OCEEJBD 6 Practice Test

Started on	Monday, 22 October 2018, 9:22 PM
State	Finished
Completed on	Monday, 22 October 2018, 9:23 PM
Time taken	15 secs
Grade	**0.00** out of 60.00 (**0**%)
Result	

QUESTION 1 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given several class declarations:

```
public class Data {
   private int value = 0;
   // getter and setter
}
@Stateless
@Remote(Whizlabs.class)
public class WhizlabsBean implements Whizlabs {
   public void setInteger(Data data) {
      data.setValue(1);
   }
}
@Stateless
public class MyBean {
   @EJB
   private Whizlabs whizlabs;
   public void processData() {
      Data data = new Data();
      data.setValue(2);
      whizlabs.setInteger(data);
      System.out.println(data.getValue());
   }
}
```

It is known that setInteger(Data) is a method defined by the Whizlabs interface and there is no associated deployment descriptor. What happens when the MyBean.processData is executed?

Please select :

- A. Number 0 is printed on the console
- B. Number 1 is printed on the console
- C. Number 2 is printed on the console
- D. An exception is thrown at runtime

Your answer is incorrect.

Answer: D

Explanation:

As per the EJB 3.1 Specification (subsection 3.2.3), objects that are passed as parameters on remote calls must be serializable. The Data class in the given scenario does not implement the Serializable interface, thus it fails to meet the requirement.

The correct answer is: An exception is thrown at runtime

Submit your Feedback/Queries to our Experts

QUESTION 2 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given an interface and a bean class that implements this interfaces:

```
public interface Whizlabs {
   public void myMethod();
}
@Stateless
@LocalBean
public class WhizlabsBean implements Whizlabs {
   public void myMethod() { ... }
   // other declarations
}
```

}

Assume there is no associated deployment descriptor. Which TWO of the following are valid ways to inject the above bean into a local client?

Please select :

- A. @EJB(beanInterface = Whizlabs.class)
  private Whizlabs whizlabs;
- B. @EJB(beanInterface = WhizlabsBean.class)
  private Whizlabs whizlabs;
- C. @EJB(beanInterface = WhizlabsBean.class)
  private WhizlabsBean whizlabs;
- D. @EJB(beanInterface = Whizlabs.class)
  private WhizlabsBean whizlabs;

Your answer is incorrect.

Answer: B and C

Explanation:

The Whizlabs interface is not explicitly specified as a local or remote business interface. Thus, WhizlabsBean exposes only the no-interface client.

The beanInterface element of the @EJB annotation must be set to a client view of an enterprise bean. As such, Whizlabs.class cannot be specified here.

The correct answers are: @EJB(beanInterface = WhizlabsBean.class)
private Whizlabs whizlabs;, @EJB(beanInterface = WhizlabsBean.class)
private WhizlabsBean whizlabs;

Submit your Feedback/Queries to our Experts

QUESTION 3 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given a session bean:

```
@Stateless
public class MyBean {
   @EJB(name = "whizlabsName", beanName = "whizlabsBean")
   private Whizlabs whizlabs;
   @Resource
   private SessionContext context;
   public void myMethod() {
      Whizlabs whizlabs = (Whizlabs) /* insert-here */;
   }
   // other declarations
}
```

Which of the following expressions can be put into /* insert-here */ to obtain a Whizlabs instance?

Please select :

- A. context.lookup("java:comp/env/whizlabsName");
- B. context.lookup("whizlabsBean");
- C. context.lookup("java:comp/env/whizlabsBean");
- D. context.lookup("java:comp/env/ejb/whizlabsBean");
- E. It depends on the metadata declared on the Whizlabs client view

Your answer is incorrect.

Answer: A

Explanation:

The @EJB annotation shown above binds an instance of the bean named whizlabsBean to whizlabsName in the JNDI naming context of MyBean. As such, this reference can be looked up in the component namespace using the name whizlabsName.

The correct answer is: context.lookup("java:comp/env/whizlabsName");

Submit your Feedback/Queries to our Experts

QUESTION 4 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given an enterprise bean in the com.whizlabs package:

```
@Stateless
public class Whizlabs {
   @EJB
   MyBean myBean;
   // other declarations
}
```

How can the bean referenced by variable myBean be looked up in a local bean client?

Please select :

- A. Context context = new InitialContext();
  MyBean myBean = (MyBean) context.lookup("java:comp/env/com.whizlabs.Whizlabs/myBean");
- B. Context context = new InitialContext();
  MyBean myBean = (MyBean) context.lookup("com.whizlabs.Whizlabs/myBean");
- C. // variable context references a valid EJBContext instance

MyBean myBean = (MyBean) context.lookup("java:com.whizlabs.Whizlabs/myBean");

D. // variable context references a valid EJBContext instance
MyBean myBean = (MyBean) context.lookup("java:comp/env/Whizlabs/myBean");

Your answer is incorrect.

Answer: A

Explanation:

The name element of annotation @EJB in Whizlabs is not specified, implying that the bean is bound to <bean-class-fully-qualified-name>/<variable-name> in the JNDI namespace.

Option C is incorrect as the naming context is wrong (it must be java:comp/env). Option D is also incorrect since the bean class name is unqualified.

When using InitialContext, you must use a full name rather than a relative one. Therefore, option B is incorrect.

The correct answer is: Context context = new InitialContext();
MyBean myBean = (MyBean) context.lookup("java:comp/env/com.whizlabs.Whizlabs/myBean");

Submit your Feedback/Queries to our Experts

QUESTION 5 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Which TWO of the following client views supports asynchronous methods?

Please select :

- A. EJB 2.x local interfaces
- B. EJB 3.x remote interfaces
- C. Web service client views
- D. No-interface views

Your answer is incorrect.

