Weblogic Administrator Interview Questions

1)How do I provide user credentials for starting a server?

When you create a domain, the Configuration Wizard prompts you to provide the username and password for an initial administrative user. If you create the domain in development mode, the wizard saves the username and encrypted password in a boot identity file. A WebLogic Server instance can refer to a boot identity file during its startup process. If a server instance does not find such a file, it prompts you to enter credentials.
If you create a domain in production mode, or if you want to change user credentials in an existing boot identity file, you can create a new boot identity file.

2)Can I start a Managed Server if the Administration Server is unavailable?

By default, if a Managed Server is unable to connect to the specified Administration Server during startup, it can retrieve its configuration by reading a configuration file and other files directly. You cannot change the server's configuration until the Administration Server is available. A Managed Server that starts in this way is running in Managed Server Independence mode.

3)What is the function of T3 in WebLogic Server?

T3 provides a framework for WebLogic Server messages that support for enhancements. These enhancements include abbreviations and features, such as object replacement, that work in the context of WebLogic Server clusters and HTTP and other product tunneling. T3 predates Java Object Serialization and RMI, while closely tracking and leveraging these specifications. T3 is a superset of Java Object. Serialization or RMI; anything you can do in Java Object Serialization and RMI can be done over T3. T3 is mandated between WebLogic Servers and between programmatic clients and a WebLogic Server cluster. HTTP and IIOP are optional protocols that can be used to communicate between other processes and WebLogic Server. It depends on what you want to do. For example, when you want to communicate between a browser and WebLogic Server-use HTTP, or an ORB and WebLogic Server-IIOP.

4)How do you set the classpath?

WebLogic Server installs the following script that you can use to set the classpath that a server requires:
WL_HOME\server\bin\setWLSEnv.cmd (on Windows)
WL_HOME/server/bin/setWLSEnv.sh (on UNIX)

5)How do stubs work in a WebLogic Server cluster?

Clients that connect to a WebLogic Server cluster and look up a clustered object obtain a replica-aware stub for the object. This stub contains the list of available server instances that host implementations of the object. The stub also contains the load balancing logic for distributing the load among its host servers.

What happens when a failure occurs and the stub cannot connect to a WebLogic Server instance?
When the failure occurs, the stub removes the failed server instance from its list. If there are no

servers left in its list, the stubb uses DNS again to find a running server and obtain a current list of running instances. Also, the stub periodically refreshes its list of available server instances in the cluster; this allows the stub to take advantage of new servers as they are added to the cluster.

6)How does a server know when another server is unavailable?

WebLogic Server uses two mechanisms to determine if a given server instance is unavailable.

Each WebLogic Server instance in a cluster uses multicast to broadcast regular "heartbeat" messages that advertise its availability. By monitoring heartbeat messages, server instances in a cluster determine when a server instance has failed. The other server instances will drop a server instance from the cluster, if they do not receive three consecutive heartbeats from that server instance

WebLogic Server also monitors socket errors to determine the availability of a server instance. For example, if server instance A has an open socket to server instance B, and the socket unexpectedly closes, server A assumes that server B is offline.

7)How are notifications made when a server is added to a cluster?

The WebLogic Server cluster broadcasts the availability of a new server instance each time a new instance joins the cluster. Cluster-aware stubs also periodically update their list of available server instances.

8)How do clients handle DNS requests to failed servers?

If a server fails and DNS continues to send requests to the unavailable machine, this can waste bandwidth. For a Java client application, this problem occurs only during startup. WebLogic Server caches the DNS entries and removes the unavailable ones, to prevent the client from accessing a failed server twice.

Failed servers can be more of a problem for browser-based clients, because they always use DNS. To avoid unnecessary DNS requests with browser-based clients, use a third-party load-balancer such as Resonate, BigIP, Alteon, and LocalDirector. These products mask multiple DNS addresses as a single address. They also provide more sophisticated load-balancing options than round-robin, and they keep track of failed servers to avoid routing unnecessary requests.

9)How many WebLogic Servers can I have on a multi-cpu machine?

There are many possible configurations and each has its own advantages and disadvantages. BEA WebLogic Server has no built-in limit for the number of server instances that can reside in a cluster. Large, multi-processor servers such as Sun Microsystems, Inc. Sun Enterprise 10000, therefore, can host very large clusters or multiple clusters.

In most cases, WebLogic Server clusters scale best when deployed with one WebLogic Server instance for every two CPUs. However, as with all capacity planning, you should test the actual deployment with your target web applications to determine the optimal number and distribution of server instances.

10)How can I set deployment order for applications?

WebLogic Server allows you to select the load order for applications. WebLogic Server deploys server-level resources (first JDBC and then JMS) before deploying applications. Applications are deployed in this order: connectors, then EJBs, then Web Applications. If the application is an EAR, the individual components are loaded in the order in which they are declared in the application.xml deployment descriptor.

What is difference between welogic server and WebSphere?

WebLogic is one of the leading J2EE™ application servers in today's marketplace. Monitoring WebLogic for its performance and availability becomes inevitable. Applications Manager, a tool for monitoring the performance and availability of applications and servers, helps in BEA WebLogic Management. WebSphere is IBM's powerful J2EE application server that enables businesses and organizations to build robust, Web-based applications. WebSphere, a transaction-oriented web server, allows you to develop, launch, and integrate powerful e-business applications—customer management systems, transaction processing, infrastructure adjustment, and many others. This high-performance server provides solutions for connecting people, systems, and applications with your internal and external resources.

In cluster , the load balancing that simply redirect the client request to any available server in weblogic server cluster. Suppose assume we have 4 managed servers and one admin server. can we trace that request is going to which managed server in the cluster? is it possible to tell that request is going to which ipaddr/managed server?

whenever the request gets routed from any Load balancer or Web-server to any application server, the routed request contains a header part which includes information of the server like Port, Listen address etc based on which it routes to its appropriate server hosting application. And we can trace this information in web-server log file.

Diff b/w managed server and non managed server? can u briefly explain?

Managed servers are the servers which have one and only one separate Admin server. Admin server manages all the configurations and resources

for managed servers. No managed server could only be Admin server.

what is digital certificate?

For the purposes of digital signing of documents, verification of digital signatures, and handling digital certificates in the Java platform, the Java

Cryptography Architecture (JCA) is used. JCA is a specification that gives the programmers a standard way to access cryptographic services, digital

signatures, and digital certificates.

The Most Important Classes in JCA

java.security.KeyStore

java.security.PublicKey
java.security.PrivateKey
java.security.Cert.Certificate

Digital certifiacte is an electronic document which uses a digital signature to bind together a public key with an identity — information such as the

name of a person or an organization, their address, and so forth. The certificate can be used to verify that a public key belongs to an individual.

what is core dump? diff b/w core dump and c dump?

Core dump consists of the recorded state of the working memory of a computer program at a specific time, generally when the program has

terminated abnormally.A copy of the data stored in the core memory of a computer, usually used for debugging purposes.

What is advantage if silent mode installation ?

for this sailent mode of installation you need to specify the log file and xml file. The difference between command and sailent mode is command

-mode=console i.e here step by step are visible but in sailent mode every thing is configured.

The biggest advantage of silent mode installation is that it is non-interactive and hence your intervention is not required during installation.

All the parameters to be used during installation are defined in xml file (usually silent.xml)

eg:./filename.bin -mode=silent -silent_xml=silent.xml

What is Connection Pool ?

Answer 1

Pool: Maintain equal instance of objects is called Pool

Connection Pool : the connection objects maintain in the form pool is called connection pool.

Advantages of Connection Pool: the Connection is very precies . application need connections getting connections from db and close connection's

it's normal process. if we use connection pool no need hit every time db , get from server(in con pool) , in the middle server will get the some amount

connections that give the connections . server will take physical connection to db , server will give to logical connection to application if need more

connections server will increase connections are maintained que process.

Answer 2

Pool: Maintain equal instance of objects is called Pool

Connection Pool : the connection objects maintain in the form pool is called connection pool.

Adv Con Pool: the Connection is very precies . application need connections getting connections from db and close.

What is the difference between Connection Pool and data source?

The Connection pool means equals instance of connections and it refereed Data Source object.

What does 'stub' mean in weblogic server?

clients that connect to a WLS instances and look like a single clustered object obtain a replica-aware-stub of the object. The stub contains the list of

all the available server instancesof the object.It also has a load balancing logic to distribute the load across the multiple hosts.
What is managed server ?

Any WebLogic Server instance apart from Administration Server is called as Managed Servers. This is weblogic server where you deploy your

application (Though you can deploy your application in Administration server as well but it is not recommended in production/UAT instance).

A weblogic server instance is called managed server.An administration have no of manager servers.

How to find the heap memory of managedServer?

1. For weblogic 8 you can get the details from /weblogic81/common/bin/startManagedWebLogic.sh Here there is a parameter set in the script with

memory details.
2. you can varify it from out file of that server by searching Xms and Xms keyword. in ant cersion of WLS.
3. use the below command .ps -auwwwx | grep .In the output you will find parameters Xms and Xmx for heap size.

in which file/script we need to change the heap size?.What is the variable name to search to

change the heap size ?

In start up script,we can change ms and mx value.we can change the heap size in Javaw.exe file, which is located in bin folder.

what is WLS T3 protocol?

Answer 1. Weblogic's implementationof the RMI specification uses a proprietary wire-protocol known as T3. javasoft's reference implementation of

RMI uses a proprietary protocol called JRMP. weblogic developed T3 because they needed a scalable,efficient protocol for building enterprise-class

destributed object systems with java.

Answer 2. T3 provides a framework for WebLogic Server messages that support for enhancements. These enhancements include abbreviations and

features, such as object replacement, that work in the context of WebLogic Server clusters and HTTP and other product tunneling. T3 predates Java

Object Serialization and RMI, while closely tracking and leveraging these specifications. T3 is a superset of Java Object. Serialization or RMI;

anything you can do in Java Object Serialization and RMI can be done over T3. T3 is mandated between WebLogic Servers and between

programmatic clients and a WebLogic Server cluster. HTTP and IIOP are optional protocols that can be used to communicate between other

processes and WebLogic Server. It depends on what you want to do. For example, when you want to communicate between a browser and

WebLogic Server-use HTTP, or an ORB and WebLogic Server-IIOP.

Answer 3. T3 protocal is used to communicate betwen the EJB's in the wl serevr instances.

Answer 4. T3 is an optimized protocol used to transport data between WebLogic Server and other Java programs,including clients and other

WebLogic Servers. WebLogic Server keeps track of every Java Virtual Machine (JVM) with which it connects, and creates a single T3 connection to

carry all traffic for a JVM.For example, if a Java client accesses an enterprise bean and a JDBC connection pool on WebLogic Server, a single

network connection is established between the WebLogic Server JVM and the client JVM. The EJB and JDBC services can be written as if they had

sole use of a dedicated network connection because the T3 protocol invisibly multiplexes packets on the single connection.

T3 is an efficient protocol for Java-to-Java applications because it avoids unnecessary network connection events and uses fewer OS resources.

The protocol also has internal enhancements that minimize packet sizes

What happens when database down in weblogic ?

Answer 1 : Connection pool staus changes to suspended.it tries to poll the connection. It depends on the configuration of connection pool .I will log

an error if
application tires to use the connection pool.

Answer 2 : using connection pool we can retrive the value in the database.

Answer # 3 : in app server u will diffine that connect to which data base according to that connection will be establised and u can get data from data

base.if the data base is not working then u will get a error page(related to database).if u want to give 100% availability for clint then u can use failover

concept and place the data in to different databases and give instructions for u r application that if one data base is fail to serve the data after

partucular time the request will rediirect to another database where data is present.

Answer # 4 : in WLSS 9.2 When DB is down the connection pool/Data source will go to SUSPENDED state waiting to re-establish the connection

with DB. when DB is up it will automaticaly re-connect with DB . but in the middle if you re-cycle the server. the server will go to ADMIN mode.

Answer # 5 : Impact only particular application which is using failure database. Mean while we need to untarget the failure database to aviod erros in

logs. Once DB back needs to target the same.

How to set Connection Pool size in Weblogic Server ?

we can set connection pool size in weblogic server through admin console.At first you have to login to the adminserver and after putting username

and password you will see connectionpool within service configuration,then you have to go through the connectionpool and there will be two tab one is

configuration and another will be monitoring.From monitoring tab you can view maxcapasity and highconnection of connectionpool(jdbc).From that

you can customize the value .

