, thread etc. The syntax of the destroy method of the Servlet interface is given below:

public void destroy()

6]what are the class are used in struts?

a.

>ActionServlet:

ActionServlet is the back bone of the whole web application.

>ActionForm:

ActionForm is a java bean that associates one or more ActionMappings.

>Action:

This class is extended by the org.apache.struts.action.Action class

>ActionMapping:

The ActionMapping is used to provide mappings for Objects to Actions. An action map is used in association with InputMap to locate an action at time of pressing a key.

>ActionForward:

It is represented as a destination for the controller, RequestProcessor.

**ActionServlet:**  
  
ActionServlet is the back bone of the whole web application.  
  
**ActionForm:**  
  
ActionForm is a java bean that associates one or more ActionMappings.The bean is initialized from the corresponding parameters of the requests before the Action.execute method is invoked. The bean’s validate() method will be called, before the execution of execute() method. This process is done before the bean properties have been populated. The verification of properties submitted by the user input form is validated by the validate() method. If the method finds problems, an error messages object those are encapsulated the problems will be returned and the controller servlet will return the control back to the input form. If not, the validate() method returns null, which is the indication of accepting everything, and the Action.execute method should be invoked.  
  
**Action:**  
  
This class is extended by the org.apache.struts.action.Action class. The business logic is wrapped by the Action class and this class provides an interface to the Model of the application. Action class can be viewed as a glue between the View and the Model layer of MVC architecture. The data transfer from the view layers to the specific business process layer(View to Model) is done by this class. The processed data from the business layer to view layer is returned ultimately. The struts controller i.e., ActionServlet , chooses an appropriate Action and the instance is created if necessary and invokes execute() method.  
  
**ActionMapping:**  
  
The ActionMapping is used to provide mappings for Objects to Actions. An action map is used in association with InputMap to locate an action at time of pressing a key. An action mapping can contain a reference to a form bean that the action can use. A reference to a form bean is contained by an action mapping. The reference to a form is used by an action.  
  
**ActionForward:**  
  
It is represented as a destination for the controller, RequestProcessor, that might be directed a RequestDispatcher.forward or HttpServletResponse.sendRedirect to. This is the result of processing activities of an Action class. The instances of this classes is created dynamically as the need arises.

**Q-2[C]FIVE MARK:**

1]what is j2ee?explain j2ee features.

a. J2EE is a platform-independent, Java-centric environment from Sun 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 multitiered, Web-based applications.

The purpose of the J2EE platform is to develop multi-tier, web-based applications using a series of protocols and application programming interfaces (APIs). J2EE provides the enterprise level with a Java platform that is fast, safe and secure, improving on the J2SE (Java 2 Platform, Standard Edition).

**Java™ Database Connectivity (JDBC™) API**

The standard way to connect to a database from a J2EE application or module is through a JDBC driver. Sun ONE Application Server supports the core JDBC 3.0 API and the JDBC 2.0 extensions and works with a wide range of JDBC Compliant™ drivers. A JDBC resource associates a JDBC driver and database to a JNDI name that applications and modules can reference.

**Transaction Service**

The purpose of a transaction is to ensure that data is updated in an all-or-nothing fashion in order to preserve data integrity. The transaction service provides transactional resource managers for the JDBC API, the JMS API, and resource adapters (connector modules). In the Sun ONE Application Server, you can configure transactions and reference them using the JNDI API.

**Java Naming and Directory Interface™ (JNDI) API**

The JNDI API allows application components and clients to look up distributed resources, services, and EJB™ components. The J2EE resources described in this guide are made available through the JNDI API. External JNDI resources and custom resources are also configurable in the Sun ONE Application Server.

**JavaMail™ API**

The JavaMail API allows J2EE applications to create, send, receive, and read mail messages. The JavaMail API includes support for the IMAP4, POP3, and SMTP mail protocols. JavaMail sessions are made available through the JNDI API.