Answer: B and D

Explanation:

As per the Java EE 6 API documentation, asynchronous method invocation semantics only apply to the no-interface, local business, and remote business client views. Methods exposed through the EJB 2.x local, EJB 2.x remote, and web service client views must not be designated as asynchronous.

References:

http://docs.oracle.com/javaee/6/api/javax/ejb/Asynchronous.html

The correct answers are: EJB 3.x remote interfaces, No-interface views

Submit your Feedback/Queries to our Experts

QUESTION 6 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given a stateful session bean that uses a @PostConstruct method to allocate resources. Where should you put clean-up logic code to make sure those resources are ALWAYS released when the bean is discarded?

Please select :

- A. Inside the finalize method
- B. Inside a @Remove method
- C. Inside a @PreDestroy method
- D. This must be done using a separate mechanism

Your answer is incorrect.

Answer: D

Explanation:

Enterprise beans are not allowed to define the finalize method. Option A is thus incorrect in the first place.

A stateful session bean may be discarded without invoking a @Remove method, such as when the bean times out. Hence, option B is incorrect.

Resources are normally cleaned up in a @PreDestroy method. However, even doing so does not guarantee that resources are always released since the bean may be discarded unexpectedly, e.g. when the container crashes or a system exception is thrown.

A separate mechanism should be provided to periodically clean up the unreleased resources.

The correct answer is: This must be done using a separate mechanism

Submit your Feedback/Queries to our Experts

QUESTION 7 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given a business interface and a bean class:

```
public interface Whizlabs {
  @Asynchronous
  public Future<Integer> calculate(List<Integer> integers);
}
@Stateful
@Local(Whizlabs.class)
public class WhizlabsBean {
```

```
  public Integer calculate(List<Integer> integers) {
    int total = 0;
    for (Integer integer : integers) {
      total += integer;
    }
    return total;
  }
  // other declaration
}
```

Which of the following changes must be made to the above declarations to make the WhizlabsBean session bean expose a method with asynchronous client invocation semantics?

Please select :

- A. Declaring Whizlabs as an implemented interface of WhizlabsBean using the implements clause
- B. Moving the @Asynchronous annotation from the interface method to the bean class method
- C. Changing the return type of the interface method from Future<Integer> to Integer
- D. Changing the return type of the bean class method from Integer to Future<Integer>
- E. Nothing needs to be done since the existing bean exposes the calculate method as asynchronous

Your answer is incorrect.

Answer: E

Explanation:

The Whizlabs interface has already been specified in the @Local annotation on the bean class, thus there is no need for the bean class to implement this interface. Option A is incorrect, then.

The @Asynchronous annotation can be declared on the business interface or the bean class, at both type-level and method-level. So, option B is incorrect.

If we change the return type of the interface method to Integer, this method will be invalid as an asynchronous method can only return either void or Future. Hence, option C is incorrect.

The change described in option D is valid, but unnecessary. If a bean class exposes an asynchronous method through a local or remote business interface and the bean method return a type other than Future, the container will wrap this return type into Future when creating the client view.

The correct answer is: Nothing needs to be done since the existing bean exposes the calculate method as asynchronous

Submit your Feedback/Queries to our Experts

QUESTION 8 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Which TWO of the following ways can be used to obtain a SessionContext reference in a session bean?

Please select :

- A. Using dependency injection with the @Resource annotation
- B. Making the bean class implement the SessionBean interface
- C. Using JNDI lookup
- D. Using a constructor of the SessionContext type
- E. Using dependency injection with the @EJB annotation

Your answer is incorrect.

Answer: A and B

Explanation:

According to the EJB 3.1 Specification (subsection 4.3.2), a session bean may use dependency injection mechanisms to acquire references to resources or other objects in its environment. If a session bean makes use of dependency injection, the container injects these references after the bean instance is created, and before any business methods are invoked on the bean instance. If a dependency on the SessionContext is declared, or if the bean class implements the optional SessionBean interface, the SessionContext is also injected at this time. If dependency injection fails, the bean instance is discarded.

Note that SessionContext is not a class, thus there are no constructors for it, while JNDI lookup can be used on a SessionContext instance to get references to some other resources, not the SessionContext itself. Meanwhile, the @EJB annotation cannot be used as a SessionContext instance is not an enterprise bean.

The correct answers are: Using dependency injection with the @Resource annotation, Making the bean class implement the SessionBean interface

Submit your Feedback/Queries to our Experts

QUESTION 9 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given an invalid session bean declaration:

```
@Stateful
public class WhizlabsBean implements SessionSynchronization {
  @AfterBegin
  public void aBegin() { ... }
  @AfterCompletion
  public void aCompletion(boolean arg) { ... }
  @BeforeCompletion
  public void bCompletion() { ... }
  // other declarations
}
```

What change should be made to the above bean declaration to make it valid?

Please select :

- A. Annotate the bean class with @TransactionManagement(BEAN)
- B. Remove the SessionSynchronization interface in the implements clause of the class declaration
- C. Add one method named bBegin to the class and annotate it with @BeforeBegin
- D. Remove the only parameter of the aCompletion method
- E. Rename methods aBegin, aCompletion and bCompletion to afterBegin, afterCompletion and beforeCompletion, respectively

Your answer is incorrect.

Answer: B

Explanation:

As per the EJB 3.1 Specification (subsection 4.3.7), a stateful session bean class may use either the SessionSynchronization interface OR the session synchronization annotations, but not both. Therefore, option B is a correct solution.

References:

http://docs.oracle.com/javaee/6/api/javax/ejb/SessionSynchronization.html

The correct answer is: Remove the SessionSynchronization interface in the implements clause of the class declaration

Submit your Feedback/Queries to our Experts

QUESTION 10 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given a session bean class that implements the SessionSynchronization interface. What can you tell about this bean?