What is multicast Address?

an address that can be used to send the messages to the same host addresses but in different network addresses.that addresses can be referred

as multicast address.The multicast addresses are in the range 224.0.0.0 to 239.255.255.255.

how can we handle if server is hang? and how to handle out of memory in weblogic?

When a Server is hanging, we can ping the server using java weblogic.Admin t3://server:port PING. If the server can respond to the ping, it may be

that the application is hanging and not the server itself.

We can use -verbose gc flag to check whether the server is doing the proper garbage collection or not.

Also we can analyze the thread dumps to check out which thread are actually stucking.

where can we set classpath, which will reflect to whole domain ?

right click on my computer icon->advanced->environment
variables.here you will find to panels
1.path and
2.classpth.
1. go to path click on add
a)name: name of the path variable.
b)value: Absolute path of your bin folder or your application path.

what is the differencees between work manager and executequeue?

1.Unlike execute queues, all WorkManagers share a common thread pool and a priority-based queue.

2. The size of the thread pool is determined automatically by the kernel and resized as needed.

3.WorkManagers become very lightweight, and customers can create WorkManagers without worrying about the size of the
thread pool.

4.Thread dumps look much cleaner with fewer threads. In the new model, it is possible to specify different service-level agreements (SLAs) such as

fair shares or response-time goals for the same servlet invocation depending on the user associated with the invocation.

5.In earlier releases, each servlet or RMI request was associated with a dispatch policy that mapped to an execute queue.Requests without an

explicit dispatch policy use the server-wide default execute queue. In WebLogic Server 9.0, requests are still associated with a dispatch policy but
are mapped to a WorkManager instead of to an execute queue.

6.Execute queues are always global whereas WorkManagers are always application scoped. Even WorkManagers defined globally in the console are
application scoped during runtime. This means that each application gets into own runtime instance that is distinct from others, but all of them share

the same characteristics like fair-share goals.

7. Another advantage is Thread Count Self-tuning. One of the major differences between execute queues and the new thread scheduling model is

that the thread count does not need to be set. In earlier releases, customers defined new thread pools and configured their size to avoid deadlocks

and provide differentiated service. It is quite difficult to determine the exact number of threads needed in production to achieve optimal throughput

and avoid deadlocks. WebLogic Server 9.0 is self-tuned, dynamically adjusting the number of threads to avoid deadlocks and
achieve optimal throughput subject to concurrency constraints. It also meets objectives for differentiated service. These objectives are stated as fair

shares and response-time goals as explained in the next section.

8. Execute Queue concepts used till 8.1+ weblogic releases. Work Manager came from Weblogic 9+.

dIFFERNCE BETWEEN VERSIONS 8.X,9.X,10.X ?

There are various differences ranging from 8.X,9.X,10.X
1. Changes made in console UI.
2. The node Manager uses the port no 5555 in 8.x and in 10.x it uses 5556.
3. Every functionality should be done only after hitting LOCK AND EDIT option in the version of 10x, Where as it is not needed in the version of 8.x.
4. There is no need of creating Connection pools in 10.x . Datasource itself directs requests where as Connection pools should be created in the

version 8.x.

8.x---9.x and 10.x
1)In 8 no loc& edit option--------1)In 9 lock & edit option
2)here Execute ques are there-----2)here work managers
3)here you can use in node manager portno:-5556--3)5557

What is config.xml.booted file ?

Config.xml.booted file is a replica of config.xml file. It is stored inside the root file of the admin server.When your msi-config.xml is not running on

independent mode and your config.xml file is destroyed or currupted for any cause,then you can start your managed server by taking the

configuration from this file.

What is config.xml?

it is heart of the weblogic server and its maintains the all the info about managed and admin server details like Ipadd,portno...etc, and whenever you

update the admin console that info is updated in config.xml and while restart the server,server will get details from config.xml only and using we can

deploy the application nothing but hard deployment.

What is DataSource ?how can its associated with connection pools ?

DataSoource is an Interface that is an alternative to Driver manager.It has methods :
getConnection()
getConnection(username,pwd)
...and many more..

Through JNDI it connects with database.Or It takes from Connection pool.

1. **What is the difference between URL instance and URLConnection instance?** - A URL instance represents the location of a resource, and a URLConnection instance represents a link for accessing or communicating with the resource at the location.
2. **What are the two important TCP Socket classes?** - Socket and ServerSocket. ServerSocket is used for normal two-way socket communication. Socket class allows us to read and write through the sockets. getInputStream() and getOutputStream() are the two methods available in Socket class.
3. **What technologies are included in J2EE?** - The primary technologies in J2EE are: Enterprise JavaBeansTM (EJBsTM), JavaServer PagesTM (JSPsTM), Java Servlets, the Java Naming and Directory InterfaceTM (JNDITM), the Java Transaction API (JTA), CORBA, and the JDBCTM data access API.
4. **What is the Java Authentication and Authorization Service (JAAS) 1.0?** - The Java Authentication and Authorization Service (JAAS) provides a way for a J2EE application to authenticate and authorize a specific user or group of users to run it. JAAS is a Java programing language version of the standard Pluggable Authentication Module (PAM) framework that extends the Java 2 platform security architecture to support user-based authorization.
5. **What's the difference between JNDI lookup(), list(), listBindings(), and search()?** - lookup() attempts to find the specified object in the given context. I.e., it looks for a single, specific object and either finds it in the current context or it fails. list() attempts to return an enumeration of all of the NameClassPair's of all of the objects in the current context. I.e., it's a listing of all of the objects in the current context but only returns the object's name and the name of the class to which the object belongs. listBindings() attempts to return an enumeration of the Binding's of all of the objects in the current context. I.e., it's a listing of all of the objects in the current context with the object's name, its class name, and a reference to the object itself. search() attempts to return an enumeration of all of the objects matching a given set of search criteria. It can search across multiple contexts (or not). It can return whatever attributes of the objects that you desire. It's by far the most complex and powerful of these options but is also the most expensive.
6. **Components of JNDI** - Naming Interface- The naming interface organizes information hierarchically and maps human-friendly names to addresses or objects that are machine-friendly. It allows access to named objects through multiple

namespaces. Directory Interface - JNDI includes a directory service interface that provides access to directory objects, which can contain attributes, thereby providing attribute-based searching and schema support. Service Provider Interface - JNDI comes with the SPI, which supports the protocols provided by third parties.

7. **What is the Max amount of information that can be saved in a Session Object?** - As such there is no limit on the amount of information that can be saved in a Session Object. Only the RAM available on the server machine is the limitation. The only limit is the Session ID length(Identifier), which should not exceed more than 4K. If the data to be store is very huge, then it's preferred to save it to a temporary file onto hard disk, rather than saving it in session. Internally if the amount of data being saved in Session exceeds the predefined limit, most of the servers write it to a temporary cache on Hard disk.
8. **Must my bean-managed persistence mechanism use the WebLogic JTS driver?** - BEA recommend that you use the TxDataSource for bean-managed persistence.
9. **Do EJBs have to be homogeneously deployed across a cluster? Why?** - Yes. Beginning with WebLogic Server version 6.0, EJBs must be homogeneously deployed across a cluster for the following reasons:
   - To keep clustering EJBs simple
   - To avoid cross server calls which results in more efficiency. If EJBs are not deployed on all servers, cross server calls are much more likely.
   - To ensure that every EJB is available locally
   - To ensure that all classes are loaded in an undeployable way
   - Every server must have access to each EJB's classes so that it can be bound into the local JNDI tree. If only a subset of the servers deploys the bean, the other servers will have to load the bean's classes in their respective system classpaths which makes it impossible to undeploy the beans.
10. **Is an XSLT processor bundled in WebLogic Server?** - Yes, an XSLT processor, based on Apache's Xalan 2.0.1 processor, in WebLogic Server 6.1.
11. **I plugged in a version of Apache Xalan that I downloaded from the Apache Web site, and now I get errors when I try to transform documents. What is the problem?** - You must ensure that the version of Apache Xalan you download from the Apache Web site is compatible with Apache Xerces version 1.3.1. Because you cannot plug in a different version of Apache Xerces , the only version of Apache Xerces that is compatible with WebLogic Server 6.1 is 1.3.1. The built-in parser (based on version 1.3.1 of Apache Xerces) and transformer (based on version 2.0.1 of Apache Xalan) have been modified by BEA to be compatible with each other.
12. **How do I increase WebLogic Server memory?** - Increase the allocation of Java heap memory for WebLogic Server. (Set the minimum and the maximum to the same size.) Start WebLogic Server with the -ms32m option to increase the allocation, as in this example:

    ```
     $ java ... -ms32m -mx32m ...
    ```

    This allocates 32 megabytes of Java heap memory to WebLogic Server, which improves performance and allows WebLogic Server to handle more simultaneous connections. You can increase this value if necessary.

13. **What causes Java.io exceptions in the log file of WebLogic Server?** - You may see messages like these in the log file:

    ```
    (Windows NT)  java.io.IOException Connection Reset by Peer  java.io.EOFException
    Connection Reset by Peer (Solaris)  java.io.Exception: Broken pipe
    ```

    These messages occur when you are using servlets. A client initiates an HTTP request, and then performs a series of actions on the browser:

    - Click Stop or enter equivalent command or keystrokes
    - Click Refresh or enter equivalent command or keystrokes
    - Send a new HTTP request.

    The messages indicate that WebLogic Server has detected and recovered from an interrupted HTTP request.

14. **What is the function of T3 in WebLogic Server?** - T3 provides a framework for WebLogic Server messages that support for enhancements. These enhancements include abbreviations and features, such as object replacement, that work in the context of WebLogic Server clusters and HTTP and other product tunneling. T3 predates Java Object Serialization and RMI, while closely tracking and leveraging these specifications. T3 is a superset of Java Object. Serialization or RMI; anything you can do in Java Object Serialization and RMI can be done over T3. T3 is mandated between WebLogic Servers and between programmatic clients and a WebLogic Server cluster. HTTP and IIOP are optional protocols that can be used to communicate between other processes and WebLogic Server. It depends on what you want to do. For example, when you want to communicate between a browser and WebLogic Server-use HTTP, or an ORB and WebLogic Server-IIOP.
15. **What are the enhancements in EJB 2.0 specification with respect to Asynchronous communication?** - EJB 2.0 mandates integration between JMS and EJB. We have specified the integration of Enterprise JavaBeans with the Java Message Service, and have introduced message-driven beans. A message-driven bean is a stateless component that is invoked by the container as a result of the arrival of a JMS message. The goal of the message-driven bean model is to make developing an enterprise bean that is asynchronously invoked to handle the processing of incoming JMS messages as simple as developing the same functionality in any other JMS MessageListener.
16. **What are the enhancements in EJB 2.0 with respect to CMP?** - EJB 2.0 extends CMP to include far more robust modeling capability, with support for declarative management of relationships between entity EJBs. Developers no longer need to re-establish relationships between the various beans that make up their application — the container will restore the connections automatically as beans are loaded, allowing bean developers to navigate between beans much as they would

between any standard Java objects.
EJB 2.0 also introduces for the first time a portable query language, based on the abstract schema, not on the more complex database schema. This provides a database and vendor-independent way to find entity beans at run time, based on a wide variety of search criteria.

