1Z0-895 Q&As

Java EE 6 Enterprise JavaBeans Developer Certified Expert

Pass Oracle 1Z0-895 Exam with 100% Guarantee

Free Download Real Questions & Answers **PDF** and **VCE** file from:

https://www.pass4lead.com/1Z0-895.html

100% Passing Guarantee
100% Money Back Assurance

Following Questions and Answers are all new published by Oracle Official Exam Center

QUESTION 1

A developer writes a stateless session bean with one local business interface and with container- managed transactions. All business methods have transaction attribute REQUIRED. The bean has an injected field sessionCtx of the type SessionContext. Which two operations are allowed in a business method of the bean? (Choose two.)

A. sessionCtx. getEJBObject

B. sessionCtx.setRollbackOnly

C. sessionCtx. getMessageContext

D. sessionCtx. getBusinessObject

E. sessionCtx. getEJBLocalObject

Correct Answer: BD

QUESTION 2

Given the following client-side code that makes use of the session bean Foo:

10. @EJB Foo bean1;

12. @EJB Foo bean2; // more code here

20.

boolean test1 = beanl.equals(bean1);

21.

boolean test2 = beanl.equals(bean2) ; Which three statements are true? (Choose three)

A. If Foo is stateful, test1 is true, and test2 is true.

B. If Foo is stateful, test1 is true, and test2 is false.

C. If Foo is stateless, test1 is true, and test2 is true.

D. If Foo is stateless, test1 is true, and test2 is false.

E. If Foo is singleton, test1 is true, and test2 is true.

F. If Foo is singleton, test1 is true, and test2 is false.

Correct Answer: BCE

QUESTION 3

Assume a client will be accessing a Singleton bean.

Which client views is a Singleton bean capable of exposing? (Choose two)

A. Web Service

B. Message listener

C. EJB 2.x Remote Home

D. EJB 3.x local business

E. Java Persistence API entity

Correct Answer: AB

Singleton session beans are appropriate in the following circumstances.

*

State needs to be shared across the application.

*

A single enterprise bean needs to be accessed by multiple threads concurrently.

*

The application needs an enterprise bean to perform tasks upon application startup and shutdown.

*

The bean implements a web service. (A)

B: An interceptor method you define in a separate interceptor class takes an invocation context as argument: using this context, your interceptor method implementation can access details of the original session bean business method or message-driven bean message listener method invocation. Singleton Interceptors If your interceptors are stateless, you can use an OC4J optimization extension to the EJB 3.0 specification that allows you to specify singleton interceptors. When you configure a session bean or message-driven bean to use singleton interceptors and you associate the bean with an interceptor class, OC4J creates a single instance of the interceptor class that all bean instances share. This can reduce memory requirements and life cycle overhead.

Note:

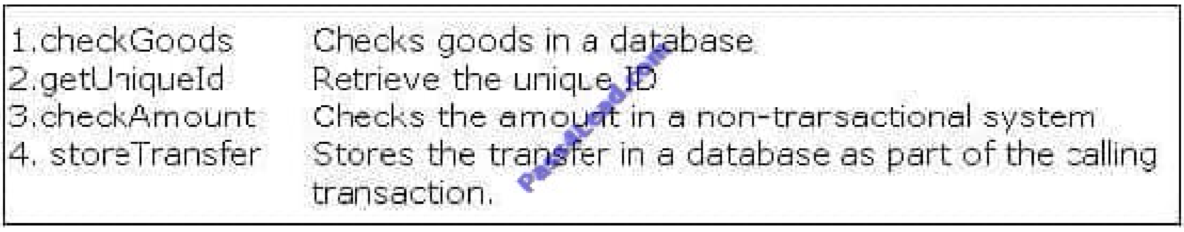
Singleton session beans offer similar functionality to stateless session beans but differ from them in that there is only one singleton session bean per application, as opposed to a pool of stateless session beans, any of which may respond to a

client request. Like stateless session beans, singleton session beans can implement web service endpoints.

Reference: The Java EE 6 Tutorial, What Is a Session Bean? Reference: Oracle Containers for J2EE Enterprise JavaBeans Developer\\'s Guide, How do you use an Enterprise Bean in Your Application

QUESTION 4

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.addTransfer REQUIRED 1.checkGoods REQUIRED 2.getUnigueId REQUIRES_NEW 3.checkAmounts NOT_SUPPORTED 4.storeTransfer MANDATORY

B. 0.addTransfer REQUIRED 1.checkGoods REQUIRED 2.getUnigueId REQUIRED 3.checkAmounts REQUIRED 4.storeTransfer REQUIRED

C. 0.addTransfer REQUIRED 1.checkGoods REQUIRED 2.getUnigueId REQUIRES_NEW 3.checkAmounts NEVER 4.storeTransfer MANDATORY

D. 0.addTransfer NOT_SUPPORTED 1.checkGoods REQUIRED 2.getUnigueId REQUIRES_NEW 3.checkAmounts NOT_SUPPORTED 4.storeTransfer MANDATORY

Correct Answer: A

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.