2]architecture of j2ee

a. **J2EE Architecture**

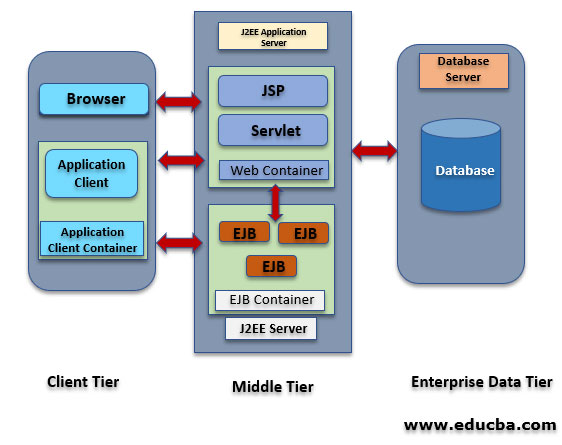
J2EE Uses Three Tiers:

Client Tier: Client tier consists of user programs that interact with the user for request and response.

Middle Tier: Middle tier usually contains enterprise beans and web services that distribute business logic for the applications.

Enterprise Data Tier: Enterprise data is stored in a relational database. This tier contains containers, components and services.

**Representation of J2EE Architecture**



Usually in J2EE architecture consist of four tiers Client Tier, Web Tier, Enterprise JavaBean Tier, and Enterprise Information Tier. The middle tier consists of Web Tier+EJB tier.

**1. Client Tier**

The client tier consists of programs or applications interact with the user. Usually, they are located in a different machine from the server. Client tier prompts the user inputs into user requests then forwarded to the J2EE server then processed result returned back to the client. A client can be a web browser, standalone application or server that runs on a different machine.

Clients can be classified as a Web Client and Application Client.

**Web Client**

Web client consists of dynamic web pages of various mark-up languages that are generated by web components running in web tier or web browser which renders pages received from the server. Web clients are also called as thin clients that usually do not perform things like query database, execute business rules. When using thin client-heavy operations are offloaded to enterprise beans executing in the J2EE server.

**Applets:**Web pages received from web tier embedded an [Applet these run](https://www.educba.com/applets-in-java/) on a web browser. Web components are APIs for creating a web client program. Web components enable the user to design cleaner and more modular applications. They provide a way to separate application programming.

**Application Clients**

The application client runs on the client machine and handles the tasks that give richer user interfaces. Typically the GUI is created by [Swings or AWT](https://www.educba.com/awt-vs-swing/). Application clients can directly access EJBs running in the business tier using an HTTP connection.

**2. Middle Tier (Web tier & EJB Tier)**

Below are the components of Middle Tier:

**Web Tier /Web Component**

Web components can be servlets or JSP pages. [Servlets can dynamically](https://www.educba.com/what-is-servlet/) process the request and generate the responses. Compared to [JSP and servlets](https://www.educba.com/jsp-vs-servlet/) – servlets are dynamic pages to some extent but JSP pages are static in nature.

During application assembly process Client’s Static HTML programs and applet, codes are bundled in web tier/ Web Component. Actually these HTML and applets are not considered as elements of web components. Server-side utility classes are also bundled with web component but they are not considered as web components.

Web tier might include EJB components for processing user inputs and sends the input to Enterprise bean running in the business tier.

**EJB Tier /EJB Component**

Enterprise components handle usually business code that is logic to solve particular business domains such as banking or finance are handled by enterprise bean running in the business tier.

Enterprise Container receives data from client processes if necessary, sends it to the enterprise information system for storage. Enterprise bean also retrieves data from storage, processes it and sends it back to the client.

**Three kinds of beans:**

* **Session Bean:** Session bean is used for a conversation with the client. Once the client finishes the execution session bean destroys.
* **Entity Bean**: Holds the particular data stored in a database. Once the server shutdowns or client finishes its execution entity bean data is preserved.
* **Message Driven Bean:** Message bean combines the properties of Session Bean and JMS. Which benefits the business component to receive messages asynchronously.

