Oracle

Exam 1z0-895

Java EE 6 Enterprise JavaBeans Developer Certified Expert Exam

Verson: Demo

[Total Questions: 10]

Question No : 1

A developer wants to package an enterprise bean FooBean within a .war file:

```
@Stateless
public xlass FooBean {

public void foo () {}
}
```

Which package approach is correct?

A. / (Root)
I – META – INF /
I – acme
I – FooBean.class
B. / (Root)
I – acme
I – FooBean.class
C. / (Root)
I – WEB – INF /
I – acme
I – FooBean.class
D. / (Root)
I – WEB – INF /
I – Classes/
I – acme
I – FooBean.class

Answer: D

Explanation: To include enterprise bean class files in aWARmodule, the class files should be in the
WEB-INF/classes directory.

Note: Enterprise beans often provide the business logic of a web application. In these cases, packaging the enterprise bean within the web application'sWARmodule simplifies deployment and application organization. Enterprise beans may be packaged within aWARmodule as Java
programming language class files or within a JAR file that is bundled within theWARmodule.

Reference: The Java EE 6Tutorial, Packaging Enterprise Beans inWAR Modules

Question No : 2

MyMsg is a JMS message-driven bean with container-managed transaction demarcation. FooBean is an EJB 3.x stateless session bean that sends message to the JMS destination with MyMsgBean is associated.

MyMsgBean's message listener method has transaction attribute REQUIRED, and is defined as follows:

```
10. public class MyMsgBean implements javax.jms.messageListener {
11. public void onMessage(javax.jms.Message message) {
12. / / do some work not shown here
13. thrown new RuntimeException("unexpected error . . . ");
14. }
```

Which statement is true about the result of message processing?

A. FooBean receives javax.ejb.EJBException.
B. The container discards the MyMsgBean bean instance.
C. FooBean receives the original RuntimeException thrown from the message listener method.
D. The container does NOT roll back the transaction, and FooBean can continue the transaction.

Answer: C

Explanation:

Note: public interface MessageListener

A MessageListener object is used to receive asynchronously delivered messages.

Each session must insure that it passes messages serially to the listener. This means that a listener assigned to one or more consumers of the same session can assume that the onMessage method is not called with the next message until the session has completed

the last call.

Reference: Enum TransactionAttributeType

Question No : 3

A developer writes three interceptor classes: AInt, BInt, and CInt. Each interceptor class defines an AroundInvoke method called interceptor. In the ejb-jar.xml descriptor, CInt is declared as the default interceptor.

FooBean is a stateless session bean with a local business interface Foo that declares a method Foo ():

10. @Stateless

11. @Interceptors(AInt.class)

12. public class FooBean Implements Foo {

13.

14. @Interceptors (BInt.class)

15. @ExcludeClassInterceptors

16. public void foo () {}

17. }

What is the interceptor order when the business method foo () is invoked?

A. BInt
B. CInt, BInt
C. CInt, AInt, BInt
D. BInt, AInt, CInt

Answer: B

Explanation: The default Intercepter, CInt, comes first. The class intercepter AInt is excluded by @ExcludeClassInterceptors, so the Method Intercepter BInt would be next in

order.

Note 1: By default the ordering of interceptors when invoking a method are

* External interceptors
** Default interceptors, if present
** Class interceptors, if present
** Method interceptors, if present
*Bean class interceptor method
Note 2: Annotation Type ExcludeClassInterceptors
Used to exclude class-level interceptors for a business method or timeout method of a target class.

Reference: EJB Interceptors

http://docs.jboss.org/ejb3/app-server/tutorial/interceptor/interceptor.html

Question No : 4

Suppose a developer wants to create an automatic persistent timer that performs data validation every hour. Given the following stateless session bean:

@Stateless

Public class OrderVerificationBean {

Private void verificationExternalOrders () {

/ / do something

}

}

What is the minimum modification you would need to make to the bean to create the automatic persistent timer?