Please select :

- A. It must be a stateless session bean with bean-managed transaction demarcation
- B. It must be a stateful session bean with container-managed transaction demarcation
- C. It must be stateless or singleton session bean with container-managed transaction demarcation
- D. It must be a singleton session bean
- E. It must be a stateful session bean with bean-managed transaction demarcation

Your answer is incorrect.

Answer: B

Explanation:

As per the Java EE 6 API documentation, the SessionSynchronization interface allows a stateful session bean instance to be notified by its container of transaction boundaries.

Only a stateful session bean with container-managed transaction demarcation can receive session synchronization notifications. Other bean types must not implement the SessionSynchronization interface or use the session synchronization annotations.

References:

http://docs.oracle.com/javaee/6/api/javax/ejb/SessionSynchronization.html

The correct answer is: It must be a stateful session bean with container-managed transaction demarcation

Submit your Feedback/Queries to our Experts

QUESTION 11 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given a session bean:

```
@Stateless
public class Whizlabs {
   @Resource
   private SessionContext context;
   @Resource(name = "limit")
   private int maxAmount = 10;
   public void doSomething() {
      int limit = (Integer) context.lookup("limit");
      System.out.println(limit);
   }
   // other declarations
}
```

What must be done to get number 10 printed on the console when the doSomething method is executed?

Please select :

- A. An env-entry element must be specified within the deployment descriptor element that is associated with bean Whizlabs; this element contains two sub-elements: env-entry-name with a value of limit, and env-entry-value with a value of 10
- B. An env-entry element must be specified within the deployment descriptor element that is associated with bean Whizlabs; this element contains sub-element env-entry-name with a value of limit, but does not need to include sub-element env-entry-value
- C. No env-entry element is needed, but the value element of the @Resource annotation must be set to 10
- D. Nothing needs to be done, as the current setting is enough for a successful JNDI lookup

Your answer is incorrect.

Answer: A

Explanation:

According to the EJB 3.1 Specification (subsection 16.4.1.3), if the env-entry-value element is not specified, no value will be injected. The named resource is not initialized in the naming context, and explicit lookups of the named resource will fail. Therefore, no matter what other settings are, an env-entry element with suitable sub-element values must be specified in the deployment descriptor.

The correct answer is: An env-entry element must be specified within the deployment descriptor element that is associated with bean Whizlabs; this element contains two sub-elements: env-entry-name with a value of limit, and env-entry-value with a value of 10

Submit your Feedback/Queries to our Experts

QUESTION 12 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given an EJB injection within the AnotherBean bean class:
@EJB(name = "myName", beanName = "myBean")
MyBean myBean;
Which of the following deployment descriptor elements can be equivalent to the above declaration?

Please select :

A. <ejb-ref>
<ejb-ref-name>myName</ejb-ref-name>
<local>MyBean</local>
<ejb-link>myBean</ejb-link>
<injection-target>
<injection-target-class>AnotherBean</injection-target-class>
<injection-target-name>myBean</injection-target-name>
</injection-target>
</ejb-ref>

B. <ejb-ref>
<ejb-ref-name>myName</ejb-ref-name>
<remote>MyBean</remote>
<ejb-link>myBean</ejb-link>
<injection-target>
<injection-target-class>AnotherBean</injection-target-class>
<injection-target-name>myBean</injection-target-name>
</injection-target>
</ejb-ref>

C. <ejb-ref>
<ejb-ref-name>myBean</ejb-ref-name>
<remote>MyBean</remote>
<ejb-link>myName</ejb-link>
<injection-target>
<injection-target-class>AnotherBean</injection-target-class>
<injection-target-name>myBean</injection-target-name>
</injection-target>
</ejb-ref>

D. <ejb-ref>
<ejb-ref-name>myBean</ejb-ref-name>
<business-interface>MyBean</business-interface>
<ejb-link>myName</ejb-link>
<injection-target>
<injection-target-class>AnotherBean</injection-target-class>
<injection-target-name>myBean</injection-target-name>
</injection-target>
</ejb-ref>

Your answer is incorrect.

Answer: B

Explanation:

The ejb-ref element does not contain the local element, thus option A is incorrect.

Option C is incorrect as the values of ejb-ref-name and ejb-link elements should be the other way around.

Option D is incorrect as the ejb-ref element does not contain the business-interface element.

The correct answer is: <ejb-ref>
<ejb-ref-name>myName</ejb-ref-name>
<remote>MyBean</remote>
<ejb-link>myBean</ejb-link>
<injection-target>
<injection-target-class>AnotherBean</injection-target-class>
<injection-target-name>myBean</injection-target-name>
</injection-target>
</ejb-ref>

Submit your Feedback/Queries to our Experts

QUESTION 13 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given a session bean:
@Stateless

```
public class Whizlabs{
  @Asynchronous
  public void throwException() {
    throw new IllegalStateException();
  }
  // other declarations
}
```

The asynchronous method shown above is invoked by a local client:

```
@Stateless
public class MyBean {
  @EJB
  private Whizlabs whizlabs;
  public invokeWhizlabs() {
    whizlabs.throwException();
    System.out.println("Invocation completed");
  }
  // other declarations
}
```

Which of the following statements is correct when the invokeWhizlabs method is executed and the invocation is then dispatched to the Whizlabs bean?

Please select :

- A. The invokeWhizlabs method throws an IllegalStateException up the call stack
- B. The invokeWhizlabs method throws an EJBException up the call stack
- C. The string "Invocation completed" gets printed
- D. The given declarations are invalid

Your answer is incorrect.

