1.Discuss in detail about session beans and its
types?
Enterprise Java Beans (EJB) is one of the several Java APIs for standard manufacture of enterprise software. EJB is a server-side software element that summarizes business logic of an application. Enterprise Java Beans web repository yields a runtime domain for web related software elements including computer reliability, Java Servlet Lifecycle (JSL) management, transaction procedure and other web services. The EJB enumeration is a subset of the Java EE enumeration.
To run EJB application we need an application server (EJB Container) such as Jboss, Glassfish, Weblogic, Websphere etc. It performs:
1. Life cycle management
2. Security
3. Transaction management
4. Object pooling
Types of Enterprise Java Beans
There are three types of EJB:
1. Session Bean: Session bean contains business logic that can be invoked by local, remote or webservice client. There are two types of session beans: (i) Stateful session bean and (ii) Stateless session bean.
(i) Stateful Session bean :
Stateful session bean performs business task with the help of a state. Stateful session bean can be used to access various method calls by storing the information in an instance variable. Some of the applications require information to be stored across separate method calls. In a shopping site, the items chosen by a customer must be stored as data is an example of stateful session bean.
(ii) Stateless Session bean :
Stateless session bean implement business logic without having a persistent storage mechanism, such as a state or database and can used shared data. Stateless session bean can be used in situations where information is not required to used across call methods.
2. Message Driven Bean: Like Session Bean, it contains the business logic but it is invoked by passing message.
3. Entity Bean: It summarizes the state that can be remained in the database. It is deprecated. Now, it is replaced with JPA (Java Persistent API). There are two types of entity bean:
(i) Bean Managed Persistence :
In a bean managed persistence type of entity bean, the programmer has to write the code for database calls. It persists across multiple sessions and multiple clients.
(ii) Container Managed Persistence :
Container managed persistence are enterprise bean that persists across database. In container managed persistence the container take care of database calls.

2. Explain the protocols used in understanding Java
mail?
The JavaMail API provides a platform-independent and protocol-independent framework to build mail and messaging applications. The JavaMail API provides a set of abstract classes defining objects that comprise a mail system. It is an optional package (standard extension) for reading, composing, and sending electronic messages.
JavaMail provides elements that are used to construct an interface to a messaging system, including system components and interfaces. While this specification does not define any specific implementation, JavaMail does include several classes that implement RFC822 and MIME Internet messaging standards. These classes are delivered as part of the JavaMail class package.
Following are some of the protocols supported in JavaMail API:
SMTP: Acronym for Simple Mail Transfer Protocol. It provides a mechanism to deliver email.

POP: Acronym for Post Office Protocol. POP is the mechanism most people on the Internet use to get their mail. It defines support for a single mailbox for each user. RFC 1939 defines this protocol.

IMAP: Acronym for Internet Message Access Protocol. It is an advanced protocol for receiving messages. It provides support for multiple mailbox for each user, in addition to, mailbox can be shared by multiple users. It is defined in RFC 2060.

MIME: Acronym for Multipurpose Internet Mail Extensions. . It is not a mail transfer protocol. Instead, it defines the content of what is transferred: the format of the messages, attachments, and so on. There are many different documents that take effect here: RFC 822, RFC 2045, RFC 2046, and RFC 2047. As a user of the JavaMail API, you usually don't need to worry about these formats. However, these formats do exist and are used by your programs.

NNTP and Others:There are many protocols that are provided by third-party providers. Some of them are Network News Transfer Protocol (NNTP), Secure Multipurpose Internet Mail Extensions (S/MIME) etc.
Architecture
As said above the java application uses JavaMail API to compose, send and receive emails.The following figure illustrates the architecture of JavaMail:
The abstract mechanism of JavaMail API is similar to other J2EE APIs, such as JDBC, JNDI, and JMS. As seen the architecture diagram above, JavaMail API is divided into two main parts:

An application-independent part: An application-programming interface (API) is used by the application components to send and receive mail messages, independent of the underlying provider or protocol used.

A service-dependent part: A service provider interface (SPI) speaks the protocol-specific languages, such as SMTP, POP, IMAP, and Network News Transfer Protocol (NNTP). It is used to plug in a provider of an e-mail service to the J2EE platform.

3. Explain the role of servlets in session
management?
are the Java programs that run on the Java-enabled web server or application server. They are used to handle the request obtained from the webserver, process the request, produce the response, then send a response back to the webserver

HTTP is a "stateless" protocol, which means that each time a client requests a Web page, the client establishes a new connection with the Web server, and the server does not retain track of prior requests.

The conversion of a user over a period of time is referred to as a session. In general, it refers to a certain period of time.
The recording of the object in session is known as tracking.
Session tracking is the process of remembering and documenting customer conversions over time. Session management is another name for it.
The term "stateful web application" refers to a web application that is capable of remembering and recording client conversions over time.
Session Tracking employs Four Different techniques
Cookies
Hidden Form Field
URL Rewriting
HttpSession
Cookies:-
Cookies are little pieces of data delivered by the web server in the response header and kept by the browser. Each web client can be assigned a unique session ID by a web server. Cookies are used to keep the session going. Cookies can be turned off by the client.
Hidden Form Field