17. **Can you briefly describe local interfaces?** - EJB was originally designed around remote invocation using the Java Remote Method Invocation (RMI) mechanism, and later extended to support to standard CORBA transport for these calls using RMI/IIOP. This design allowed for maximum flexibility in developing applications without consideration for the deployment scenario, and was a strong feature in support of a goal of component reuse in J2EE. Many developers are using EJBs locally - that is, some or all of their EJB calls are between beans in a single container. With this feedback in mind, the EJB 2.0 expert group has created a local interface mechanism. The local interface may be defined for a bean during development, to allow streamlined calls to the bean if a caller is in the same container. This does not involve the overhead involved with RMI like marshalling etc. This facility will thus improve the performance of applications in which co-location is planned. Local interfaces also provide the foundation for container-managed relationships among entity beans with container-managed persistence.
18. **What are the special design care that must be taken when you work with local interfaces?** - It is important to understand that the calling semantics of local interfaces are different from those of remote interfaces. For example, remote interfaces pass parameters using call-by-value semantics, while local interfaces use call-by-reference. This means that in order to use local interfaces safely, application developers need to carefully consider potential deployment scenarios up front, then decide which interfaces can be local and which remote, and finally, develop the application code with these choices in mind. While EJB 2.0 local interfaces are extremely useful in some situations, the long-term costs of these choices, especially when changing requirements and component reuse are taken into account, need to be factored into the design decision.
19. **What happens if remove( ) is never invoked on a session bean?** - In case of a stateless session bean it may not matter if we call or not as in both cases nothing is done. The number of beans in cache is managed by the container. In case of stateful session bean, the bean may be kept in cache till either the session times out, in which case the bean is removed or when there is a requirement for memory in which case the data is cached and the bean is sent to free pool.
20. **What is the difference between creating a distributed application using RMI and using a EJB architecture?** - It is possible to create the same application using RMI and EJB. But in case of EJB the container provides the requisite services to the component if we use the proper syntax. It thus helps in easier development and lesser error and use of proven code and methodology. But the investment on application server is mandatory in that case. But this investment is warranted because it results in less complex and maintainable code to the client, which is what the end client wants. Almost all the leading application servers provide load balancing and performance tuning techniques. In case of RMI we have to code the services and include in the program the way to invoke these services.
21. **Why would a client application use JTA transactions?** - One possible example would be a scenario in which a client needs to employ two (or more) session beans, where each session bean is deployed on a different EJB server and each bean performs operations against external resources (for example, a database) and/or is managing one or more entity beans. In this scenario, the client's logic could required an all-or-nothing guarantee for the operations performed by the session beans; hence, the session bean usage could be bundled together with a JTA UserTransaction object. In the previous scenario, however, the client application developer should address the question of whether or not it would be better to encapsulate these operations in yet another session bean, and allow the session bean to handle the transactions via the EJB container. In general, lightweight clients are easier to maintain than heavyweight clients. Also, EJB environments are ideally suited for transaction management.

```
Context c = new InitialContext(); UserTransaction ut = (UserTransaction)
c.lookup("java:comp/UserTransaction"); ut.begin();  // perform multiple operations...
ut.commit() ...
```

22. **Can the bean class implement the EJBObject class directly? If not why?** - It is better not to do it will make the Bean class a remote object and its methods can be accessed without the containers? security, and transaction implementations if our code by mistake passed it in one of its parameters. Its just a good design practice.
23. **What does isIdentical() method return in case of different type of beans?** - Stateless - true always. Stateful - depends whether the references point to the same session object. Entity - Depends whether the primary key is the same and the home is same.
24. **How should you type cast a remote object? Why?** - A client program that is intended to be interoperable with all compliant EJB Container implementations must use the javax.rmi.PortableRemoteObject.narrow(…) method to perform type-narrowing of the client-side representations of the remote home and remote interfaces. Programs using the cast operator for narrowing the remote and remote home interfaces are likely to fail if the Container implementation uses RMI-IIOP as the underlying communication transport.
25. **What should you do in a passive method?** - You try to make all nontransient variables, which are not one of the following to null. For the given list the container takes care of serializing and restoring the object when activated. Serializable objects, null, UserTransaction, SessionContext, JNDI contexts in the beans context, reference to other beans, references to connection pools.
Things that must be handled explicitly are like a open database connection etc. These must be closed and set to null and retrieved back in the activate method.

What is BEA Weblogic?

BEA WebLogic is a J2EE application server and also an HTTP web server by BEA Systems of San Jose, California, for Unix, Linux, Microsoft Windows,

and other platforms. WebLogic supports Oracle, DB2, Microsoft SQL Server, and other JDBC-compliant databases. WebLogic Server supports WS-Security and is compliant with J2EE 1.3.
BEA WebLogic Server is part of the BEA WebLogic Platform™. The other parts of WebLogic Platform are:
* Portal, which includes Commerce Server and Personalization Server (which is built on a BEA-produced Rete rules engine),
* WebLogic Integration,
* WebLogic Workshop, an IDE for Java, and
* JRockit, a JVM for Intel CPUs.

WebLogic Server includes .NET interoperability and supports the following native integration capabilities:
* Native enterprise-grade JMS messaging
* J2EE Connector Architecture
* WebLogic/Tuxedo Connector
* COM+ Connectivity
* CORBA connectivity
* IBM WebSphere MQ connectivity

BEA WebLogic Server Process Edition also includes Business Process Management and Data Mapping functionality.
WebLogic supports security policies managed by Security Administrators. The BEA WebLogic Server Security Model includes:
* Separate application business logic from security code
* Complete scope of security coverage for all J2EE and non-J2EE components

Which of the following statements are true regarding MDBs (Message Driven Beans) on version 6.0 of WebLogic App Server?
a. MDBs support concurrent processing for both Topics and Queues.
b. MDBs support concurrent processing for only Topics.
c. MDBs support concurrent processing for only Queues.
d. MDBs support concurrent processing neither Topics nor Queues.
Choice A is correct. MDBs support concurrent processing for both Topics and Queues. Previously, only concurrent processing for Queues was supported. To ensure concurrency, change the weblogic-ejb-jar.xml deployment descriptor max-beans-in-free-pool setting to >1. If this element is set to more than one, the container will spawn as many threads as specified. WebLogic Server maintains a free pool of EJBs for every stateless session bean and message driven bean class.
The max-beans-in-free-pool element defines the size of this pool. By default, max-beans-in-free-pool has no limit; the maximum number of beans in the free pool is limited only by the available memory.

Can I use a "native" two-tier driver for a browser applet?
No. Within an unsigned applet, you cannot load native libraries over the wire, access

the local file system, or connect to any host except the host from which you loaded the applet. The applet security manager enforces these restrictions on applets as protection against applets being able to do unsavory things to unsuspecting users.
If you are trying to use jDriver for Oracle from an applet, then you are violating the first restriction. Your applet will fail when it attempts to load the native (non-Java layer) library that allows jDriver for Oracle to make calls into the non-Java Oracle client libraries. If you look at the exception that is generated, you will see that your applet fails in java.lang.System.loadLibrary, because the security manager determined that you were attempting to load a local library and halted the applet.
You can, however, use the WebLogic JTS or Pool driver for JDBC connectivity in applets. When you use one of these WebLogic multitier JDBC drivers, you need one copy of WebLogic jDriver for Oracle (or any other two-tier JDBC driver) for the connection between the WebLogic Server and the DBMS.

I'm using a WebLogic multitier driver in an applet as an interface to a DBMS. If I run the class using the Sun Appletviewer on my local machine, I have no problems. But when I try to run the applet in a Netscape browser, it will not connect.

If Appletviewer works and Netscape does not, it is an indication that you are violating a Netscape security restriction. In this case, the violation is that an applet cannot open a socket to a machine other than the one from which it loaded the applet. To solve this problem, you will have to serve your applet code from the same machine that hosts the DBMS.
In addition, the IP naming format you use in the applet CODEBASE and the constructor for the T3Client must match. That is, you can't use dot-notation in one place and a domain name in the other.

I tried to run two of the applets in the examples directory of the distribution. I installed the WebLogic classes on my local machine (NT server) and on another machine (a Windows 95 client). I am not using any browsers, just trying to run the applets with Appletviewer. The applets work fine when I run Appletviewer from the NT server, but do not work at all from the Windows 95 client.

There are two possible problems: Either the CODEBASE tag is not properly set in the applet HTML file, or the class files are not properly loaded on the HTTP server.

The applet works on the NT server because you installed the WebLogic distribution on your NT server. Even if the applet cannot successfully load the necessary classes from the HTTP server, it does find them in your local CLASSPATH. But when you try to run it from the Windows 95 client, the applet must load the classes over the wire from the HTTP server, and if you haven't installed them correctly, it will fail.

The two primary cluster services provided by WebLogic Server are?

a. Http Session State Clustering
b. File Service Clustering
c. Time Service Clustering
d. Object Clustering
e. Event Clustering

Choices A and D are correct. A WebLogic Server cluster is a group of servers that work together to provide a more scalable and reliable application platform than a single server. A clustered service is an API or interface that is available on multiple servers in the cluster. HTTP session state clustering and object clustering are the two primary cluster services that WebLogic Server provides. WebLogic Server also provides cluster support for JMS destinations and JDBC connections. WebLogic Server provides clustering support for servlets and JSPs by replicating the HTTP session state of clients that access clustered servlets and JSPs. To benefit from HTTP session state clustering, you must ensure that the session state is persistent, either by configure in-memory replication, file system persistence, or JDBC persistence. If an object is clustered, instances of the object are deployed on all WebLogic Servers in the cluster. The client has a choice about which instance of the object to call. This is Object Clustering. The APIs and internal services that cannot be clustered in WebLogic Server version6.0 are File services, Time services, WebLogic Events, Workspaces and ZAC.

How do stubs work in a WebLogic Server cluster?

Clients that connect to a WebLogic Server cluster and look up a clustered object obtain a replica-aware stub for the object. This stub contains the list of available server instances that host implementations of the object. The stub also contains the load balancing logic for distributing the load among its host servers.

What happens when a failure occurs and the stub cannot connect to a WebLogic Server instance?

When the failure occurs, the stub removes the failed server instance from its list. If there are no servers left in its list, the stub uses DNS again to find a running server and obtain a current list of running instances. Also, the stub periodically refreshes its list of available server instances in the cluster; this allows the stub to take advantage of new servers as they are added to the cluster.

Why did my JDBC code throw a rollback SQLException?

Your JDBC code may throw the following exception:
"The coordinator has rolled back the transaction.
No further JDBC access is allowed within this transaction."

The WebLogic JTS JDBC driver throws this exception when the current JDBC connection transaction rolls back prior to or during the JDBC call. This exception indicates that the transaction in which the JDBC connection was participating was rolled back at some point prior to or during the JDBC call.
The rollback may have happened in an earlier EJB invoke that was part of the transaction, or the rollback may have occurred because the transaction timed out. In either case, the transaction will be rolled back, the connection returned to the pool and the database resources released. In order to proceed, the JTS JDBC connection must be closed and reopened in a new transaction.

Must my bean-managed persistence mechanism use the WebLogic JTS driver?

Use the TxDataSource for bean-managed persistence.

Why is there no polymorphic-type response from a create() or find() method?

The EJB Specification prohibits this behavior, and the weblogic.ejbc compiler checks for this behavior and prohibits any polymorphic type of response from a create() or find() method.

The reason the create() and find() methods are not polymorphic is similar to the reason constructors are not polymorphic in Java. The derived classes generally do not know or cannot initialize the base class properly.

Must EJBs be homogeneously deployed across a cluster? Why?

Yes. Beginning with WebLogic Server version 6.0, EJBs must be homogeneously deployed across a cluster for the following reasons:
* To keep clustering EJBs simple
* To avoid cross server calls which results in more efficiency. If EJBs are not deployed on all servers, cross server calls are much more likely.
* To ensure that every EJB is available locally
* To ensure that all classes are loaded in an undeployable way
* Every server must have access to each EJB's classes so that it can be bound into the local JNDI tree. If only a subset of the servers deploys the bean, the other servers will have to load the bean's classes in their respective system classpaths which makes it impossible to undeploy the beans.

Which of the following are recommended practices to be performed in the ejbPassivate() method of a stateful session bean?

a. Close any open resources, like database connections
b. All non-transient, non-serializable fields(except some special types) should be set to null.
c. All transient fields should be set to null
d. Make all database connection reference fields transient
e. All primitive type fields should be set to null

Choices A, B and D are correct. When a bean is about to be passivated, its ejbPassivate() method is invoked, alerting the bean instance that it is about to enter the Passivated state. At this time, the bean instance should close any open resources and set all non transient, non serializable fields to null. This will prevent problems from occurring when the bean is serialized. Transient fields will simply be ignored.Serializable fields will be saved. Open resources such as sockets or JDBC connections must be closed whenever the bean is passivated. In stateful session beans, open resources will not be maintained for the life of the bean instance. When a stateful session bean is passivated, any open resource can cause problems with the activation mechanism.
A bean's conversational state may consist of only primitive values, objects that are serializable, and the following special types-SessionContext, EJBhome, EJBObject, UserTransaction and Context (only when it references the JNDI ENC) . The types in this list (and their subtypes) are handled specially by the passivation mechanism. They don't need to be serializable; they will be maintained through passivation and restored automatically to the bean instance when it is activated

While packaging the Web Application DefaultWebApp for deployment into the WebLogic server, the home and remote interfaces of the enterprise beans used by the servlets should reside in which directory?

a. DefaultWebApp/META_INF/classes
b. DefaultWebApp/META_INF/lib
c. DefaultWebApp/WEB_INF/lib
d. DefaultWebApp/WEB_INF/classes
e. DefaultWebApp/classes
Choice D is correct. When packaging a web application create META-INF and WEB-INF subdirectories in the application directory to hold deployment descriptors and compiled Java classes. All servlet classes and helper classes should reside in the WEB-INF/classes subdirectory. The home and remote interface classes for enterprise beans used by the servlets into the WEB-INF/classes subdirectory.
All the HTML files, JSP files, images, and any other files that these Web pages reference should exist in the application directory, maintaining the directory structure for referenced files. The META_INF directory contains the deployment descriptors for the enterprise beans, but not the classes.