Answer: C

Explanation:

As per the EJB 3.1 Specification (subsection 4.5.5), if an asynchronous method has return type void, then once control has returned from the client's method call no exceptions occurring during the processing of the invocation will be delivered to the client.

The correct answer is: The string "Invocation completed" gets printed

Submit your Feedback/Queries to our Experts

QUESTION 14 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given the producer variable referencing a MessageProducer instance on which a method is invoked:

```
producer.send(message);
```

Which of the following method invocations is equivalent to the above statement?

Please select :

- A. producer.send(message, NON_PERSISTENT)
- B. producer.send(message, NON_PERSISTENT, 0, 0)
- C. producer.send(message, PERSISTENT, 0)
- D. producer.send(message, PERSISTENT, 4, 0)

Your answer is incorrect.

Answer: D

Explanation:

The method in options A and C are not defined in the MessageProducer interface. Among the remaining, only the invocation in option D has arguments matching default values: delivery mode is PERSISTENT, priority level is 4, and time to live is 0.

The correct answer is: producer.send(message, PERSISTENT, 4, 0)

Submit your Feedback/Queries to our Experts

QUESTION 15 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Which of the following statements is correct about a MessageListener instance in a JMS application?

Please select :

- A. It should be registered on a consumer after the associated connection starts
- B. It onMessage method can throw a JMSException
- C. At any time, only one of the session's message listeners is running
- D. None of the above

Your answer is incorrect.

Answer: C

Explanation:

The following are some excerpts from the Oracle's Java EE 6 Tutorial:

- After you register the message listener, you call the start method on the Connection to begin message delivery. If you call start before you register the message listener,

you are likely to miss messages.

- Your onMessage method should handle all exceptions. It must not throw checked exceptions, and throwing a RuntimeException is considered a programming error.

- The session used to create the message consumer serializes the execution of all message listeners registered with the session. At any time, only one of the session's message listeners is running.

References:

http://docs.oracle.com/javaee/6/tutorial/doc/bnceh.html#bnceq

The correct answer is: At any time, only one of the session's message listeners is running

Submit your Feedback/Queries to our Experts

QUESTION 16 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Which TWO of the following are valid injections of a Topic resource into an enterprise bean, given the resource name in the JNDI namespace is jms/topic?

Please select :

A. @Resource(name = "jms/topic")
private Topic topic;

B. @Resource(name = "jms/topic")
public static Topic topic;

C. @Resource(lookup = "jms/topic")
final Topic topic;

D. @Resource(lookup = "jms/topic")
Topic topic;

Your answer is incorrect.

Answer: A and D

Explanation:

As per the EJB 3.1 Specification (subsection 16.2.2), a field or method, which is an injection target, may have any access qualifier (public, private, etc.) but must not be static.

If a field is injected, it must not be final. By default, the name of the field is combined with the name of the class in which the annotation is used and is used directly as the name in the bean's naming context.

Please see subsection 2.3 of the Common Annotations for the Java Platform version 1.1 (JSR 250) for more details on the name and lookup elements of the @Resource annotation.

The correct answers are: @Resource(name = "jms/topic")
private Topic topic;, @Resource(lookup = "jms/topic")
Topic topic;

Submit your Feedback/Queries to our Experts

QUESTION 17 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Which of the following deployment descriptor activation-config elements restrict the associated message-driven bean to receive only messages whose property someProp has the value "someValue"?

Please select :

A. <activation-config>
<activation-config-property>
<activation-config-property-name>messageSelector</activation-config-property-name>
<activation-config-property-value>someProp='someValue'</activation-config-property-value>
</activation-config-property>
</activation-config>

B. <activation-config>
<activation-config-property-name>selector</activation-config-property-name>
<activation-config-property-value>someProp='someValue'</activation-config-property-value>
</activation-config>

C. <activation-config>
<activation-config-property>
<someProp>someValue</someProp>
</activation-config-property>
</activation-config>

D. <activation-config>
<property>
<property-name>beanSelector</property-name>
<property-value>someProp='someValue'</property-value>
</property>
</activation-config>

Your answer is incorrect.

Answer: A

Explanation:

The deployment descriptor element shown in option A demonstrates the correct syntax to apply a JMS messages selector. Please see subsection 5.4.16 of the EJB 3.1 Specification for more details.

The correct answer is: <activation-config>

<activation-config-property>
<activation-config-property-name>messageSelector</activation-config-property-name>
<activation-config-property-value>someProp='someValue'</activation-config-property-value>
</activation-config-property>
</activation-config>

Submit your Feedback/Queries to our Experts

QUESTION 18 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Which of the following is NOT a correct way for a message-driven bean to acquire a MessageDrivenContext instance?

Please select :

- A. Invoke the EJBConfig.getMessageDrivenContext method
- B. Inject a MessageDrivenContext instance using the @Resource annotation
- C. Implement the MessageDrivenBean interface
- D. Specify the context in the resource-env-ref deployment descriptor element

Your answer is incorrect.

Answer: A

Explanation:

The solution in option A is incorrect as there is no EJBConfig type. All other options are valid ways to obtain a MessageDrivenContext instance.

Please see subsection 5.4.3 of the EJB 3.1 Specification for more details.

The correct answer is: Invoke the EJBConfig.getMessageDrivenContext method

Submit your Feedback/Queries to our Experts

QUESTION 19 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Which of the following statements is correct about message-driven beans?

Please select :

- A. Message receipt is acknowledged by the container if a bean uses bean-managed transaction demarcation
- B. Message receipt is automatically acknowledged as part of the transaction commit if the message listener method runs within the context of a transaction
- C. Messages sent to a queue are always processed in sequence
- D. If a timeout method expires when the message listener method is running, a new thread must be generated to handle the timeout event

Your answer is incorrect.