**3. Enterprise Information System**

This tier consists of database servers, enterprise resource planning systems and other data sources. Resources are typically located on a separate machine than the J2EE Server and accessed by components on the business tier.

**Technologies used in EIS Tier:**

* Java Database Connectivity API (JDBC).
* [Java Persistence API](https://www.educba.com/java-persistence-api/).
* Java Connector Architecture.
* Java Transaction API.

3]JSP LIFE CYCLE

A. JSP technology is used to create web application just like Servlet technology. It can be thought of as an extension to Servlet because it provides more functionality than servlet such as expression language, JSTL, etc.

A JSP page consists of HTML tags and JSP tags.

>Translation of JSP page to Servlet :-

This is the first step of JSP life cycle. This translation phase deals with Syntactic correctness of JSP. Here test.jsp file is transllated to test.java.

>Compilation of JSP page :-

Here the generated java servlet file (test.java) is compiled to a class file (test.class).

>Classloading :-

Servlet class which has been loaded from JSP source is now loaded into container.

>Instantiation :-

Here instance of the class is generated. The container manages one or more instance by providing response to requests.

>Initialization :-

jspInit() method is called only once during the life cycle immediately after the generation of Servlet instance from JSP.

>Request processing :-

\_jspService() method is used to serve the raised requests by JSP.It takes request and response object as parameters.This method cannot be overridden.

>JSP Cleanup :-

In order to remove the JSP from use by the container or to destroy method for servlets jspDestroy()method is used. This method is called once, if you need to perform any cleanup task like closing open files, releasing database connections jspDestroy() can be overridden.

4]explain scripting element of JSP with example

A. The scripting elements provides the ability to insert java code inside the jsp. There are three types of scripting elements:

* scriptlet tag
* expression tag
* declaration tag

### JSP scriptlet tag

A scriptlet tag is used to execute java source code in JSP. Syntax is as follows:

1. <%  java source code %>

### Example of JSP scriptlet tag that prints the user name

In this example, we have created two files index.html and welcome.jsp. The index.html file gets the username from the user and the welcome.jsp file prints the username with the welcome message.

*File: index.html*

1. **<html>**
2. **<body>**
3. **<form** action="welcome.jsp"**>**
4. **<input** type="text" name="uname"**>**
5. **<input** type="submit" value="go"**><br/>**
6. **</form>**
7. **</body>**
8. **</html>**

*File: welcome.jsp*

1. <html>
2. <body>
3. <%
4. String name=request.getParameter("uname");
5. out.print("welcome "+name);
6. %>
7. </form>
8. </body>
9. </html>

# JSP expression tag

The code placed within **JSP expression tag** is written to the output stream of the response. So you need not write out.print() to write data. It is mainly used to print the values of variable or method.

### Syntax of JSP expression tag

1. **<**%=  statement %**>**

### Example of JSP expression tag that prints user name

To display the current time, we have used the getTime() method of Calendar class. The getTime() is an instance method of Calendar class, so we have called it after getting the instance of Calendar class by the getInstance() method.

In this example, we are printing the username using the expression tag. The index.html file gets the username and sends the request to the welcome.jsp file, which displays the username.

*File: index.jsp*

1. **<html>**
2. **<body>**
3. **<form** action="welcome.jsp"**>**
4. **<input** type="text" name="uname"**><br/>**
5. **<input** type="submit" value="go"**>**
6. **</form>**
7. **</body>**
8. **</html>**

*File: welcome.jsp*

1. **<html>**
2. **<body>**
3. **<**%= "Welcome "+request.getParameter("uname") %**>**
4. **</body>**
5. **</html>**

# JSP Declaration Tag

The **JSP declaration tag** is used to declare fields and methods.

The code written inside the jsp declaration tag is placed outside the service() method of auto generated servlet.