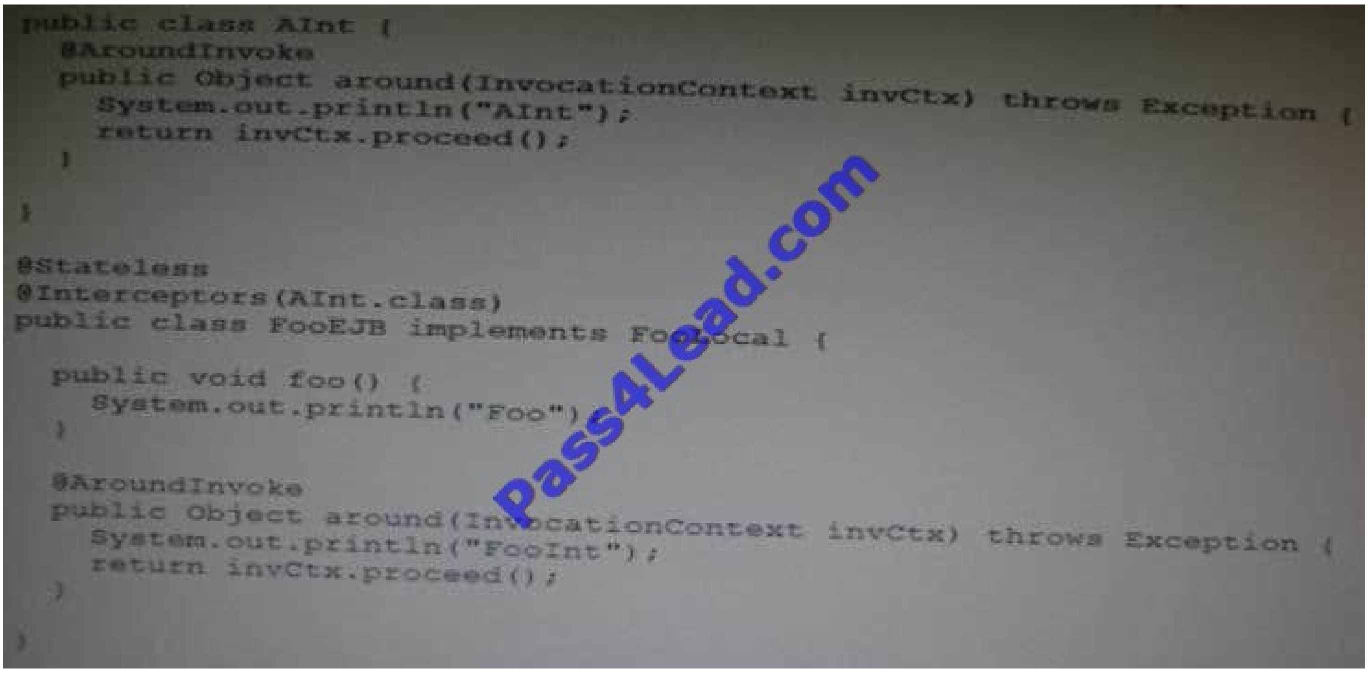
If the client is not associated with a transaction, the container does not start a new transaction before running the method.

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 5

A developer writes an interceptor class and a stateless session bean: A client acquires an EJB reference to the

FooLocal business interface and invokes the foo() method one time. Which describes the output?



```
public class AInt {
  @AroundInvoke
  public Object around(InvocationContext invCtx) throws Exception {
    System.out.println("AInt");
    return invCtx.proceed();
  }

}

@Stateless
@Interceptors(AInt.class)
public class FooEJB implements FooLocal {

  public void foo() {
    System.out.println("Foo");
  }

  @AroundInvoke
  public Object around(InvocationContext invCtx) throws Exception {
    System.out.println("FooInt");
    return invCtx.proceed();
  }

}
```

A. Foo FooInt AInt

B. AInt Foo

C. AInt FooInt Foo

D. FooInt AInt Foo

Correct Answer: C

*

At the end of the chain of interceptors, the actual bean method gets called.

*

Interceptors can be bound in three different ways:

Default Class level Method level

In this question both class level and method level interceptors are used.

The class level interceptor intercepts before the method-level interceptor.

Note:

*

Interceptors are used in conjunction with Java EE managed classes to allow developers to invoke interceptor methods on an associated target class, in conjunction with method invocations or lifecycle events. Common uses of interceptors

are logging, auditing, and profiling.

*

An interceptor can be defined within a target class as an interceptor method, or in an associated class called an interceptor class. Interceptor classes contain methods that are invoked in conjunction with the methods or lifecycle events of the

target class.

Interceptor classes and methods are defined using metadata annotations, or in the deployment descriptor of the application containing the interceptors and target classes.

*

javax.interceptor.AroundInvoke

Designates the method as an interceptor method.

*

The target class can have any number of interceptor classes associated with it. The order in which the interceptor classes are invoked is determined by the order in which the interceptor classes are defined in the

javax.interceptor.Interceptors annotation.

Reference: Introduction to EJB3 Interceptors

Reference: The Java EE 6 Tutorial, Overview of Interceptors