A. Modify the verifyExternalOrders methos to look like this:
@Schedule

private void verifyExternalOrders () {
/ do something
}

B. Modify the verifyExternalOrders method to look like this:
@Schedule (hour = “*”)
private void verifyExternalOrders () {
/ / do something
}

C. Modify the verifyExternalOrders method to look like this:
@Schedule (persistent = true)
private void verifyExceptionalOrders () {
/ / do something
}

D. Modify the verifyExternalOrders method to look like this:
@Schedule (hour = “*”, persistent = true)
private void verifyExceptionalOrders () {
/ / do something
}

Answer: B

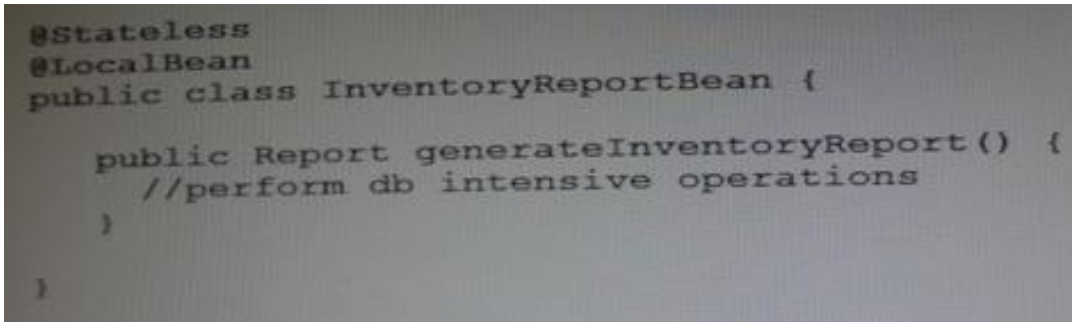
Explanation:

Not D: Timers are persistent by default. If the server is shut down or crashes, persistent timers are saved and will become active again when the server is restarted. If a persistent timer expires while the server is down, the container will call the @Timeout method when the server is restarted.

Nonpersistent programmatic timers are created by calling TimerConfig.setPersistent(false) and passing the TimerConfig object to one of the timer-creation methods.

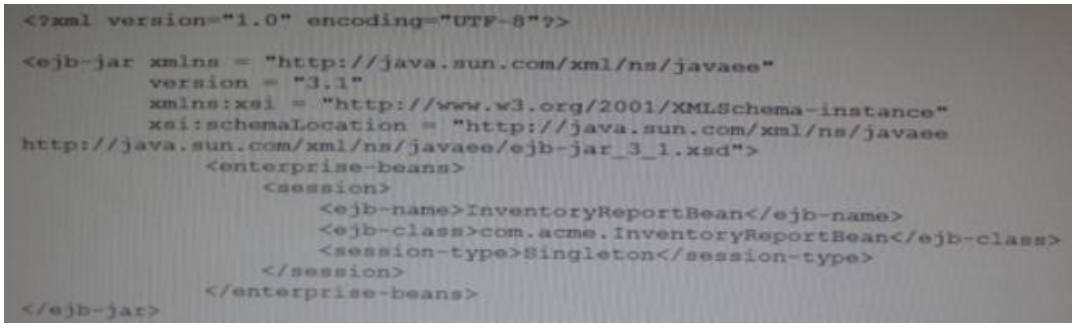
Question No : 5

Assume an EJB application is comprised of the following EJB fragment:



You have been asked to convert the type of InventoryReportBean into a singleton session bean. How would you achieve this task?

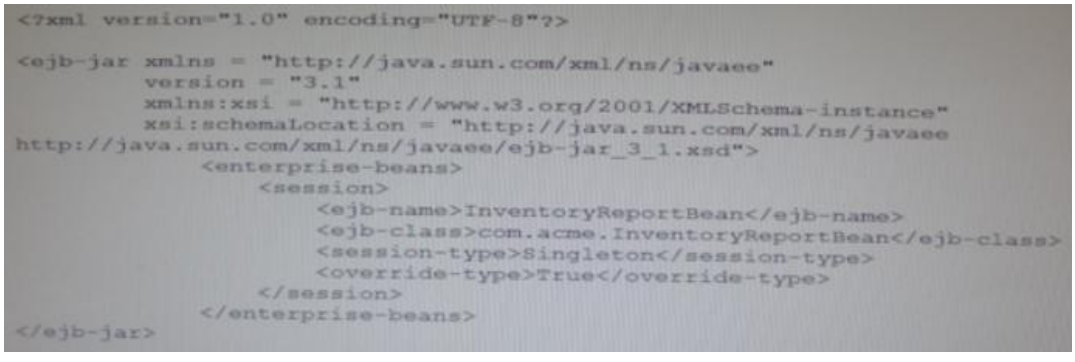
Exhibit C:



```
<?xml version="1.0" encoding="UTF-8"?>

<ejb-jar xmlns = "http://java.sun.com/xml/ns/javaee"
         version = "3.1"
         xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation = "http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/ejb-jar_3_1.xsd">
             <enterprise-beans>
                 <session>
                     <ejb-name>InventoryReportBean</ejb-name>
                     <ejb-class>com.acme.InventoryReportBean</ejb-class>
                     <session-type>Singleton</session-type>
                 </session>
             </enterprise-beans>
</ejb-jar>
```