So it doesn't get memory at each request

#### Syntax of JSP declaration tag

The syntax of the declaration tag is as follows:

1. **<**%!  field or method declaration %**>**

### Example of JSP declaration tag that declares method

In this example of JSP declaration tag, we are defining the method which returns the cube of given number and calling this method from the jsp expression tag. But we can also use jsp scriptlet tag to call the declared method.

### index.jsp

1. **<html>**
2. **<body>**
3. **<**%!
4. int cube(int n){
5. return n\*n\*n\*;
6. }
7. %**>**
8. **<**%= "Cube of 3 is:"+cube(3) %**>**
9. **</body>**
10. **</html>**

5]program

**Q-3[A]TWO MARK:**

1]features of hibernate

a. Hibernate is a Java framework that simplifies the development of Java application to interact with the database. It is an open source, lightweight, ORM (Object Relational Mapping) tool. Hibernate implements the specifications of JPA (Java Persistence API) for data persistence.

1) Open Source and Lightweight

Hibernate framework is open source under the LGPL license and lightweight.

2) Fast Performance

The performance of hibernate framework is fast because cache is internally used in hibernate framework. There are two types of cache in hibernate framework first level cache and second level cache. First level cache is enabled by default.

3) Database Independent Query

HQL (Hibernate Query Language) is the object-oriented version of SQL. It generates the database independent queries. So you don't need to write database specific queries. Before Hibernate, if database is changed for the project, we need to change the SQL query as well that leads to the maintenance problem.

4) Automatic Table Creation

Hibernate framework provides the facility to create the tables of the database automatically. So there is no need to create tables in the database manually.

5) Simplifies Complex Join

Fetching data from multiple tables is easy in hibernate framework.

6) Provides Query Statistics and Database Status

Hibernate supports Query cache and provide statistics about query and database status.

2]inversion of control in spring

a.

3]sevelet api.

a. The javax.servlet and javax.servlet.http packages represent interfaces and classes for servlet api.

The javax.servlet package contains many interfaces and classes that are used by the servlet or web container. These are not specific to any protocol.

The javax.servlet.http package contains interfaces and classes that are responsible for http requests only.

4]explain callable statement with example

a. CallableStatement interface is used to call the stored procedures and functions.

We can have business logic on the database by the use of stored procedures and functions that will make the performance better because these are precompiled.

Suppose you need the get the age of the employee based on the date of birth, you may create a function that receives date as the input and returns age of the employee as the output.

5]list out types of ejb

a. There are 3 types of enterprise bean in java.

Session Bean

Session bean contains business logic that can be invoked by local, remote or webservice client.

Message Driven Bean

Like Session Bean, it contains the business logic but it is invoked by passing message.

Entity Bean

It encapsulates the state that can be persisted in the database. It is deprecated. Now, it is replaced with JPA (Java Persistent API).

6]what is structs?

a. Struts is used to create a web applications based on servlet and JSP. Struts depend on the MVC (Model View Controller) framework. Struts application is a genuine web application. Struts are thoroughly useful in building J2EE (Java 2 Platform, Enterprise Edition) applications because struts takes advantage of J2EE design patterns. Struts follows these J2EE design patterns including MVC.

**Q-3[B]THREE MARK:**

1]AOP terminologies

a. Aspect Oriented Programming (AOP) compliments OOPs in the sense that it also provides modularity. But the key unit of modularity is aspect than class.

AOP breaks the program logic into distinct parts (called concerns). It is used to increase modularity by cross-cutting concerns.

A cross-cutting concern is a concern that can affect the whole application and should be centralized in one location in code as possible, such as transaction management, authentication, logging, security etc.

Why use AOP?

It provides the pluggable way to dynamically add the additional concern before, after or around the actual logic.

Where use AOP?

AOP is mostly used in following cases:

* to provide declarative enterprise services such as declarative transaction management.
* It allows users to implement custom aspects.