Answer: A

Explanation:

Option B is incorrect as this happens only if the bean uses container-managed transaction demarcation. Note that this (message receipt is automatically acknowledged as part of the transaction commit) is true even if the listener method runs outside a transaction, which occurs when the method is declared with transaction attribute NotSupported. The reason is that the container starts a new transaction before sending a message to the bean. When this transaction is suspended, the listener method runs with no transaction context. After this method returns, the container-started transaction resumes and the acknowledgement becomes part of this transaction.

Option C is incorrect since the container may create multiple bean instances to handle multiple messages at the same time. Option D is incorrect as well since all method invocations on a message-driven bean are serialized. Specifically, the EJB 3.1 Specification (subsection 5.4.10) says that the container serializes calls to each message-driven bean instance. Most containers will support many instances of a message-driven bean executing concurrently; however, each instance sees only a serialized sequence of method calls. Therefore, a message-driven bean does not have to be coded as reentrant.

The correct answer is: Message receipt is acknowledged by the container if a bean uses bean-managed transaction demarcation

Submit your Feedback/Queries to our Experts

QUESTION 20 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Which of the following statements is correct about callback methods of a message-driven bean?

Please select :

- A. A message-driven bean supports only two lifecycle callback methods: @PostConstruct and @PreDestroy
- B. Lifecycle callback methods run within a transaction started by the container
- C. Timeout callback methods never run within a transaction
- D. The @RunAs annotation does not apply to callback methods when declared on the containing bean

Your answer is incorrect.

Answer: A

Explanation:

As per the EJB 3.1 Specification (subsection 5.6.5), PostConstruct and PreDestroy lifecycle callback interceptor methods may be defined for message-driven beans. If PrePassivate or PostActivate lifecycle callbacks are defined, they are ignored. Thus, option A is the correct answer.

Subsection 5.4.12 of the Specification declares that if the bean is specified as using container-managed transaction demarcation, either the REQUIRED, REQUIRES_NEW, or the NOT_SUPPORTED transaction attribute must be used for timeout callback methods. The newInstance method, setMessageDrivenContext, the message-driven bean's dependency injection methods, and lifecycle callback methods are called with an unspecified transaction context. Therefore, options B and C are incorrect.

Here is an extract taken from subsection 5.4.13 of the Specification: The bean provider can use the @RunAs metadata annotation (or corresponding deployment descriptor element) to define a run-as identity for the enterprise bean. The run-as identity applies to the bean's message listener methods and timeout methods. So,

option D is incorrect.

The correct answer is: A message-driven bean supports only two lifecycle callback methods: @PostConstruct and @PreDestroy

Submit your Feedback/Queries to our Experts

QUESTION 21 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Which of the following statements is correct about lifecycle callback interceptor methods?

Please select :

- A. Lifecycle callback interceptor methods may throw either system or application exceptions
- B. A runtime exception thrown by any lifecycle interceptor callback method causes the bean instance and its interceptors to be discarded, except for singleton
- C. A lifecycle callback interceptor must take in an InvocationContext argument
- D. None of the above

Your answer is incorrect.

Answer: B

Explanation:

Lifecycle callback interceptor methods may throw system runtime exceptions, but not application exceptions. Thus, option A is incorrect.

A lifecycle callback interceptor takes in an InvocationContext argument only if it is defined within an interceptor class. It has no parameters if declared within the target class. So, option C is incorrect.

A runtime exception thrown by any lifecycle interceptor callback method causes the bean instance and its interceptors to be discarded after the interceptor chain unwinds. In this case, The @PreDestroy callbacks are not invoked when the bean and the interceptors are discarded and the lifecycle callback interceptor methods in the chain should perform any necessary clean-up operations as the interceptor chain unwinds.

The correct answer is: A runtime exception thrown by any lifecycle interceptor callback method causes the bean instance and its interceptors to be discarded, except for singleton

Submit your Feedback/Queries to our Experts

QUESTION 22 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given a stateless session bean:

```
@Stateless
public class Whizlabs {
   public void output(String text) {
      System.out.println(text);
   }
   @AroundInvoke
   Object intercept(InvocationContext context) throws Exception {
      System.out.print("Intercepting");
      return context.proceed();
   }
   // other declarations
}
```

What happens when a client invokes the Whizlabs.output method with argument "Whizlabs", provided there is no associated deployment descriptor?

Please select :

- A. "Whizlabs" gets printed on the console
- B. "Intercepting" gets printed on the console
- C. "InterceptingWhizlabs" gets printed on the console
- D. The interceptor method is invalid as it cannot return a value while the target method does not

Your answer is incorrect.

Answer: C

Explanation:

When the target method has return type void, the InvocationContext.proceed in the previous method in the interceptor chain returns null. As such, there is nothing wrong with the given declaration.

The intercept method is invoked before the output method is. Thus, "Intercepting" gets printed prior to "Whizlabs".

The correct answer is: "InterceptingWhizlabs" gets printed on the console

Submit your Feedback/Queries to our Experts

QUESTION 23 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Which of the following bean declarations is valid?

Please select :

- A. @Stateless
  @Interceptors(InterceptorA.class)
  @ExcludeClassInterceptors
  public class Whizlabs {
  @Interceptors(InterceptorB.class)
  public void doSomething(String text) { ... }

// other declarations
}

B. @Stateless
@Interceptors(InterceptorA.class, InterceptorB.class)
public class Whizlabs {
public void doSomething(String text) { ... }
// other declarations
}

C. @Stateless
public class Whizlabs {
@Interceptors(InterceptorA.class, InterceptorB.class)
@ExcludeDefaultInterceptors
public void doSomething(String text) { ... }
// other declarations
}

D. @Stateless
@Interceptors(InterceptorA.class)
public class Whizlabs {
@Interceptors(InterceptorB.class)
@ExcludeClassInterceptors
@ExcludeDefaultInterceptors
public void doSomething(String text) { ... }
// other declarations
}

Your answer is incorrect.