How do I set up my CLASSPATH?
Setting up your CLASSPATH correctly depends on what you are trying to do. The most common tasks are described below:
* Starting WebLogic Server. See Setting the Classpath Option in the Starting and Stopping WebLogic Servers section of the Administration Guide. In addition, your WebLogic distribution includes shell scripts that you can use to start the server. These scripts, which are located in the domain directories under the config directory of your WebLogic Server distribution, automatically set up the CLASSPATH variable in the shell before starting the server.

Why do I get the following exception when viewing the JNDI tree?
isSerializable(class.javax.naming.Binding)
java.io.NotSerializableException:
java.io.PrintWriter at
java.io.ObjectOutputStream.OutputObject

The Weblogic Server JNDI implementation requires objects to be serializable, not referencable. A PrintWriter cannot be serialized and therefore should be declared transient.

When deploying a resource adapter (.rar) to WebLogic Server, are its classes placed in the WebLogic classpath?
For instance, I am deploying an EJB and a resource adapter (.rar), the EJB has no dependencies on the .rar because the EJB is writing to the common client interface (CCI). The EJB client application has sends/marshals as parameter classes that are defined in the .rar. For some reason the EJB's class loader hierarchy cannot find the definition of this .rar-specific class, even though the .rar is deploying successfully. I receive the following error on the EJB client:
java.rmi.UnmarshalException: error unmarshalling arguments; nested
exception
is:
java.lang.ClassNotFoundException:
com.mycompany.InteractionSpecImpl

When you pass an instance of com.myclientcompany.server.eai.InteractionSpecImpl as an argument to your EJB, the appServer needs to de-serialize (unmarshal) the object under the EJB context, and it needs the required class for unmarshalling, inside the ejb-jar(raTester.jar). So if you include the interactionspecimpl class in your ejb-jar file, then you do not need to include those classes in your server's classpath.

How is security handled in the WebLogic J2EE Connector Architecture?

Due to the fact that the current configuration and packaging requirements for resource adapters in WebLogic Server require the hand-editing of the weblogic-ra.xml file, any new passwords specified in the security-principal-map entries are done in clear-text.

BEA understands the importance of protecting security passwords. Hence, we provide a Converter Tool that allows for the encryption of all passwords present in the weblogic-ra.xml file. The Converter Tool is shipped in the standard weblogic.jar file.

Can I enable requests to a JDBC connection pool for a database connection to wait until a connection is available?

No, there's no way to allow a request to wait for a pool connection, and from the system point of view there should not be. Each requests that waits for a connection ties up one of the fixed number of execute threads in the server, which could otherwise be running another server task. Too many waiting requests could tie up all of the execute threads and freeze the server.

How do I use multibyte character sets with WebLogic jDriver for Informix?

Currently, multibyte character sets are not supported for the WebLogic jDriver for Informix driver.

How do I connect to an SQL Server instance that is running on a machine with multiple instances of SQL Server 2000?

Each instance of MS SQL Server must be listening on a different port. So, you can use the port number in the properties that you pass to the getConnection() method or, in case of connection pools, you can specify the port property in the following properties:

```
server=machineName
port=instancePort
```

To find the port number where each MS SQL Server instance is running, run the server network utility (in the Microsoft SQL Server program group), select the server instance, select TCP/IP, and click the properties button.

Why does FOR UPDATE in Oracle 8 cause an ORA-01002 error?

The Oracle 8 server generates an ORA-01002:fetch out of sequence error message when you use a FOR UPDATE statement with AUTOCOMMIT turned on (which is the default state when using JDBC). This is known to happen on Oracle 8.0 and 8.1 on Solaris and on Oracle 8.1 on Windows NT. If you turn AUTOCOMMIT off, you will not receive this error. Because this problem is due to a change in the Oracle 8 server, you should contact Oracle support for more information.

What causes an OCIW32.dll error?

You may receive the following error message when using your JDBC driver for Oracle: "The ordinal 40 could not be loaded in the dynamic link library OCIW32.dll." This problem is caused by an out-of-date version of OCIW32.DLL in your system directory. Some programs install this file in the system directory in order to run. If you remove this file from the system directory you should no longer receive this error.

What transaction isolation levels does the WebLogic jDriver for Oracle support?

Your servlet application may use Oracle Thin Drivers to access a database that includes BLOB fields. If you install and try to use WebLogic jDriver for Oracle and the same code fails and produces an exception similar to the following:

```
com.roguewave.jdbtools.v2_0.LoginFailureException:
TRANSACTION_READ_UNCOMMITTED isolation level not allowed
The Stack Trace:
com.roguewave.jdbtools.v2_0.LoginFailureException:
TRANSACTION_READ_UNCOMMITTED isolation level not allowed
at
com.roguewave.jdbtools.v2_0.jdbc.JDBCServer.createConnection
(JDBCServer.java :46)
at com.roguewave.jdbtools.v2_0.ConnectionPool.getConnection_
(ConnectionPool.jav a:412)
at com.roguewave.jdbtools.v2_0.ConnectionPool.getConnection
(ConnectionPool.java :109)
```

Setting the Isolation_level to 1 in the code that calls the RogueWave JDBCServer class works with the Oracle thin driver but fails with WebLogic jDriver for Oracle.

WebLogic jDriver for Oracle supports the following transaction isolation levels:

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
```

According to the Oracle documentation, the Oracle DBMS only supports these two isolation levels. Unlike other JDBC drivers, WebLogic's drivers throw an exception if you try to use an isolation level that is unsupported. Some drivers silently ignore attempts to set an unsupported isolation level. WebLogic suggests testing whether the Oracle thin driver is not just ignoring settings for unsupported isolation events.

How do I use OS Authentication with WebLogic jDriver for Oracle and Connection Pools?
Using OS authentication in connection pools essentially means that you are using the UserId of the user who started WebLogic Server. OS authentication is available on NT and UNIX, but not on Solaris. This means that database security will rely strictly on the security of WebLogic; that is, if you are allowed to make a client connection to the WebLogic Server and access the pool, then you can get to the database.

You can do this with WebLogic jDriver for Oracle because Oracle uses the process owner to determine who is attempting the connection. In the case of WebLogic JDBC, this is always the user that started the WebLogic Server.

To set up your Oracle instance to use this feature, your DBA needs to follow these basic steps. The full procedure is described in more detail in your Oracle documentation.

1. Add the following line to the INIT[sid].ORA file:
OS_AUTHENT_PREFIX = OPS$
Note that the string "OPS$" is arbitrary and up to the DBA.
2. Log in to the Oracle server as SYSTEM.
3. Create a user named OPS$userid, where userid is some operating system login ID. This user should be granted the standard privileges (for example, CONNECT and RESOURCE).
4. Once the userid is set up, you can connect with WebLogic jDriver for Oracle by specifying "/" as the username property and "" as the password property. Here is an example for testing this connection with the dbping utility:

```
$ java utils.dbping ORACLE "/" "" myserver
```

Here is a code example for WebLogic jDriver for Oracle:

```
Properties props = new Properties();
props.put("user", "/");
props.put("password", "");
props.put("server", "myserver");

Class.forName("weblogic.jdbc.oci.Driver").newInstance();
Connection conn = myDriver.connect("jdbc:weblogic:oracle",
props);
```

1. Use the Administration Console to set the attribute for your connection pool. The following code is an example of a JDBC connection pool configuration using the WebLogic jDriver for Oracle:

```
<JDBCConnectionPool
Name="myPool"
Targets="myserver,server1"
DriverName="weblogic.jdbc.oci.Driver"
InitialCapacity="1"

MaxCapacity="10"
CapacityIncrement="2"
Properties="databaseName=myOracleDB"
```

What type of object is returned by ResultSet.getObject()?
WebLogic jDriver for Oracle always returns a Java object that preserves the precision of the data retrieved. WebLogic jDriver for Oracle returns the following from the getObject() method:

* For columns of types NUMBER(n) and NUMBER(m,n): a Double is returned if the defined precision of the column can be represented by a Double; otherwise BigDecimal is returned.

* For columns of type NUMBER: Because there is no explicit precision, the Java type to return is determined based on the actual value in each row, and this may vary from row to row. An Integer is returned if the value has a zero-valued fractional component and the value can be represented by an integer.

For example, 1.0000 will be an integer. A long is returned for a value such as 123456789123.00000. If a value has a non-zero fractional component, a Double is returned if the precision of the value can be represented by a Double; otherwise a BigDecimal is returned.

How do I limit the number of Oracle database connections generated by WebLogic Server?

You can use connection pools to limit the number of Oracle database connections generated by WebLogic Server in response to client requests. Connection pools allow T3 applications to share a fixed number of database connections. For information on how to set up connection pools,

How do I call Oracle stored procedures that take no parameters?

Here is what we use that works:

```
CallableStatement cstmt = conn.prepareCall("Begin procName;
END;");
cstmt.execute();
```

where procName is the name of an Oracle stored procedure. This is standard Oracle SQL syntax that works with any Oracle DBMS. You might also use the following syntax:

```
CallableStatement cstmt = conn.prepareCall("{call procName};");
cstmt.execute();
```

This code, which conforms to the Java Extended SQL spec, will work with any DBMS, not just Oracle.

Why do I get unexpected characters from 8-bit character sets in WebLogic jDriver for Oracle?

If you are using an Oracle database with an 8-bit character set on Solaris, make sure you set NLS_LANG to the proper value on the client. If NLS_LANG is unset, it defaults to a 7-bit ASCII character set, and tries to map characters greater than ASCII 128 to a reasonable approximation (for example, á, à, â would all map to a). Other characters are mapped to a question mark (?).

How do I learn what codesets are available in Oracle?

To find out what codesets you currently have available in Oracle, execute the following SQL query from SQLPlus at the command line:

```
SQL> SELECT value FROM v$nls_valid_values WHERE parameter='CHARACTERSET';
```

The response lists of all codesets currently installed on your system. This listing will look something like the following shortened list:

```
VALUE
---------------
US7ASCII
WE8DEC
WE8HP
US8PC437
WE8EBCDIC37
WE8EBCDIC500
WE8EBCDIC285
...
```

If you want to constrain the value in the query to a specific codeset you are searching for, you might use a SQL query like the following:

```
SQL> SELECT value FROM v$nls_valid_values
WHERE parameter='CHARACTERSET' and VALUE='AL24UTFFSS';
```

This would produce the following response if the codeset is installed:

```
VALUE
-------------------
AL24UTFFSS
```

You can use Oracle's installation tools to install additional codesets. Contact Oracle for more information.

How many deployment descriptor files does a CMP entity bean deployed on the WebLogic Server have?

a. One J2EE specific deployment descriptor and two WebLogic specific deployment descriptors
b. One J2EE specific deployment descriptor and one WebLogic specific deployment descriptors
c. One J2EE specific deployment descriptor only
d. One WebLogic specific deployment descriptor only

Choice A is correct. Deployment descriptors are text documents formatted with XML tags. The J2EE specifications define standard, portable deployment descriptors for J2EE components and applications. BEA defines additional WebLogic-specific deployment descriptors required to deploy a component or application in the WebLogic Server environment.
When packaging an enterprise bean, we need to create an ejb-jar.xml deployment descriptor in the META-INF subdirectory and add entries for the bean. We also need to create a weblogic-ejb-jar.xml deployment descriptor in the META-INF subdirectory and add entries for the bean. If the bean is an entity bean with container-managed persistence, first we create a weblogic-rdbms-cmp-jar-bean_name.xml deployment descriptor in the META-INF directory with entries for the bean. Then we map the bean to this CMP deployment descriptor with a attribute in the weblogic-ejb-jar.xml file.

Why do I get an error while trying to retrieve the text for ORA-12705?
This error occurs when you have not set the ORACLE_home environment variable properly — a common mistake. In order to use WebLogic jDriver for Oracle, the Oracle client software needs to be installed and ORACLE_home must be set.
You may also see this error message if you try to use WebLogic jDriver for Oracle's internationalization capabilities with a language/codeset combination that is not installed on your system. If you get the ORA-12705 error with the correct error text, then either you have set NLS_LANG improperly, or you do not have the right codesets installed on your system.

Why do I run out of resources during updates with Oracle's database link?
When you use Oracle's database link to update your database, you may get error "maximum number of temporary table locks exceeded" even if you close your result sets and statements when you finish. The database link is an object in the local database that allows you to access tables, views, and such in a remote database. The database link is controlled by the Oracle server, so the driver has no control over its use of resources. The link appears to perform the commit (since other processes could see the records that were being created), but it doesn't free any resources until the connection is closed. The solution is to remove the database link and use the JDBC driver to do your selects, inserts, and updates.

How do I prevent errors when running t3dbping?
When you are testing your Oracle database connections under UNIX, you can run SQL*PLUS and can successfully ping the database using utils.dbping. However, when you use the multitier utils.t3dbping utility, you receive an ORA-12154 error message.

First, make sure that your ORACLE_home environment variable is correctly set to point to your Oracle installation. This variable must be set in the environment where the WebLogic server is running.

In the C-shell issue the following command:
```
$ setenv ORACLE_home path
```

where path is the path to your Oracle installation.
In the Bourne shell, issue the following commands:
```
$ ORACLE_home=path
$ export ORACLE_home
```

Where path is the path to your Oracle installation. When you ping your database using the two-tier utils.dbping utility, the JDBC driver loads the database client library and establishes the connection to the database. When you use the multitier utils.t3dbping utility, the WebLogic Server loads a two-tier driver and uses it to establish a database connection. In both cases, the same method is used to connect to the database. SQL*PLUS works because it doesn't require ORACLE_home to find the client libraries.

If you are still experiencing problems, try this:
1. Open a command shell.
2. Run the two-tier version of utils.dbping in this shell.
3. Start WebLogic in this shell from the command line:
$ java -ms32m -mx32m weblogic.server
4. Open a second command shell.
5. Run the utils.t3dbping in the second shell against the server running in the first command shell.
If this procedure doesn't work, please send the output from these commands to WebLogic technical support.

Why does executing the PreparedStatement class cause a "TRUNC fails: ORA-00932: inconsistent datatypes" error?
According to Oracle Metalink Bug Database Doc ID: 144784.1, in the absence of explicit data typecasting, OCI assumes that a bind variable is a CHAR data type. If the SQL statement intends to use the bind variable as a DATE data type, but OCI thought it was a CHAR, the SQL parser will have a conflict in data types. The fix is to explicitly use data conversion functions to convert the bind variables in the problem queries. For example, a select string of
String st = "select count(*) from simple_table where
TRUNC(mydate) = TRUNC(?)";

should be changed to:
String st = "select count(*) from simple_table where
TRUNC(mydate) = TRUNC(TO_DATE(?))";

Why am I getting an "ORA-01000: maximum open cursors exceeded" error, even though I closed all ResultSet, Statement, and Connection objects?
This is an Oracle issue. According to Oracle's documentation, dynamic cursors can remain open from run to run in a session and are not closeable when a procedure closes. To work around this issue, you can increase the number of open cursors allowed in the database or you can reset the connection pool (close and reopen database connections in the connection pool).
To reset the connection pool, you can untarget and retarget the connection pool using the Administration Console. You can also use the reset() method through the JMX API or the RESET_POOL command on the WebLogic Server command line interface.

An instance of stateful session EJB when accessed simultaneously from more than one clients on same VM results in RemoteException or EJBException. In case the client is a Servlet thread, which of the following techniques can be used to avoid RemoteException/EJBException?

a. Not possible.
b. Store the reference to the EJB instance as an instance variable of Servlet class.
c. Store the reference to the EJB instance as a local variable of Servlet class.
d. Make the Servlet client to be remote instead of internal to WebLogic server.

Choice C is the correct choice. An instance of a stateful session EJB can be accessed from only one client virtual machine at a time. Multiple client threads from the same virtual machine can access the same instance of a stateful session EJB, but they must do so in a serial fashion. If a client-invoked business method is in progress on an instance when another client-invoked call, from the same or different client, arrives at the same instance of a stateful session bean class, the container may throw

the java.rmi.RemoteException to the second client , if the client is a remote client, or the javax.ejb.EJBException, if the client is a local client. Thus choice D is incorrect.
To avoid any exception, each Servlet should store a reference to a particular EJB instance in a local variable of the Servlet's service() method. Please note that variables local to methods like service(), doGet(), doPost() are not shared between different requests and are automatically thread safe. Thus choice C is correct. An instance variable unlike local variable is shared. Thus Choice B is incorrect.
An implication of this rule is that an application cannot make loop back calls to a session bean instance.

This restriction does not apply to a stateless session bean because the container routes each request to a different instance of the session bean class.

Are there C/C++ interfaces to WLS JMS?

No, this is not supported.
* Write your own interfaces using JNI.
* Setup a Servlet that your C/C++ client calls to generate a JMS message. You should spawn multiple threads in C++ and use multiple posts to pass messages via http.

How do I start WLS and configure JMS?

On Windows, start WLS 6.X by selecting Start -< Programs -< BEA WebLogic E-Business Platform -< WebLogic Server 6.X -< Start Default Server and enter the administrator password.

On Windows, to configure JMS, start the console by selecting Start -< Programs -< BEA WebLogic E-Business Platform -< WebLogic Server 6.X -< Start Default Console.

1. In the console tree view on the left, select JMS.

2. If you want persistent messages, first create a Store - Select Stores. In the window on the right, Select Create a new JMSFile Store for a file store, give it a name, give it a directory, select create. If you want a JDBCStore, you first need to create a JDBC connection pool by selecting JDBC in the tree view, Connection Pools, create a new JDBC Connection Pool. Select Targets, select a Target server, select the arrow that points to the right and select Apply. Then go back to Stores, Create a new JMSJDBCStore.

3. If you want to use a template, first create a Template - Select Templates. You need a template to create temporary queues. Select Create a new JMS Template, give it a name, select create, then you can move to the Thresholds &Quotas tab or the Override tab. Select Apply when done with your changes.

4. Select Servers. Select Create a new JMSServer, give it a name, select a Store if you created one, select a template if you created one, Select Create. Now you can move to the other tabs, make changes, select Apply. In particular, you must select Targets, select a Target server, select the arrow that points to the right, and select Apply. This is the server on which JMS will boot.

5. Create Destinations - from the tree view in the left panel, select the + in front of JMS, select the + in front of Servers, select the + in front of your server, select Destinations, Select Create a new JMSQueue or Create a new JMSTopic, fill in the first page and Select Create, then you can select, fill in, and Apply other tabs.

6. Create Connection Factories - on left tree view, open JMS. Select Connection Factory. Select Create a new JMS Connection Factory on the right panel. Type in the name and JNDI name. Select Create (lower right hand corner). Select the Targets tab. Select the name of the server on which you want to deploy the connection factory. Select the arrow pointing to the right - the server moves to chosen. Then select Apply (lower right hand corner).

How do I configure JMS security?

The correct way to set up security for JMS is to go to the console, select ACLs in the tree view, then create some access control lists.
1. Set the ACL name which should be weblogic.jms.queue.QUEUENAME or weblogic.jms.topic.TOPICNAME.
2. Select Create.
3. Enter the New Permission of send or receive.

4. Select Create.
5. Enter a comma separated list of users or groups.
6. Select Grant Permission.
7. Select "saved to the realm implementation" to save your changes.
8. Select Yes.

This will update the fileRealm.properties file with lines that look like the following:
acl.send.weblogic.jms.queue.TestQueue1=user1
acl.receive.weblogic.jms.queue.TestQueue1=user1
If you don't have an ACL for a queue or topic, security is wide open.
There are also ACL's for accessing the JNDI context; the JNDI context is a requirement for initially accessing JMS. See the JNDI documentation.

Can I still use the default connection factories supported in WebLogic Release 5.1?

Yes. The following two names for the default connection factories have been deprecated:

javax.jms.QueueConnectionFactory
javax.jms.TopicConnectionFactory.

However, these connection factories are still defined and usable in this release for backwards compatibility.
WebLogic JMS 6.1 defines one connection factory, by default:
weblogic.jms.ConnectionFactory
You have to Enable the JMS default connection factories. Go to the console->your server->tuning->click on the check box Enable Default JMS Connection Factories.
You can also specify user-defined connection factories using the Administration Console.

Why does JMSSession.createTopic or JMSSession.createQueue fail to create a destination in WLS JMS 6.1 (it worked in 5.1)?

In WLS 5.1 createTopic() or createQueue() creates the destination permanently in the database if it doesn't already exist, but does not modify the weblogic.properties file.
According to the JavaSoft JMS specification version 1.0.2 regarding createQueue() and createTopic(), they are not for creating destinations dynamically. They are used to retrieve the destination referenced by using a string name instead of using JNDI lookup. The destination has to be in your config.xml file first. This change is documented in WLS 6.0 since it behaves differently than the previous release. You can use the WLS JMS helper class (weblogic.jms.extensions.JMSHelper) or the console to create destinations at the run time (note that there was a bug in 6.0 that caused a problem when the server restarted; this is fixed in Service Pack 1). These mechanisms create the destination and also modify the configuration file.
For more information on the JMSHelper classes, see the subsection called Creating Destinations Dynamically in Programming WebLogic JMS.
The following program creates a Topic.

```
import java.io.*;
import java.util.Hashtable;
import javax.jms.*;
import javax.naming.*;
import weblogic.jms.extensions.JMSHelper;


class t {
public final static String
JNDI_FACTORY="weblogic.jndi.WLInitialContextFactory";
public final static String JMS_SERVER_NAME="TestJMSServer";
public final static String DEST_JNDI_PREFIX="javax.destination.";


static public void main(String [] args) throws Exception {
try {
Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY, JNDI_FACTORY);
env.put(Context.PROVIDER_URL, "t3://localhost:7001");
Context ctx = new InitialContext(env);


String topicName = "JMSHelperTestQueue01";
String topicJNDI = DEST_JNDI_PREFIX + topicName;
System.out.println("topic name=" + topicName + ", jndi=" +
topicJNDI);
JMSHelper.createPermanentTopicAsync(ctx, JMS_SERVER_NAME,
topicName,
topicJNDI);
} catch (JMSException e) {
e.printStackTrace();
}
}
}
```

How do I programmatically get a list of Queues or Topics?
The following program uses Mbeans:

```
import weblogic.management.*;
import weblogic.management.configuration.*;


InitialContext ic = new InitialContext();
MBeanhome home = (MBeanhome)ic.lookup(MBeanhome.ADMIN_JNDI_NAME);
for(Iterator i = o.getMBeansByType("JMSTopic").iterator();
i.hasNext(); ){
WebLogicMBean wmb = (WebLogicMBean)i.next();
System.out.println("topic name found: " + wmb.getName());
}


for(Iterator i = o.getMBeansByType("JMSQueue").iterator();
i.hasNext(); ){
WebLogicMBean wmb = (WebLogicMBean)i.next();
System.out.println("queue name found: " + wmb.getName());
}
```

How do I use a temporary destination?

You must create a template on every JMSServer where you want to be able to create temporary destinations. You can specify multiple JMSServer entries to support TemporaryTemplate and the system will load balance among those JMS servers to setup the temporary destination. See How do I start WLS and configure JMS? for a description about how to configure JMS. The resulting template definition looks something like the following:

```
<JMSTemplate Name="MyTemplate"/>
```

The JMSServer is defined something like:

After the template name, you can set any queue/topic attribute you want in the template (not including a JNDI name or topic multicast settings). The template is at the outer most level; that is, it should not be nested in your .

Temporary destinations can only be consumed by the creating connection. Using topics, you create your temporary topic and subscribe to that temporary topic. If you want someone to publish to that temporary topic, you need to tell that someone what your topic is. You can send them a message and include your temporary topic in the JMSReplyTo field. The creator of the TemporaryTopic and the subscriber must be one in the same.