2]explain mesaage driven beans

a. A message driven bean (MDB) is a bean that contains business logic. But, it is invoked by passing the message. So, it is like JMS Receiver.

MDB asynchronously receives the message and processes it.

A message driven bean receives message from queue or topic, so you must have the knowledge of JMS API.

Message Driven Bean Example

To create the message driven bean, you need to declare @MessageDriven annotation and implement MessageListener interface.

In eclipse ide, create **EJB Project** then create a class as given below:

*File: MyListener.java*

1. **package** com.javatpoint;
2. **import** javax.ejb.MessageDriven;
3. **import** javax.jms.\*;
5. @MessageDriven(mappedName="myTopic")
6. **public** **class** MyListener **implements** MessageListener{
7. @Override
8. **public** **void** onMessage(Message msg) {
9. TextMessage m=(TextMessage)msg;
10. **try**{
11. System.out.println("message received: "+m.getText());
12. }**catch**(Exception e){System.out.println(e);}
13. }
14. }

Export the ejb project and deploy the application.

In **glassfish server**, click on **applications** -> **deploy** -> select mdb jar file by **Choose File** -> **OK**.

3]what is annotation in hibernate

a. The hibernate application can be created with annotation. There are many annotations that can be used to create hibernate application such as @Entity, @Id, @Table etc.

Hibernate Annotations are based on the JPA 2 specification and supports all the features.

All the JPA annotations are defined in the **javax.persistence** package. Hibernate EntityManager implements the interfaces and life cycle defined by the JPA specification.

The core advantage of using hibernate annotation is that you don't need to create mapping (hbm) file. Here, hibernate annotations are used to provide the meta data.

4]explain session v/s cookie in jsp

a. **Session**

A session creates a file in a temporary directory on the server where registered session variables and their values are stored. This data will be available to all pages on the site during that visit.

A session ends when the user closes the browser or after leaving the site, the server will terminate the session after a predetermined period of time, commonly 30 minutes duration.

**Cookies**

Cookies are text files stored on the client computer and they are kept of use tracking purpose. Server script sends a set of cookies to the browser. For example name, age, or identification number etc. The browser stores this information on a local machine for future use.

When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.

5]explain java bean properties

a. avaBean property is a named feature that can be accessed by the user of the object. The feature can be of any Java data type, containing the classes that you define.

A JavaBean property may be read, write, read-only, or write-only. JavaBean features are accessed through two methods in the JavaBean's implementation class:

**1. getPropertyName ()**

For example, if the property name is firstName, the method name would be getFirstName() to read that property. This method is called the accessor.

**2. setPropertyName ()**

For example, if the property name is firstName, the method name would be setFirstName() to write that property. This method is called the mutator.

6]name any three servlet API methods for session life time

a.

**Q-3[C]FIVE MARK:**

1]structs flow of control

a.

2]spring architecture

a. The Spring Framework provides about 20 modules which can be used based on an application requirement.

Core Container

The Core Container consists of the Core, Beans, Context, and Expression Language modules the details of which are as follows −

* The **Core** module provides the fundamental parts of the framework, including the IoC and Dependency Injection features.
* The **Bean** module provides BeanFactory, which is a sophisticated implementation of the factory pattern.
* The **Context** module builds on the solid base provided by the Core and Beans modules and it is a medium to access any objects defined and configured. The ApplicationContext interface is the focal point of the Context module.
* The **SpEL** module provides a powerful expression language for querying and manipulating an object graph at runtime.

Data Access/Integration

The Data Access/Integration layer consists of the JDBC, ORM, OXM, JMS and Transaction modules whose detail is as follows −

* The **JDBC** module provides a JDBC-abstraction layer that removes the need for tedious JDBC related coding.
* The **ORM** module provides integration layers for popular object-relational mapping APIs, including JPA, JDO, Hibernate, and iBatis.
* The **OXM** module provides an abstraction layer that supports Object/XML mapping implementations for JAXB, Castor, XMLBeans, JiBX and XStream.
* The Java Messaging Service **JMS** module contains features for producing and consuming messages.
* The **Transaction** module supports programmatic and declarative transaction management for classes that implement special interfaces and for all your POJOs.