Answer: D

Explanation:

The declaration in option A is invalid as the @ExcludeClassInterceptors can only be declared on a method.

The declarations in options B and C are invalid since a list of interceptor classes specified for the @Interceptors annotation must be wrapped within an array: {InterceptorA.class, InterceptorB.class}

The correct answer is: @Stateless
@Interceptors(InterceptorA.class)
public class Whizlabs {
@Interceptors(InterceptorB.class)
@ExcludeClassInterceptors
@ExcludeDefaultInterceptors
public void doSomething(String text) { ... }
// other declarations
}

Submit your Feedback/Queries to our Experts

QUESTION 24	NOT ANSWERED	MARK FOR REVIEW	Ask our Experts

Given an interceptor class:

```
@Interceptor
public class MyInterceptor {
   @AroundInvoke
   Object intercept(InvocationContext context) throws Exception {
      // insert-here
      return context.proceed();
   }
}
```

The above interceptor class is specified within a bean:

```
@Stateless
public class Whizlabs {
   @Interceptors(MyInterceptor.class)
   public void printText(String text) {
      System.out.println(text);
   }
   // other declarations
}
```

Which of the following code fragments should be put at // insert-here to make the Whizlabs.printText method print out "WHIZLABS" when the string "whizlabs" is passed in?

Please select :

A. Object[] parameters = context.getParameters();
String parameter = (String) parameters[0];
parameters[0] = parameter.toUpperCase();
context.setParameters(parameters);

B. Object[] arguments = context.getArguments();
String argument = (String) arguments[0];
arguments[0] = argument.toUpperCase();
context.setArguments(arguments);

C. String parameter = context.getStringParameter();
parameter = parameter.toUpperCase();
context.setStringParameter(parameter);

D. String argument = context.getStringArgument();
argument = argument.toUpperCase();
context.setStringArgument(argument);

Your answer is incorrect.

Answer: A

Explanation:

As per the Java EE 6 API documentation, the setParameters and getParameters methods of the InvocationContext interface sets and gets the parameter values that will be passed to the method of the target class, respectively. If setParameters has been called, getParameters returns the values to which the parameters have been set.

References:

http://docs.oracle.com/javaee/6/api/javax/interceptor/InvocationContext.html

The correct answer is: Object[] parameters = context.getParameters();
String parameter = (String) parameters[0];
parameters[0] = parameter.toUpperCase();
context.setParameters(parameters);

Submit your Feedback/Queries to our Experts

QUESTION 25 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given an enterprise bean:

```
@Stateless
public class Whizlabs {
  public void doSomething() { ... }
  @RolesAllowed("alpha")
  public void doSomething(String arg) { ... }
  // other declarations
}
```

The following is a snippet taken from the EJB deployment descriptor:

```
<method-permission>
  <role-name>beta</role-name>
  <method>
    <ejb-name>Whizlabs</ejb-name>
    <method-name>doSomething</method-name>
  </method>
</method-permission>
```

Which of the following statements is correct about permissions for methods of the Whizlabs bean?

Please select :

- A. Both overloaded doSomething methods are accessible to role beta only
- B. Method doSomething() is accessible to role beta only, while method doSomething(String) is accessible to role alpha only
- C. Method doSomething(String) is accessible to role alpha only, while method doSomething() is accessible to all clients
- D. Specifying a method-name element in the deployment descriptor that matches two method is invalid

Your answer is incorrect.

Answer: A

Explanation:

There is nothing wrong with specifying a method-name element that matches multiple overloaded methods. If this is the case, the permission refers to all these methods. In the given scenario, the security role value beta overrides metadata annotations and applies to both methods.

The correct answer is: Both overloaded doSomething methods are accessible to role beta only

Submit your Feedback/Queries to our Experts

QUESTION 26 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Which of the following is NOT a correct way to prevent a bean method from being called?

Please select :

- A. Decorating the method with the @DenyAll annotation
- B. Specifying the method within the exclude-list element in the deployment descriptor
- C. Specifying the method along with an unchecked element within a method-permission in the deployment descriptor
- D. None of the above

Your answer is incorrect.

Answer: C

Explanation:

When specifying a bean method alongside an unchecked element, such as:

<method-permission>

```
  <unchecked/>
  <method>
    <ejb-name>MyBean</ejb-name>
    <method-name>myMethod</method-name>
  </method>
</method-permission>
```

The opposite is true: The unchecked element indicates that all roles are permitted.

The correct answer is: Specifying the method along with an unchecked element within a method-permission in the deployment descriptor

Submit your Feedback/Queries to our Experts

QUESTION 27 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Which of the following statements is correct about the @RunAs annotation?

Please select :

- A. It can only be specified on an enterprise bean at the class level and apply to all methods of the annotated bean
- B. Once a bean is annotated with @RunAs, the deployment descriptor cannot override the value specified by the annotation
- C. @RunAs specifies identities of the callers of methods of the annotated bean
- D. If the @RunAs annotation and the run-as deployment descriptor element are not specified, the default identity is the container's representation of the unauthenticated identity

Your answer is incorrect.

Answer: A

Explanation:

The @RunAs annotation can only be specified on a type, not a method. It applies to all methods in the annotated bean class, and option A is correct.

The bean provider or application assembler can use the security-identity deployment descriptor element to specify a run-as identity for the execution of the bean's methods, or to override a security identity specified in metadata. So, option B is incorrect.