```
import javax.jms.TopicSession;
TemporaryTopic myTopic = mySession.createTemporaryTopic();
TopicSubscriber = mySession.createSubscriber(myTopic);
```

Temporary topics do not get names and cannot be subscribed to by other connections. When you create a temporary topic, the JMS provider returns a javax.jms.Topic. You then need to advertise that topic to other parties (those who want to publish to the topic), putting it in your JMSReplyTo field so that they can respond. In general, no one else can subscribe to the topic. You advertise the topic any way you want. Topics are serializable (or, in our case, externalizable), which allows you to pass them around in RMI calls, through a file, binding it to a name in JNDI, etc. In short, create the topic at the subscriber side and advertise so that others can publish. You can get multiple subscribers on the same connection and get concurrent processing using multiple sessions.

Can two JMS servers share the same persistent store?
No. Each JMS server must have its own unique persistent store. Two file-based JMS persistent stores may share the same directory, but their messages will be stored in different files. In this case, the filenames will contain different prefixes.
Two JDBC-based JMS persistent stores may share the same database, but they must be configured to use a different Prefix Name which will be prepended to the database tables. For more information on configuring the JDBC Prefix Name, see "JMS JDBC Stores" in the Administration Console Online Help. If they are configured with the same Prefix Name, persistent messages will be corrupted and/or lost.

Which types of JDBC databases does WebLogic JMS support?
The JMS database can be any database that is accessible through a JDBC driver. WebLogic supports and provides JDBC drivers for the following databases:
* Cloudscape
* Informix
* Microsoft SQL (MSSQL) Server (Versions 6.5 and 7)
* Oracle (Version 8.1.6)
* Sybase (Version 12)

How do I use a third-party JDBC driver with JMS?
If your JDBC driver is not included in the list of drivers in the question about JDBC databases supported by WebLogic JMS, then the tables required by JMS must be created manually.

Note: WebLogic Server only guarantees support for the JDBC drivers included in the previous list. Support for any other JDBC driver is not guaranteed.

The .ddl files located in the weblogic/jms/ddl directory of the weblogic.jar file may be used as templates. Use the jar utility supplied with the JDK to extract them to the weblogic/jms/ddl directory using the following command:

```
jar xf weblogic.jar weblogic/jms/ddl
```

Note: If you omit the second parameter (weblogic/jms/ddl), the entire jar file is extracted.
Follow the procedures in JDBC Database Utility in Programming WebLogic JMS to manually create the database tables for the JDBC store.
Another option is to consider using a file store instead of a JDBC store. File stores are easier to configure and may provide significantly better performance.

The Multicast TTL setting for a cluster in the WebLogic Admin console sets which of the following values?

a. Maximum time taken for multicast messages to reach their final destination

b. The number of routers a multicast message can pass through before the packet can be discarded
c. The multicast address to be used by the messages sent from the cluster
d. Minimum time taken for broadcasting a multicast message from the cluster

Choice B is correct. The Multicast TTL(TTL-Time to Live) setting specifies the number of routers a multicast message can pass through before the packet can be discarded. To configure the multicast TTL for a cluster, you should change the Multicast TTL value in the WebLogic Server administration console. This sets the number of network hops a multicast message makes before the packet can be discarded.
If you choose to distribute a cluster over a WAN (or across multiple subnets), you must plan and configure your network topology to ensure that multicast messages are reliably transmitted to all servers in the cluster. One of the requirements to be met by the network is that the multicast Time To Live (TTL) value must be high enough to ensure that routers do not discard multicast packets before they reach their final destination.

Which of the following algorithms is used by the WebLogic Server as the default load balancing strategy for clustered object stubs when no algorithm is specified ?
a. Round-robin
b. Weight-based
c. Random
d. None of the above

8. Choice A is correct. The basic idea behind load balancing is that by distributing the load proportionally among all the servers in the cluster, the servers can each run at full capacity. WebLogic Server clusters support several algorithms for load balancing clustered objects. The particular algorithm you choose is maintained within the replica-aware stub obtained for the clustered object. Configurable algorithms for load balancing clustered objects are: Round-robin, Weight-based and Random. WebLogic Server uses the round-robin algorithm as the default load balancing strategy for clustered object stubs when no algorithm is specified. Round-robin is the only load balancing strategy used by WebLogic proxy plug-ins for HTTP session state clustering. The round-robin algorithm cycles through a list of WebLogic Server instances in order. For clustered objects, the server list consists of WebLogic Server instances that host the clustered object. For proxy plug-ins, the list consists of all WebLogic Servers that host the clustered servlet or JSP.

How do I use persistence?
Use the following guidelines:
1. Make sure the JMSServer you are using has a store configured. The JMSServer configuration entry in the config.xml file should contain a line of the form

Store=""

Note that if JMS boots without a store configured, it is assumed the customer did not want one, and persistent messages are silently downgraded to non-persistent (as specified for JMS 1.0.2).

2. Make sure you are not using "Message.setJMSDeliveryMode". This is overwritten, as it is a vendor-only method.

3. Make sure you are calling either:

QueueSender.send(msg, deliveryMode, ...)

-- or --

QueueSender.setDeliveryMode(deliveryMode)

-- or --

set the DefaultDeliveryMode mode on connection factory in the config.xml file to persistent (the QueueSender.setDeliver/send overrides this value). Similarly, for topics, you would set this via the TopicPublisher.

4. Make sure you don't have "DeliveryModeOverride" set to Non-Persistent on the Destination in the config.xml file.

5. If you are using pub/sub, only durable subscriptions persist messages. Non-durable subscriptions have no need to persist messages, as by definition they only exist for the life of the server.

6. If you are using JDBC, the JDBC tables, JMSSTATE and JMSSTORE, are created automatically when the JMS server boots. The DDL files used to create the tables are stored in weblogic.jar in weblogic/jms/ddl. The example configuration below shows a JDBC store for Oracle (client version 8.1.7 or later is needed to run with WLS 6.1 on JDK 1.3). To manually create the tables (also deleting any existing tables), run java utils.Schema as described in the previous question.

See the question, "How do I start WLS and configure JMS?" for a description of how to configure JMS.

Here is a sample config.xml file resulting from configuring JMS. It should look similar to yours. If you want JMS to use a file store instead of a database, just change JDBCStore to FileStore in the JMSServer section.

```
ListenPort="7001" DefaultProtocol="t3"
ThreadPoolSize="8" >

GuestDisabled="false" />
FileRealm="defaultFileRealm" />
<FileRealm Name="defaultFileRealm"
/>
TemporaryTemplate="TestTemplate1"
Targets="myserver" Store="JDBCStore">
JNDIName="jms.queue.TestQueue1"
Template="TestTemplate1"
/>

<JMSTemplate Name="TestTemplate1"
/>
<JMSFileStore Name="FileStore"
Directory="myfilestore"
JMSServer="TestJMSServer"
/>
ConnectionPool="testpool2"
JMSServer="TestJMSServer"
/>
<JDBCConnectionPool Name="testpool2"
Targets="myserver"
URL="jdbc:weblogic:oracle"
DriverName="weblogic.jdbc.oci.Driver"
InitialCapacity="0"
MaxCapacity="1"
CapacityIncrement="1"
Properties="user=SCOTT;password=tiger;server=bay816"
/>
```

The following is a sample class that sends a Topic message on construction:

```
import javax.naming.*;
import javax.jms.*;
import java.util.Hashtable;

public class t

{
public final static String DESTINATION="jms.topic.TestTopic1";

private TopicConnectionFactory connectionFactory;
private TopicConnection connection;
private TopicSession session;
private TopicPublisher producer;
private TextMessage message;
private Topic destination;

public t()
{
try {
Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY,
"weblogic.jndi.WLInitialContextFactory");
env.put(Context.PROVIDER_URL, "t3://localhost:7001");
InitialContext ctx = new InitialContext(env);
destination = (Topic) ctx.lookup(DESTINATION);
connectionFactory = (TopicConnectionFactory)
ctx.lookup("javax.jms.TopicConnectionFactory");
connection = (TopicConnection)
connectionFactory.createTopicConnection();
session = (TopicSession) connection.createTopicSession(false,

Session.AUTO_ACKNOWLEDGE);
producer = (TopicPublisher) session.createPublisher(destination);
producer.setDeliveryMode(DeliveryMode.PERSISTENT);
message = (TextMessage) session.createTextMessage();
message.setText("hello world");
producer.publish(message);
} catch (Exception e) {
}
}
}
```

How does a file store compare with a JDBC store?

The following are some similarities and differences between file stores and JDBC stores:
* Both have the same transaction semantics, including rolling back transactions (e.g., received messages are put back on the queue).
* Both have the same application interface (no difference in application code).
* The file store should be much faster.
* JDBC may make it easier to handle failure recovery since the JDBC interface can access the database from any client machine; with the file store, the disk must be shared or migrated.
* File store reliability is limited to reliability of disk and O/S; run it on Veritas or a RAID 5 system. Database reliability may be higher.
* File stores will require more memory, but perhaps not significantly more; it depends on how fragmented the file store gets, if the application works roughly

* FIFO it shouldn't get very fragmented at all.
* File stores generate no additional network traffic, database stores do if the database server is on a different JVM or machine.

How do the WLS JMS 6.1 server/destination message maximum and threshold values work?

The byte and message maximum values are quotas - not flow control. Message quotas prevent a WebLogic JMS server from filling up with messages and possibly running out of memory, causing unexpected results. When you reach your quota, JMS prevents further sends with a ResourceAllocationException (rather than blocking). You can set quotas on individual destinations or on a server as a whole.
The thresholds are also not flow control - though they would be better suited to that application than the quotas. The thresholds are simply settings that when exceeded cause a message to be logged to the console to let you know that you are falling behind.
Note that the messages maximum setting on a connection factory is not a quota. This specifies the maximum numbers of outstanding messages that can exist after they have been pushed from the server but before an asynchronous consumer has seen them; it defaults to a value of 10.

How do I configure JDBC so that the JMS JDBC Store recovers automatically?

Several customers have reported a problem where they are using a JDBC store, the DBMS goes down and back up, but JMS can no longer use the store until WLS is shutdown and restarted. You can get around this problem by configuring the following attributes on the JDBC Connection Pool associated with the JMSJDBCStore:

```
TestConnectionsOnReserve="true"\
TestTableName="[[[catalog.]schema.]prefix]JMSState"
```

If they are not set, then if the JDBC resource goes down and comes back up, JMS cannot re-use the connection pool until WLS is shutdown and restarted. This has been tested against WLS 6.0 SP02 and WLS 6.1.

Does WebLogic JMS support clustering?

WebLogic JMS supports cluster-wide, transparent access to destinations from any server in the cluster. A system administrator can establish cluster-wide, transparent access to destinations from any server in the cluster by configuring multiple connection factories and using targets to assign them to WebLogic Servers. Each connection factory can be deployed on multiple WebLogic Servers.

The application uses the Java Naming and Directory Interface (JNDI) to look up a connection factory and create a connection to establish communication with a JMS server. Each JMS server handles requests for a set of destinations. Requests for destinations not handled by a JMS server are forwarded to the appropriate server.

You can configure multiple JMS servers on the various nodes in the cluster as long as you give them different names. You can assign destinations to the various JMS servers.

One problem to be aware of is the propagation delay in replicating entries in JNDI. If you have an MDB deployed on one node but reference a destination on another node, the deployment may fail with a javax.naming.NamingException exception. The problem occurs because the server is not synced up to the JNDI from the remote server (JMS server) yet, so the JNDI lookup of destination as part of MDB deployment will fail. One workaround is for each MDB to reference a local destination. Another approach is deploy the MDBs after the server boots (plus a delay for JNDI propagation). To get around losing messages before the MDB is deployed, use durable subscribers. This problem is fixed for MDBs in WLS 6.1, where the MDB will be deployed and reconnection will be retried until the destination is available. Note that this is still a problem for EJBs in general that try to reference a non-local JMS destination.

How do I do HTTP tunneling?

If you want to use HTTP tunneling (wrap every message in HTTP to get through a firewall), you need to add TunnelingEnabled="true" into your &lr;ver> definition in the config.xml file or check the appropriate box on the console. Then use a URL like http://localhost:7001 instead of t3://localhost:7001 for Context.PROVIDER_URL when getting your InitialContext. If you want HTTP tunneling with SSL, use https://localhost:7002 (where https uses HTTP tunneling with SSL and 7002 is the secure port that you configured). You will pay a performance penalty for doing this, so only use tunneling it if you really need to (i.e., need to go through a firewall).

Which of the following statements are true regarding the identity of two EJBs?

a. Two stateful session beans are identical if their data attributes are identical.
b. Two stateful session beans are identical if their session contexts are equal.
c. Two stateless session beans are identical if they are of the same type.

d. Two stateless session beans are identical if their session contexts are equal.
e. Two entity beans are identical if they have same primary key but different home interface.
f. Two entity beans are identical if they have different primary key but same home interface.

B and C are correct. Since the stateful session beans maintain the conversational state of the clients, they are identical when their session contexts are equal. Two stateful session beans may have identical data attributes, but if the session contexts are different they are not identical. Thus choice A is incorrect and B is correct. Since stateless beans do not retain the conversational state, they are considered identical if they are of the same type. Thus choice C is correct.
If two entity objects have the same home interface and primary key, they are considered identical. The EJB specification does not mention object equality based on the = = operator. Also, if you compare two object references using the Java API, Object.equals(Object obj), the result is unspecified. The only way to compare object equality is through the isIdentical (EJBObject) API. Thus choice E and F are incorrect.

Why is my JMS work not part of a user transaction (i.e., called within a transaction but not rolled back appropriately)? How do I track down transaction problems?
Usually this problem is caused by explicitly using a transacted session which ignores the external, global transaction by design (JMS spec requirement). A transacted JMS session always has its own inner transaction. It is not affected by any transaction context that the caller may have.

It may also be caused by using a connection factory that is configured with "UserTransactionsEnabled" set to false.
1. You can check if the current thread is in a transaction by adding these two import lines:
import javax.transaction.*
import weblogic.transaction.*;

and adding the following lines (i.e., just after the begin and just before every operation).