Web

The Web layer consists of the Web, Web-MVC, Web-Socket, and Web-Portlet modules the details of which are as follows −

* The **Web** module provides basic web-oriented integration features such as multipart file-upload functionality and the initialization of the IoC container using servlet listeners and a web-oriented application context.
* The **Web-MVC** module contains Spring's Model-View-Controller (MVC) implementation for web applications.
* The **Web-Socket** module provides support for WebSocket-based, two-way communication between the client and the server in web applications.
* The **Web-Portlet** module provides the MVC implementation to be used in a portlet environment and mirrors the functionality of Web-Servlet module.

Miscellaneous

There are few other important modules like AOP, Aspects, Instrumentation, Web and Test modules the details of which are as follows −

* The **AOP** module provides an aspect-oriented programming implementation allowing you to define method-interceptors and pointcuts to cleanly decouple code that implements functionality that should be separated.
* The **Aspects** module provides integration with AspectJ, which is again a powerful and mature AOP framework.
* The **Instrumentation** module provides class instrumentation support and class loader implementations to be used in certain application servers.
* The **Messaging** module provides support for STOMP as the WebSocket sub-protocol to use in applications. It also supports an annotation programming model for routing and processing STOMP messages from WebSocket clients.
* The **Test** module supports the testing of Spring components with JUnit or TestNG frameworks.

3]hibernate annotation

a. The hibernate application can be created with annotation. There are many annotations that can be used to create hibernate application such as @Entity, @Id, @Table etc.

Hibernate Annotations are based on the JPA 2 specification and supports all the features.

All the JPA annotations are defined in the **javax.persistence** package. Hibernate EntityManager implements the interfaces and life cycle defined by the JPA specification.

The core advantage of using hibernate annotation is that you don't need to create mapping (hbm) file. Here, hibernate annotations are used to provide the meta data.

4]explain RMI with eg.

A. The **RMI** (Remote Method Invocation) is an API that provides a mechanism to create distributed application in java. The RMI allows an object to invoke methods on an object running in another JVM.

The RMI provides remote communication between the applications using two objects *stub* and *skeleton*.

Understanding stub and skeleton

RMI uses stub and skeleton object for communication with the remote object.

A **remote object** is an object whose method can be invoked from another JVM. Let's understand the stub and skeleton objects:

stub

The stub is an object, acts as a gateway for the client side. All the outgoing requests are routed through it. It resides at the client side and represents the remote object. When the caller invokes method on the stub object, it does the following tasks:

1. It initiates a connection with remote Virtual Machine (JVM),
2. It writes and transmits (marshals) the parameters to the remote Virtual Machine (JVM),
3. It waits for the result
4. It reads (unmarshals) the return value or exception, and
5. It finally, returns the value to the caller.

skeleton

The skeleton is an object, acts as a gateway for the server side object. All the incoming requests are routed through it. When the skeleton receives the incoming request, it does the following tasks:

1. It reads the parameter for the remote method
2. It invokes the method on the actual remote object, and
3. It writes and transmits (marshals) the result to the caller.

Java RMI Example

The is given the 6 steps to write the RMI program.

1. Create the remote interface
2. Provide the implementation of the remote interface
3. Compile the implementation class and create the stub and skeleton objects using the rmic tool
4. Start the registry service by rmiregistry tool
5. Create and start the remote application
6. Create and start the client application

5]NOTE:MVC architecture

a. MVC Pattern stands for Model-View-Controller Pattern. This pattern is used to separate application's concerns.

* **Model** - Model represents an object or JAVA POJO carrying data. It can also have logic to update controller if its data changes.
* **View** - View represents the visualization of the data that model contains.
* **Controller** - Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.