Establishing a run-as identity for an enterprise bean does not affect the identities of its callers, which are the identities tested for permission to access the methods of the enterprise bean. The run-as identity establishes the identity the enterprise bean will use when it makes calls. Thus, option C is incorrect.

The security principal under which a method invocation is performed is that of the component's caller by default. Thus, option D is incorrect.

The correct answer is: It can only be specified on an enterprise bean at the class level and apply to all methods of the annotated bean

Submit your Feedback/Queries to our Experts

QUESTION 28 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Who can define security roles for an EJB application?

Please select :

- A. Bean provider and application assembler
- B. Application assembler and deployer
- C. Bean provider, application assembler and deployer
- D. Bean provider, application assembler, deployer and system administrator

Your answer is incorrect.

Answer: C

Explanation:

Defining security roles is a job of the bean provider and application assembler using metadata annotations or deployment descriptor. In case the bean provider and application assembler do not do that, the deployer will have to define security roles at deployment time.

The system administrator just map principals in the operational environment to security roles. He/she has nothing to do with the definition of security roles.

The correct answer is: Bean provider, application assembler and deployer

Submit your Feedback/Queries to our Experts

QUESTION 29 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given an enterprise bean:

```
@Stateless
public class Whizlabs {
  public void doSomething() {
    // method body
  }
  public void doSomething(String arg) {
    // method body
  }
  // other declarations
}
```

Which of the following XML fragment can be put into the deployment descriptor to make the doSomething() method accessible to role alpha only, while the doSomething(String) method is accessible to role beta only?

Please select :

A. <method-permission>
<role-name>alpha</role-name>
<method>
<ejb-name>Whizlabs</ejb-name>
<method-name>doSomething</method-name>
<method-params>void</method-params>
</method>
</method-permission>
<method-permission>
<role-name>beta</role-name>
<method>
<ejb-name>Whizlabs</ejb-name>
<method-name>doSomething</method-name>
<method-params>java.lang.String</method-params>
</method>
</method-permission>

B. <method-permission>
<role-name>alpha</role-name>
<method>
<ejb-name>Whizlabs</ejb-name>
<method-name>doSomething</method-name>
<method-params></method-params>
</method>
</method-permission>
<method-permission>
<role-name>beta</role-name>
<method>
<ejb-name>Whizlabs</ejb-name>
<method-name>doSomething</method-name>
<method-params>java.lang.String</method-params>
</method>
</method-permission>

C. <method-permission>
<role-name>alpha</role-name>
<method>
<ejb-name>Whizlabs</ejb-name>
<method-name>doSomething</method-name>
<method-params></method-params>
</method>
</method-permission>
<method-permission>
<role-name>beta</role-name>
<method>
<ejb-name>Whizlabs</ejb-name>
<method-name>doSomething</method-name>
<method-params>
<method-param>java.lang.String</method-param>
</method-params>
</method>
</method-permission>

D. None of the above

Your answer is incorrect.

Answer: C

Explanation:

The method-param values are the fully-qualified Java types of the method's input parameters. If the method has no input arguments, the <method-params> element contains no elements. Arrays are specified by the array element's type, followed by one or more pair of square brackets, such as int[][].

The correct answer is: <method-permission>
<role-name>alpha</role-name>
<method>
<ejb-name>Whizlabs</ejb-name>
<method-name>doSomething</method-name>
<method-params></method-params>
</method>
</method-permission>
<method-permission>
<role-name>beta</role-name>
<method>
<ejb-name>Whizlabs</ejb-name>
<method-name>doSomething</method-name>
<method-params>
<method-param>java.lang.String</method-param>
</method-params>
</method>
</method-permission>

Submit your Feedback/Queries to our Experts

QUESTION 30 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Which type of session bean can be passivated?

Please select :

- A. Singleton only
- B. Stateful only
- C. Stateful and singleton
- D. Stateful, stateless and singleton

Your answer is incorrect.

Answer: B

Explanation:

Passivation and activation apply to only stateful session beans to maintain conversational state across client calls.

The correct answer is: Stateful only

Submit your Feedback/Queries to our Experts

QUESTION 31 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given a bean and a business interface declarations:

```
@Remote
public interface Whizlabs { ... }
@Local(Whizlabs.class)
public class WhizlabsBean implements Whizlabs { ... }
```

Which of the following statements is correct about the Whizlabs interface, provided there is no associated deployment descriptor?

Please select :

- A. It is exposed as a local business interface for WhizlabsBean
- B. It is exposed as a remote business interface for WhizlabsBean
- C. It is exposed as both a local and a remote business interface for WhizlabsBean
- D. The given declarations are invalid

Your answer is incorrect.

Answer: D

Explanation:

Subsection 4.9.8 of the EJB 3.1 Specification notes that it is an error if the @Local and/or @Remote annotations are specified both on the bean class and on the referenced interface and the values differ.

The correct answer is: The given declarations are invalid

Submit your Feedback/Queries to our Experts

QUESTION 32 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given an interface:

```
public interface Whizlabs {
   public void doSomething();
}
```

Which of the following is a correct way to declare a session bean that exposes the above interface as a client view, provided there is no associated deployment descriptor?

Please select :

- A. @Stateless
  public class WhizlabsBean implements Whizlabs {
  public void doSomething() { ... }
  // other declarations
  }
- B. @Stateless
  @Local(WhizlabsBean.class)
  public class WhizlabsBean implements Whizlabs {
  public void doSomething() { ... }
  // other declarations
  }
- C. @Stateless(clientView = Whizlabs.class)
  public class WhizlabsBean implements Whizlabs {
  public void doSomething() { ... }
  // other declarations
  }
- D. @Stateless
  @LocalBean
  public class WhizlabsBean implements Whizlabs {
  public void doSomething() { ... }

```
// other declarations
}
```

Your answer is incorrect.