The information is inserted into the web pages via the hidden form field, which is then transferred to the server. These fields are hidden from the user's view.
4. Explain about JSP directives?
JSP directives are the elements of a JSP source code that guide the web container on how to translate the JSP page into it's respective servlet.
Syntax :
<%@ directive attribute = "value"%>
Directives can have a number of attributes which you can list down as key-value pairs and separated by commas. The blanks between the @ symbol and the directive name, and between the last attribute and the closing %>, are optional.
Different types of JSP directives :
There are three different JSP directives available. They are as follows:
Page Directives : JSP page directive is used to define the properties applying the JSP page, such as the size of the allocated buffer, imported packages and classes/interfaces, defining what type of page it is etc. The syntax of JSP page directive is as follows:
<%@page attribute = "value"%>
Different properties/attributes :
The following are the different properties that can be defined using page directive :
import: This tells the container what packages/classes are needed to be imported into the program.
Syntax:
<%@page import = "value"%>
Example :

```
<%-- JSP code to demonstrate how to use page

 directive to import a package --%>

<%@page import = "java.util.Date"%>
<%Date d = new Date();%>
<%=d%>
```

contentType: This defines the format of data that is being exchanged between the client and the server. It does the same thing as the setContentType method in servlet used to.
Syntax:-
<%@page contentType="value"%>
info: Defines the string which can be printed using the 'getServletInfo()' method.
Syntax:
<%@page info="value"%>
buffer: Defines the size of the buffer that is allocated for handling the JSP page. The size is defined in Kilo Bytes.
Syntax:
<%@page buffer = "size in kb"%>
language: Defines the scripting language used in the page. By default, this attribute contains the value 'java'.
isELIgnored: This attribute tells if the page supports expression language. By default, it is set to false. If set to true, it will disable expression language.

Syntax:
<%@page isElIgnored = "true/false"%>
errorPage: Defines which page to redirect to, in case the current page encounters an exception.
Syntax:
<%@page errorPage = "true/false"%>

5.Explain Container-managed transactions?
A transaction is a single unit of work items, which follows the ACID properties. ACID stands for Atomic, Consistent, Isolated, and Durable.
Atomic − If any of the work item fails, the whole unit will be considered failed. Success meant, all items execute successfully.
Consistent − A transaction must keep the system in consistent state.
Isolated − Each transaction executes independent of any other transaction.
Durable − Transaction should survive system failure if it has been executed or committed.
EJB Container/Servers are transaction servers and handles transactions context propagation and distributed transactions. Transactions can be managed by the container or by custom code handling in bean's code.
Container Managed Transactions − In this type, the container manages the transaction states.
Bean Managed Transactions − In this type, the developer manages the life cycle of transaction states.
Container Managed Transactions
EJB 3.0 has specified following attributes of transactions, which EJB containers implement −
REQUIRED − Indicates that business method has to be executed within transaction, otherwise a new transaction will be started for that method.
REQUIRES_NEW − Indicates that a new transaction, is to be started for the business method.
SUPPORTS − Indicates that business method will execute as part of transaction.
NOT_SUPPORTED − Indicates that business method should not be executed as part of transaction.
MANDATORY − Indicates that business method will execute as part of transaction, otherwise exception will be thrown.
NEVER − Indicates if business method executes as part of transaction, then an exception will be thrown.
Example:-

```
package com.tutorialspoint.txn.required;

import javax.ejb.*

@Stateless
@TransactionManagement(TransactionManagementType.CONTAINER)
public class UserDetailBean implements UserDetailRemote {

   private UserDetail;

   @TransactionAttribute(TransactionAttributeType.REQUIRED)
   public void createUserDetail() {
      //create user details object
   }
}
```

createUserDetail() business method is made Required using Required annotation.

```
package com.tutorialspoint.txn.required;

import javax.ejb.*

@Stateless
public class UserSessionBean implements UserRemote {

   private User;

   @EJB
   private UserDetailRemote userDetail;

   public void createUser() {
      //create user
      //...
      //create user details
      userDetail.createUserDetail();
   }
}
```

createUser() business method is using createUserDetail(). If exception occurred during createUser() call and User object is not created then UserDetail object will also not be created.

6.Explain following Messaging Models.
(a) Point-to-point messaging :-
In modern software architecture, the application needs to be decoupled, high scalability, serving high performance. Messaging service come in allows us to build that kind of application. Therefore with a solid understanding of messaging models is the key to build an effective system. There are two commonly messaging models, the point-to-point and the publish/subscribe model. Both of these messaging models are based on the message queue know as a central place to send messages to or place to get the message from. The sent messages are ordered in the message queue except it has higher priority. In basically, the message will be processed following the first-in-fist-out.
In the point-to-point model, the message sent from the message sender to only one receiver even if many message receivers are listening in the same message queue. In the point-to-point model, the terms message sender and message receiver are usually applied rather than message publisher and message consumer.
There are two types of point-to-point messaging, fire-and-forget(one-way) messaging and request/reply(request-response) messaging. The difference between fire-and-forget and request/reply is how the message sender cares about the status of the sending message.
Fire-and-forget:-
In fire-and-forget, the message sender does not wait for any response from the message queue. It doesn't care did the message queue receive the message or not. In this model, the Originator and the Recipient would have no interaction at all.
The request/reply messaging model:-
Different from the fire-and-forget mode, in the request/reply messaging model, the message sender sends a message on one queue, and then it waits for the response from the receiver. with this model, the send cares about the message status that's it received or not yet.
Publish/Subscribe Messaging model
The Publish/Subscribe Messaging or normally called pub/sub messaging is a form of asynchronous. In this domain, the message producers are called publishers and the message consumers are called subscribers. The publisher produces messages to a topic then all subscribers which subscribed to that topic will receive the sending messages and consume them. The difference between the point-to-point and publish/subscribe messaging model is how many receivers for a message. Another difference is in the point-to-point model, the message sender must know the receiver but in the publish/subscribe the message publishers do not need to know where the message will be consumed. This characteristic provides a high decoupling for the application when applying the publish/subscribe message model.
Conclusion
Through this article, I have introduced you to common messaging models and I hope that will help you have a solid understanding of messaging services that have a good choice when trying to applying it to your application.