Exhibit D:



```
<?xml version="1.0" encoding="UTF-8"?>

<ejb-jar xmlns = "http://java.sun.com/xml/ns/javaee"
         version = "3.1"
         xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation = "http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/ejb-jar_3_1.xsd">
             <enterprise-beans>
                 <session>
                     <ejb-name>InventoryReportBean</ejb-name>
                     <ejb-class>com.acme.InventoryReportBean</ejb-class>
                     <session-type>Singleton</session-type>
                     <override-type>True</override-type>
                 </session>
             </enterprise-beans>
</ejb-jar>
```

A. Keep InventoryReportBean as it is, modifying the internal structure to function as a singleton
B. Change the @Stateless annotation of InventoryReportBean to @Singleton
C. Create an ejb-jar.xml file, and override the annotation configuration information asin exhibit C above.
D. Create an ejb-jar.xml file, and override the annotation configuration information asin exhibit D above.

Answer: D

Explanation: Note the line with <override-type>

Question No : 6

Which is a correct way to define a runtime exception as an EJB 3.x application exception?

A. public class MyAppException extends javax.ejb.EJBException
B. @ApplicationException
public class MyAppException extends javax.ejb.EJBException
C. public class MyAppException extends javax.lang.EJBException
D. @ApplicationException
public class MyAppException extends javax.lang.EJBException

Answer: B

Explanation: Use the @javax.ejb.ApplicationException annotation to specify that an exception class is an application exception thrown by a business method of the EJB. The EJB container reports the exception directly to the client in the event of the application error.

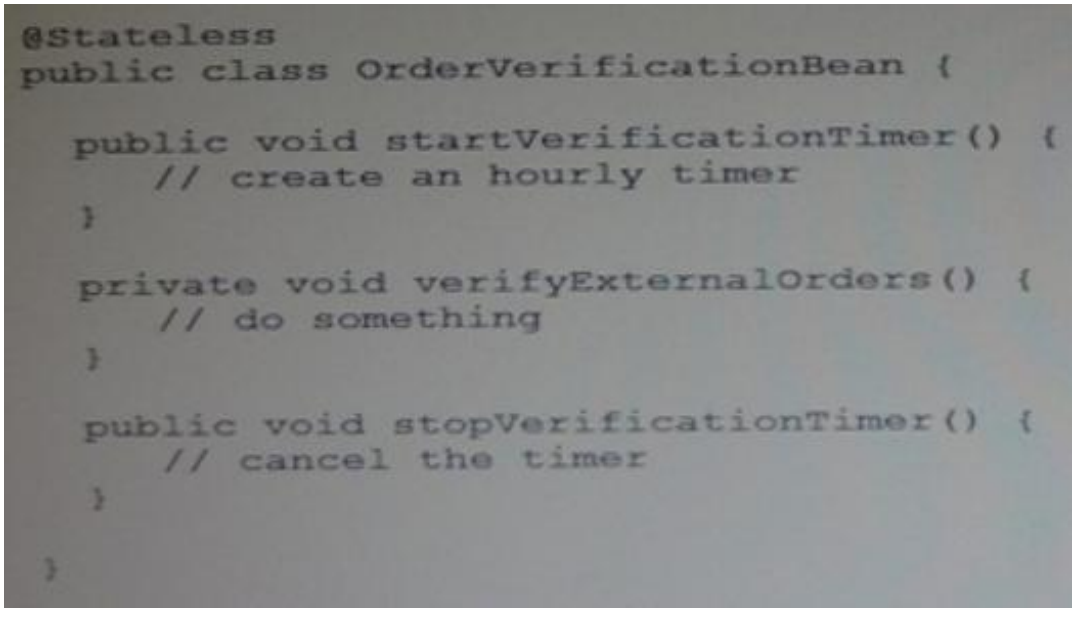
Note:
java.lang.Object
java.lang.Throwable
java.lang.Exception
java.lang.RuntimeException
javax.ejb.EJBException
javax.ejb
public class EJBException
extends java.lang.RuntimeException
The EJBException is thrown to report that the invoked business method or callback method could not be completed because of an unexpected error (e.g. the instance failed to open a database connection).