```
Transaction tran = TxHelper.getTransaction();
System.out.println(tran);
System.out.println(TxHelper.status2String(tran.getStatus()));
```

This should give a clear idea of when new transactions are starting and when infection is occurring.

2. Ensure that the thread sending the JMS message is infected with a transaction. Check that the code is not using a transacted session by setting the first parameter of createQueueSession or createTopicSession to false. Note that creating the connection and/or session is orthogonal to the transaction. You can begin your transaction before or after. You need only start the transaction before you send or receive messages.

3. Check that the UserTransactionsEnabled flag is explicitly set to true for the connection factory in the config.xml file since the default for user-configured connection factories for this value is false. If you are

using one of the pre-configured connection factories they are set as follows:

weblogic.jms.ConnectionFactory disables user transactions so
don't use this one for the case where user transactions are
desired;

javax.jms.QueueConnectionFactory and
javax.jms.TopicConnectionFactory enable user transactions.

4. You can trace JTA operations by starting the server with this additional property:
-Dweblogic.Debug.DebugJMSXA=true
You should see trace statements like these in the log:
XA ! XA(3163720,487900)
This can be used to ensure that JMS is infected with the transaction.

In the WebLogic server, if stateless session bean instances are getting frequently created and removed, performance can improved by setting a high value for which of the following?

a. max-beans-in-free-pool
b. max-beans-in-cache
c. max-beans-in-memory
d. max-stateless-beans-in-cache

Choice A is correct. WebLogic Server maintains a free pool of EJBs for every stateless session bean class. The max-beans-in-free-pool element defines the size of this pool. By default, max-beans-in-free-pool has no limit; the maximum number of beans in the free pool is limited only by the available memory.
When EJBs are created, the session bean instance is created and given an identity. When the client removes a bean, the bean instance is placed in the free pool. When you create a subsequent bean, you can avoid object allocation by reusing the previous instance that is in the free pool. So the max-beans-in-free-pool element can improve performance if EJBs are frequently created and removed. Keeping this parameter too high uses extra memory and keeping it too low causes unnecessary object creation. WebLogic Server allows you to configure the number of active beans that are present in the EJB cache (the in-memory space where beans exist). The max-beans-in-cache element specifies the maximum number of objects of this class that are allowed in memory. When max-bean-in-cache is reached, WebLogic Server passivates some EJBs that have not been recently used by a client. Choices C and D are not valid properties.

How can an application do a JMS operation and have it succeed, independent of the result of the transaction?
Basically, the JMS operation must be done using a transacted session or the transaction must be suspended/disabled as follows (pick one or more of the following).

1. Suspend the current transaction prior to making the JMS call and resume it after completing it. The code looks something like this:

```
import javax.transaction.Transaction;
import javax.transaction.TransactionManager;

TransactionManager tranManager=
TxHelper.getTransactionManager();
Transaction saveTx = null;
try {
saveTx = tranManager.suspend();
... do JMS work, it will not participate in transaction
} finally {
// must always resume suspended transactions!
if (saveTx != null) tranManager.resume(saveTx);
```

}

2. Use a transacted session by specifying true for the first parameter to createQueueSession or createTopicSession.

3. Use a connection factory with user transactions disabled. That is, check that the UserTransactionsEnabled flag is explicitly set to false for the connection factory in the config.xml file or use the default for a user-configured connection factory for this value which is false. The pre-configured connection factory weblogic.jms.ConnectionFactory disables user transactions.

A transacted JMS session always has its own inner transaction. It is not affected by any transaction context that the caller may have. A non-transacted JMS session is more complicated. If you use the WLS 6.1 default factory weblogic.jms.ConnectionFactory, the session does not participate in a user transaction because the UserTransactionsEnabled flag is set to "False". If you use the deprecated default factory javax.jms.QueueConnectionFactory or javax.jms.TopicConnectionFactory or you define your own factory and set the UserTransactionsEnabled flag to "True", the JMS session participates in the outer transaction, if one exists and the JMS session is not transacted.

What happens if acknowledge() is called within a transaction?
As per the JMS specification, when you are in a transaction, the acknowledgeMode is ignored. If acknowledge() is called within a transaction, it is ignored.

Is it possible to set aside a message and acknowledge it later?
There are no special primitives for doing this. Here are two possible solutions.

One approach is to use multiple sessions as in the following:

```
while (true) {
Create a session, subscribe to one message on durable
subscription
Close session
Save session reference in memory
To acknowledge the message, find the session reference and call
acknowledge() on it.
}
```

Another solution is to use transactions and suspend the work as follows:

```
start transaction
while(true) {
message = receive();
if (message is one that I can handle)
process the message
commit
} else {
suspend transaction
put transaction aside with message
start transaction
}
}
```

To "acknowledge" the message:

```
resume user transaction
commit
```

To "recover" the message:
```
resume user transaction
rollback
```

Each time you suspend, you need to push the transaction onto a stack or list possibly with the message so you can process it or roll it back later. This solution is high overhead in that there can be a large build up of outstanding transactions. Note that transactions have timeouts and it may rollback on its own, which means you can get the message again (in a different transaction). Note also that there are some practical limits on the number of transactions you should leave outstanding. The default limit is something like 10000. Eventually you want to go back to your stack/list and commit/rollback the transactions. Note that transaction references (javax.transaction.Transaction) are not Serializable.

How should I use sorted queues?
Destination keys are used to define the sort order for a specific destination. Destination keys can be message header or property fields. For a list of valid message header and property fields, refer to the Programming WebLogic JMS.
Queues can be sorted in ascending or descending order based on the destination key. A destination is considered to be first-in-first-out if a destination key is defined as ascending for the JMSMessageID message header field, and last-in-first-out if defined as descending. The key defined for the JMSMessageID header field, if specified, must be the last key defined in the list of keys.
You can define multiple destination keys to sort a destination.

Which of the following attributes in the Monitoring tab for a JDBC connection pool in the Administrative console tell us how many clients are currently waiting for a connection?
a. Waiters high
b. Waiters
c. Connections high
d. Clients
e. Wait seconds high

Choice B is correct. JDBC subsystem resources can also be monitored via the Administration Console. The Monitoring tab for a JDBC connection pool allows you to access a table listing statistics for the instances of that pool. These attributes provide important information for managing client database access.
The Waiters High field indicates the highest number of clients waiting for a connection at one time. The Waiters field tells you how many clients are currently waiting for a connection. The Connections High field indicates the highest number of connections that have occurred at one time. The Wait Seconds High field tells you the longest duration a client has had to wait for a database connection. These attributes allow you to gauge the effectiveness of the current configuration in responding to client requests.

The MaxPostTimeSecs attribute set in the Administration console under Servers or virtual hosts section corresponds to which of the following?
a. The amount of time that WebLogic Server waits between receiving chunks of data in an HTTP POST.
b. The total amount of time that WebLogic Server spends receiving HTTP POST data.
c. The time spent by WebLogic server to post data to other servers in the cluster.
d. The number of bytes of data received in a POST from a single request.

Choice B is correct. Web servers may face denial-of-service attacks, which is usually carried out by sending huge amounts of data in an HTTP POST method. You can set three attributes in WebLogic Server that help prevent this type of attack. These attributes are set in the console, under Servers or virtual hosts. You can limit the amount of time that WebLogic Server waits between receiving chunks of data in an HTTP POST by setting the attribute PostTimeoutSecs.

The MaxPostTimeSecs attribute limits the total amount of time that WebLogic Server spends receiving post data. If this limit is triggered, a PostTimeoutException is thrown and a message is sent to the server log. MaxPostSize attribute limits the number of bytes of data received in a POST from a single request. If this limit is triggered, a MaxPostSizeExceeded exception is thrown and a message is sent to the server log.

How does sorting on message priority work?

First, you need to add a key to the destination (by default, they are not sorted), choosing JMSPriority as the key. If you want 0 to be your highest priority, make the key ascending. If you want 9 to be the highest priority, make the key descending.
Second, the priority must be set using either the producer or on the send, not the message.
Third, the priority sorting only comes into play if there are multiple messages waiting on the queue. If the receiver is always caught up with the sender, then the messages will be processed in the order in which they come in.

How do I get a thread dump to help track down a problem?

Ways to get a thread dump:
* Try running this from the command line (after running the setEnv script in /bea/wlserver6.1/config/mydomain/):

```
java weblogic.Admin -url t3://localhost:7001 THREAD_DUMP
```

* On Windows, from the console window, enter Ctrl+Break.
* On UNIX, signal the server using kill -3.

How do I manage a queue to view and delete specific messages?

Write a program that uses a QueueBrowser. Then delete specific messages by using a selector with the message identifier as in the following example:

```
String selector = "JMSMessageID = '" + message.getMessageID() + "'";
```

Keep in mind that the queue browser is a not a "live" view of the queue. It is a snap-shot.

Why do I get an exception when trying to find a connection factory?

The exception is usually something like java.io.InvalidClassException or java.lang.NoClassDefFoundError.
Make sure weblogic.jar is in the CLASSPATH of the client. Also make sure you have the correct Java run-time jar files included (i.e., you might need rt.jar).

What precautions should I take when I use blocking receive() calls?

If your application design requires messages to be received synchronously, we recommend using one of the following methods listed in order of preference:
* Pass a timeout value as an argument to the receive() method and set it to the minimum value greater than zero, that is allowed by the application to avoid consuming threads that are waiting for a response from the server.
* Use the receiveNoWait() method which returns the next message or a null value if no message is currently available. In this case, the call does not block. The servlet should provide a way to return to or reschedule the request, without calling wait().
Note: Use of this option should be minimized, as it may deadlock a busy server.
* Ensure that more threads are configured than the number of possible simultaneous blocking receive() calls.

What is the NO_ACKNOWLEDGE acknowledge mode used for?

The NO_ACKNOWLEDGE acknowledge mode indicates that received messages do not need to be specifically acknowledged which improves performance, but risks that messages are lost. This mode is supported for applications that do not require the quality of service provided by session acknowledge and that do not want to incur the associated overhead. v Messages sent to a NO_ACKNOWLEDGE session are immediately deleted from the server. Messages received in this mode are not recovered and, as a result, messages may be lost and/or duplicate message may be delivered if an initial attempt to deliver a message fails.
Note: You should avoid using this mode if your application cannot handle lost or duplicate messages. Duplicate messages may be sent if an initial attempt to deliver a message fails.
In addition, we do not recommend that this acknowledge mode be used with persistent messaging, as it implies a quality of service that may be too low for persistent messaging to be useful.

When should I use multicast subscribers?
Multicasting enables the delivery of messages to a select group of hosts that subsequently forwards the messages to multicast subscribers. The benefits of multicasting include:

* Near real-time delivery of messages to host group.
* High scalability due to the reduction in the amount of resources required by the JMS server to deliver messages to multicast subscribers.

Note: Multicasting is only supported for the Pub/sub messaging model.

For an example of when multicasting might be useful, consider a stock ticker. When accessing stock quotes, timely delivery is more important than reliability. When accessing the stock information in real-time, if all, or a portion, of the contents is not delivered, the client can simply request the information be resent. Clients would not want to have the information recovered in this case because by the time it is redelivered it would be out-of-date.
Multicast messages are not guaranteed to be delivered to all members of the host group. For messages requiring reliable delivery and recovery, you should not use multicasting.

When should I use server session pools and connection consumers?
WebLogic JMS implements an optional JMS facility for defining a server-managed pool of server sessions. This facility enables an application to process messages concurrently. A ConnectionConsumer object uses a server session to process received messages. If message traffic is heavy, the connection consumer can load each server session with multiple messages to minimize thread context switching. Multiple connection consumers can share server sessions in a server session pool.
To learn how to use the connection consumers within an application, see the section Processing Messages Concurrently in Programming WebLogic JMS, or the javax.jms.ConnectionConsumer javadoc.
Note: Server session pools can also be implemented using Message Driven Beans. Using MDBs is preferable to using server session pools - see the answer to the question, "How do server session pools and Message Driven Beans compare?" For information on using message driven beans to implement server session pools, see Programming WebLogic Enterprise JavaBeans.

How do I issue the close() method within an onMessage() method call and what are the semantics of the close() method?
If you wish to issue the close() method within an onMessage() method call, the system administrator must select the Allow Close In OnMessage check box when configuring the connection factory. For more information, see JMS Connection Factories in the Administration Console Online Help. If this check box is not selected and you issue the close() method within an onMessage() method call, the call will hang.

The close() method performs the following steps to execute an orderly shutdown:

* Terminates the receipt of all pending messages. Applications may return a message or null if a message was not available at the time of the close.

* Waits until all message listeners that are currently processing messages have completed (except for the message listener from which the close() method is being called).

* Rolls back in-process transactions on its transacted sessions (unless such transactions are part of an external JTA user transaction).

* Does not force an acknowledge of client-acknowledged sessions. By not forcing an acknowledge, no messages are lost for queues and durable subscriptions that require reliable processing.

When you close a connection, all associated objects are also closed. You can continue to use the message objects created or received via the connection, except the received message's acknowledge() method. Closing a closed connection has no effect.
Note: Attempting to acknowledge a received message from a closed connection's session throws an IllegalStateException.

When you close a session, all associated producers and consumers are also closed.
For more information about the impact of the close() method for each object, see the appropriate javax.jms javadoc.

How do I publish an XML message?

Follow these steps:
1. Generate XML from the DOM document tree.
2. Serialize the generated DOM document to a StringWriter.
3. Call toString on the StringWriter and pass it into message.setText.
4. Publish the message.

A client wants to preserve the reference to the EJBhome object of an enterprise bean instance and use it later. Which of the following can be serialized for this purpose ?

a. home
b. Handle
c. homeHandle
d. EJBhomeHandle
e. homeObject

Choice C is correct. Once a client has obtained the EJBhome object for an EJB instance, it can create a reference to the home object by calling gethomeHandle(). gethomeHandle() returns a homeHandle object, which can be used to obtain the home interface to the same EJB instance at a later time.
A client can pass the homeHandle object as arguments to another client, and the receiving client can use the handle to obtain a reference to the same EJBhome object. Clients can also serialize the homeHandle and store it in a file for later use. The homeHandle interface has only one method getEJBhome(), which returns the EJBhome reference.

Is it possible to send or receive a message from within a message listener?

Yes. You can send to or receive from any queue or topic from within in a message listener.
If it's not an MDB, you can use the same Connection or Session that the onMessage() is part of to do this. When you create your message listener, you pass in a session in your constructor. Then you have access to the session in your onMessage method and you would be able to make synchronous, not asynchronous, calls from within the onMessage method. Do not use another Session that is servicing another onMessage() because that would multi-thread that Session and Sessions don't support multi-threading.
When things are done non-transactionally, there can be duplicates or lost messages (assuming your onMessage() code is attempting to forward messages):
1. If you call acknowledge after the publish() and the acknowledge fails for whatever reason (network/server failure), then you will see the message again and will end up publishing twice (possible duplicate semantics). You can try to keep track of sequence numbers to detect duplicates but this is not easy.
2. If you call acknowledge before the publish(), you get at-most-once semantics. If the publish() fails, you don't know if the failure occurred before or after the message reached the server.
If you want exactly once, transactional semantics using onMessage, you must use transactional MDBs. The onMessage() for a transactional MDB starts the transaction, includes the WebLogic Server JMS message received within that transaction and the publish() would also be in the same transaction. The following code sends a response to each message that it receives. It creates the connection, etc. in the ejbCreate method so that it doesn't need to create it every time onMessage is called. The QueueSender is anonymous (null Queue) since we don't know to whom we will have to reply. The ejbRemove method cleans up by closing the connection. This same approach can be used to create a receiver, subscriber or publisher.

```
import javax.ejb.CreateException;
import javax.ejb.EJBContext;
import javax.naming.*;
import javax.naming.directory.*;
import java.util.Hashtable;
import javax.ejb.MessageDrivenBean;
import javax.ejb.MessageDrivenContext;
import javax.jms.*;
```

```
public class MDB
implements MessageDrivenBean, MessageListener {
public static final String WLSqcf =
"javax.jms.QueueConnectionFactory";
public static final String WLSqname =
"jms.queue.TestQueue1";
public static final String WLSurl =
"t3://localhost:7001";
public static final String WLSJNDIfactory =
"weblogic.jndi.WLInitialContextFactory";
private MessageDrivenContext context;
private QueueSession session;
private QueueConnection connection = null;
private QueueConnectionFactory factory;
private InitialContext ctx;
private QueueSender QueueSender;

// Required - public constructor with no argument
public MDB() {}

// Required - ejbActivate
public void ejbActivate() {}
// Required - ejbRemove
public void ejbRemove() {
context = null;
if (connection != null) {
try {
connection.close();
} catch(Exception e) {}
connection = null;
}
}

// Required - ejbPassivate
public void ejbPassivate() {}

public void setMessageDrivenContext(
MessageDrivenContext mycontext) {
context = mycontext;
}

// Required - ejbCreate() with no arguments
public void ejbCreate () throws CreateException {
try {
// Get the initial context
Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY, WLSJNDIfactory);
env.put(Context.PROVIDER_URL, WLSurl);
env.put(Context.REFERRAL, "throw");
ctx = new InitialContext(env);

factory = (QueueConnectionFactory)ctx.lookup(WLSqcf);

// Create a QueueConnection, QueueSession, QueueSender
connection = factory.createQueueConnection();
session = connection.createQueueSession(false,
Session.AUTO_ACKNOWLEDGE);
queueSender = session.createSender(null);
connection.start();
} catch (Exception e) {
throw(new CreateException(e.toString()));
}
}

// Implementation of MessageListener
// Throws no exceptions
public void onMessage(Message msg) {
try {
System.out.println("MDB: " +
((TextMessage)msg).getText());
msg.clearBody();
((TextMessage)msg).setText("reply message");
queueSender.send((Queue)msg.getJMSReplyTo(), msg);
}
catch(Exception e) { // Catch any exception

e.printStackTrace();
}
}
}
```

This approach creates a connection per EJB/MDB instance, so you might want to create a producer pool that is shared by the EJB instances. This is done by writing a class that populates a static pool with producers (see the next question for a sample producer pool). The onMessage call grabs a producer when needed. Since Sessions must be single threaded, make sure there is only one producer per session within the producer pool.

How do I create a producer pool?

The following is some pseudo-code
for a producer class.

```
class ProducerPool {
static Hashmap pSets = new Hashtable();
static Hashmap inUse = new Hashtable();

QueueSender get(String contextURL,
String connectionFactoryName,
String destinationName) {
String lookup = contextURL+";
"+connectionFactName+";"+destName;
synchronized(pSets) {
producer set = pSets.get(lookup);
if (set != null && set not empty)
qs = set.removeFirst();
}
if (producer == null) {
create ctx
get connect factory
create connection
create session
look up destination
qs = create queue sender
}
synchronized(inUse) {
inUse.put(qs, lookup);
}
return qs;
}

void put(QueueSender qs) {
String lookup;
synchronized(inUse) {
lookup = inUse.remove(p);
}
synchronzied(pSets) {
producer set = pSets.get(lookup);
if (set == null) {
producer set = new producer set
pSets.put(lookup, producer set);
}
producer set.add(qs);
}
}
}
```

Note: Static classes may be garbage collected if there are no references to them, so make sure the application server has a permanent pointer to them in some manner. One way is to reference it permanently from within a servlet or EJB when they are initialized at startup.
Here is an example of using the producer pool within the onMessage method.

```
onMessage() {
QueueSender qs = ProducerPool.get(...);
qs.send(...);
ProducerPool.put(qs);
}
```

You can pre-populate this pool by calling it from
a startup class or a load-on-start servlet class.

What are pending messages in the console?

Pending means the message could have been:
* sent in a transaction but not committed.
* received and not acknowledged.
* received and not committed.
* subject to a redelivery delay (as of WebLogic Server 6.1).
* subject to a delivery time (as of WebLogic Server 6.1).
A rolled back message remains pending until the transaction actually rolls back. Rolling it back multiple times does not cause double counting, nor does an exception that set a transaction as rollbackOnly followed by an actual rollback.
Current implies messages that are not pending.
Total implies total since server last started. The byte counts only consider the payload of messages which includes the properties and the body but not the header.

How do I use a less than or greater than on a message selector in ejb-jar.xml?

Enclose the selector in a CDATA section. That will prevent the XML parser from thinking that less than or greater than is a tag.

```
<jms-message-selector>
'user' ]]>
```

Is it better to have more or fewer sessions for a given number of subscribers?

Using N sessions for N subscribers gives you concurrency up to N simultaneous threads of execution

provided you have as many threads to work with. Each Session gets its own thread as long as there are enough threads available. Otherwise, the sessions serially reuse the available threads.
One session for N subscribers serializes all subscribers through that one session. If the load is heavy they may not be able to keep up without the extra threads.
If you are using CLIENT_ACKNOWLEDGE, N sessions gives you N separate message streams that can be individually recovered. Having one session crosses the streams giving you less control.

Match the EJB functions given below with the functionality equivalent in SQL

A.) ejbStore() 1.) INSERT
B.) ejbLoad() 2.) UPDATE
C.) ejbCreate() 3.) SELECT
a. A->1, B->2, C->3
b. A->2, B->1, C->3
c. A->3, B->2, C->1
d. A->1, B->3, C->2
e. A->2, B->3, C->1
f. A->3, B->1, C->2

Choice E is correct. When the create() method on a home interface is invoked, the container delegates the create() method call to the bean instance's matching ejbCreate() method. The ejbCreate() methods are used to initialize the instance state before record is inserted into the database. The ejbCreate() method is analogous to INSERT. The ejbStore() method is invoked just before the container the container is about to write the bean container-managed fields to the database. It is analogous to the UPDATE . The ejbLoad() is invoked just after the container has refreshed the bean container-managed files with its state from the database. It is analogous to the SELECT. Thus choice E is correct and others are not.

A client invokes a method on a stateful session bean instance deployed in the WebLogic Server. While the method execution is in progress another method call arrives on the server. What will be the result?

a. RemoteException is thrown if the value of concurrency-strategy property is false
b. EJBException is thrown if the value of concurrency-strategy property is false
c. The EJB container blocks the concurrent method call and allows it to proceed when the previous call has completed if the value of allow-concurrent-calls is true
d. In all cases, RemoteException is thrown

Choice C is correct. By default, simultaneous access to a stateful session EJB results in a RemoteException. However, you can set the allow-concurrent-calls option in the WebLogic EJB deployment descriptor to specify that a stateful session bean instance will allow concurrent method calls. This access restriction on stateful session EJBs applies whether the EJB client is remote or internal to WebLogic Server. By default, allows-concurrent-calls is false. However, when this value is set to true, the EJB container blocks the concurrent method call and allows it to proceed when the previous call has completed.
The concurrency-strategy element determines ejbLoad() and ejbStore() behavior for entity EJB instances.