Example:
The following ProcessingException.java file shows how to use
the @ApplicationException annotation to specify that an exception class is an application exception thrown by one of the business methods of the EJB:
package examples;
import javax.ejb.ApplicationException;
/*** Application exception class thrown when there was a processing error* with a business method of the EJB. Annotated with the* @ApplicationException annotation.*/
@**ApplicationException()**public class ProcessingException extends Exception {

Reference: Programming WebLogic Enterprise JavaBeans, Version 3.0 programming Application Exceptions

Question No : 7

Suppose developer wants to create an EJB component that performs data validation every hour. Given the following Stateless session bean:



```
@Stateless
public class OrderVerificationBean {

  public void startVerificationTimer() {
     // create an hourly timer
  }

  private void verifyExternalOrders() {
     // do something
  }

  public void stopVerificationTimer() {
     // cancel the timer
  }

}
```

What is the minimum modification you would need to make to the bean to support notification from the TimerService once the timer expires?

A. Modify the verify external orders method to look like this:
@TimedOut
private void verifyExternalOrders () {
/ / do something
}
B. Modify the verify external orders method to look like this:
@EjbTimeOut
private void verifyExternalOrders () {
/ / do something
}
C. Modify the verify external orders method to look like this:
@ejbTimeOut
private void verifyExternalOrders () {
/ / do something
}
D. Modify the verify external orders method to look like this:
@TimeOut
private void verifyExternalOrders () {

/ / do something
}

Answer: D

Explanation: Programmatic Timers
When a programmatic timer expires (goes off), the container calls the method annotated @Timeout in the bean's implementation class. The @Timeout method contains the business logic that handles the timed event.

The @Timeout Method
Methods annotated @Timeout in the enterprise bean class must return void and optionally take a javax.ejb.Timer object as the only parameter. They may not throw application exceptions.

@Timeout
public void timeout(Timer timer) {
System.out.println("TimerBean: timeout occurred");
}

Reference: The Java EE 6 Tutorial, Using the Timer Service

Question No : 8

A developer implements a system in which transfers of goods are monitored. Each transfer needs a unique ID for tracking purposes. The unique ID is generated by an existing system which is also used by other applications. For performance reasons, the transaction that gets the unique ID should be as short as possible. The scenario is implemented in four steps which are implemented in four business methods in a CMT session bean:

1.checkGoods	Checks goods in a database
2.getUniqueId	Retrieve the unique ID
3.checkAmount	Checks the amount in a non-transactional system
4. storeTransfer	Stores the transfer in a database as part of the calling transaction.

These methods are called by the addTransfer method of a second CMT session bean in the following order:

checkGooods, getUniqueId, checkAmount, storeTranfer

Assuming no other transaction-related metadata, which is the correct set of transaction attributes for the methods in the session beans?

A.
0.addTransferREQUIRED
1.checkGoodsREQUIRED
2.getUnigueIdREQUIRES_NEW
3.checkAmountsNOT_SUPPORTED
4.storeTransferMANDATORY
B.
0.addTransferREQUIRED
1.checkGoodsREQUIRED
2.getUnigueIdREQUIRED
3.checkAmountsREQUIRED
4.storeTransferREQUIRED
C.
0.addTransferREQUIRED
1.checkGoodsREQUIRED
2.getUnigueIdREQUIRES_NEW
3.checkAmountsNEVER
4.storeTransferMANDATORY
D.
0.addTransferNOT_SUPPORTED
1.checkGoodsREQUIRED
2.getUnigueIdREQUIRES_NEW
3.checkAmountsNOT_SUPPORTED
4.storeTransferMANDATORY

Answer: D

Explanation:

Step 2: Must start a new transaction. use REQUIRES_NEW

Step 3: No need for this step: use Not Supported

Use the NotSupported attribute for methods that don't need transactions. Because transactions involve overhead, this attribute may improve performance.

Step 4: Use Mandatory:

Use the Mandatory attribute if the enterprise bean's method must use the transaction of the client.

Note:
*In an enterprise bean with container-managed transaction(CMT)demarcation, the EJB container sets the boundaries of the transactions. You can use container-managed transactions with any type of enterprise bean: session, or message-driven. Container-managed transactions simplify development because the enterprise bean code does not explicitly mark the transaction's boundaries. The code does not include statements that begin and end the transaction.

*A transaction attribute can have one of the following values:

Required

RequiresNew

Mandatory

NotSupported

Supports

Never

*Required Attribute
If the client is running within a transaction and invokes the enterprise bean's method, the method executes within the client's transaction. If the client is not associated with a transaction, the container starts a new transaction before running the method.

The Required attribute is the implicit transaction attribute for all enterprise bean methods running with container-managed transaction demarcation. You typically do not set the Required attribute unless you need to override another transaction attribute. Because transaction attributes are declarative, you can easily change them later.

*RequiresNew Attribute
If the client is running within a transaction and invokes the enterprise bean's method, the container takes the following steps:

Suspends the client's transaction

Starts a new transaction

Delegates the call to the method

Resumes the client's transaction after the method completes

If the client is not associated with a transaction, the container starts a new transaction before running the method.

You should use the RequiresNew attribute when you want to ensure that the method always runs within a new transaction.

*Mandatory Attribute
If the client is running within a transaction and invokes the enterprise bean's method, the method executes within the client's transaction. If the client is not associated with a transaction, the container throws the TransactionRequiredException.

Use the Mandatory attribute if the enterprise bean's method must use the transaction of the client.

*NotSupported Attribute
If the client is running within a transaction and invokes the enterprise bean's method, the container suspends the client's transaction before invoking the method. After the method has completed, the container resumes the client's transaction.

If the client is not associated with a transaction, the container does not start a new transaction before running the method.

Use the NotSupported attribute for methods that don't need transactions. Because transactions involve overhead, this attribute may improve performance.

Reference:The Java EE 5 Tutorial,Container-Managed Transactions

Question No : 9

Given code snippets from two files:

```
7.  public class Dog {
8.    public void onMessage(Message m) { System.out.print("1 "); }
9.  }
and
10. @MessageDriven
11. class MessageDog extends Dog implements MessageDrivenBean {
12.   MessageDog(Message m) { System.out.print("2 "); }
13. }
```

Which four code changes, when used together, create a valid JMS message-driven bean? (Choose four)

A. Make class MessageDog public
B. Make the MessageDog constructor no-arg
C. Make the MessageDog constructor public
D. Move the onMessage method to class MessageDog.
E. Change MessageDog so that it is NOT a subclass of Dog.
F. Make class MessageDog implement MessageListner instead of MessageDrivenBean.

Answer: A,B,C,F

Question No : 10

A developer implements a CMT session bean with a method storeBoth which inserts data both a related database and an LDAP server. The relational database supports transactions while the LDAP system does NOT.

Given that both updates should succeed or be rolled back, while is the best solution?

A. Implement the SessionSynchoronization interface in the session bean. In the afterCompleteion method, the LDAP inserts are rolled back if false is passed as an argument to the afterCompletion method.
B. Define the transaction attribute of the method storeBoth as REQUIRED. The container manages the transactions and will roll back modifications if something goes wrong in either database insert or LDAP insert.
C. Define the transaction attribute of the method storeBoth as REQUIRED_NEW. Carry out the database insert first. Subsequently, execute the LDAP inserts, catching LDAP exceptions. If exceptions are raised, call the SessionContext.setRollBackOnly method.
D. Define the transaction attribute of the method storeBoth as REQUIRED_NEW. Carry out the LDAP insert first. If SessionContext.getRollBackOnly returns false, execute the database inserts, catching SQL exceptions. If exceptions are raised, call the SessionContext.setRollBackOnly.

Answer: C

Explanation: The method should start a new transaction, so we use theREQUIRED_NEWattribute.

For the LDAP operation we can only detect LDAP exceptions. We cannot check the status of the LDAP operation throughSessionContext.getRollBackOnly.

Note:
* CMT -Container-Managed Transactions
*RequiresNew Attribute
If the client is running within a transaction and invokes the enterprise bean's method, the container takes the following steps:

Suspends the client's transaction

Starts a new transaction

Delegates the call to the method

Resumes the client's transaction after the method completes

If the client is not associated with a transaction, the container starts a new transaction before running the method.

You should use the RequiresNew attribute when you want to ensure that the method always runs within a new transaction.
Reference:The Java EE 5 Tutorial,Container-Managed Transactions

Thank You For Trying 1Z0-895 PDF Demo

To try our 1Z0-895 Premium Files visit link below:

https://examsland.com/latest-1Z0-895-exam-questions/

Start Your 1Z0-895 Preparation

Use Coupon EL25 for extra 25% discount on the purchase of Practice Test Software.