Answer: A

Explanation:

The declarations in options B and C are invalid, thus these options are not correct.

In order to be exposed as a client view of an enterprise bean, an interface must be declared with the @Local or @Remote annotation, or listed as the only interface in the implements clause of the bean class declaration while the bean does not expose any other client view. Option D is incorrect as the bean in this case exposes a no-interface view.

The correct answer is: @Stateless
public class WhizlabsBean implements Whizlabs {
public void doSomething() { ... }
// other declarations
}

Submit your Feedback/Queries to our Experts

QUESTION 33 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given a stateless session bean:

```
@Stateless
@WebService
public class WhizlabsBean implements Whizlabs {
  public int methodA(int arg) { ... }
  public String methodB(String arg){ ... }
  // other declarations
}
```

Which of the methods shown above are exposed through the web service client view of the bean, provided there are no associated deployment descriptor elements?

Please select :

- A. methodA only
- B. methodB only
- C. Both methodA and methodB
- D. Neither of the methods

Your answer is incorrect.

Answer: C

Explanation:

Normally, a business method that is exposed to web service clients must be annotated with @WebMethod. In the given scenario, however, no methods are annotated with this annotation, implying all public methods are exposed.

The correct answer is: Both methodA and methodB

Submit your Feedback/Queries to our Experts

QUESTION 34 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given a valid stateful session bean with container-managed transaction demarcation:

```
@Stateful
public class Whizlabs {
  public void getConnection() {
    try {
      Context initContext = new InitialContext();
      Context compContext = (Context) initContext.lookup("java:comp/env");
      DataSource dataSource = (DataSource) compContext.lookup("jdbc/myDS");
      Connection connection = dataSource.getConnection();
      // do something
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
  // other declarations
}
```

Which of the following must the getConnection method not be?

Please select :

- A. @PostConstruct method
- B. @PrePassivate method
- C. @AfterCompletion method
- D. @BeforeCompletion method
- E. Business method

F. None of the above

Your answer is incorrect.

Answer: C

Explanation:

As per Table 1 in the @EJB 3.1 Specification (subsection 4.6.1), only lifecycle callback methods, business methods, @AfterBegin and @BeforeCompletion methods have access to resource managers.

The correct answer is: @AfterCompletion method

Submit your Feedback/Queries to our Experts

QUESTION 35 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given an enterprise bean:

```
@Stateless
public class WhizlabsBean {
   public void localA { ... }
   public void localB { ... }
   // other declarations
}
```

And two interfaces:

```
public interface LocalA {
   public void localA();
}
public interface LocalB {
   public void localB();
}
```

Which of the following modifications allows the WhizlabsBean to expose only local business interfaces through LocalA and LocalB?

Please select :

- A. Declare LocalA and LocalB in the implements clause of WhizlabsBean
- B. Annotate WhizlabsBean with @Local({LocalA.class, LocalB.class})
- C. Annotate LocalA and LocalB with @Local
- D. Annotate LocalA and LocalB with @Local and WhizlabsBean with @LocalBean

Your answer is incorrect.

Answer: B

Explanation:

If you declare LocalA and LocalB in the implements clause of WhizlabsBean without using the @Local annotation, LocalA and LocalB are considered normal interfaces and the bean does not expose any client view.

If LocalA and LocalB are annotated as @Local, but not listed in the implements clause of WhizlabsBean, these interfaces are unrelated to the class bean. In this case, WhizlabsBean exposes a no-interface view. Annotating WhizlabsBean with @Local does not make any difference as the value element of the annotation is empty.

Note that a bean does not need to implement an interface to use it as a business interface. Option B is the correct answer, then.

The correct answer is: Annotate WhizlabsBean with @Local({LocalA.class, LocalB.class})

Submit your Feedback/Queries to our Experts

QUESTION 36 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given a method invocation chain among three methods in stateless session beans with container-managed transaction demarcation:
methodA (T1) --> methodB (T2) --> methodC (No)
Which of the following transaction attributes are possible for methodB and methodC, respectively?
Note: The arrow (-->) represents a method invocation; while T1, T2 denotes transaction 1, transaction 2 and No means the associated method runs outside a transaction.

Please select :

- A. RequiresNew and Never

B. Required and Supports

C. RequiresNew and NotSupported

D. Mandatory and Never

Your answer is incorrect.

Answer: C

Explanation:

methodB is executed within a different transaction from that of methodA. Thus, its transaction attribute must be RequiresNew, meaning options B and D are both incorrect.

If the transaction attribute of methodC is Never, the container will throw a RemoteException when this method is invoked from a client running within a transaction. Therefore, option A is incorrect.

References:

http://docs.oracle.com/javaee/6/tutorial/doc/bncij.html#bncik

The correct answer is: RequiresNew and NotSupported

Submit your Feedback/Queries to our Experts

QUESTION 37 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given a stateless session bean with container-managed transaction demarcation. When a method of this bean is invoked, the container throws a RemoteException. Identify the transaction attribute of the invoked method:

Please select :

A. Required

B. Mandatory

C. NotSupported

D. Never

E. None of the above

Your answer is incorrect.

Answer: D

Explanation:

Among all transaction attributes, only two may result in exceptions. Specifically, if the client not associated with a transaction invokes a Mandatory method, the container throws a TransactionRequiredException. If the client running within a transaction invokes a Never method, the container throws a RemoteException.

References:

http://docs.oracle.com/javaee/6/tutorial/doc/bncij.html#bncik

The correct answer is: Never

Submit your Feedback/Queries to our Experts

QUESTION 38 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Which THREE of the following transaction attributes allow the associated methods defined a session bean with container-managed transaction demarcation to invoke the setRollbackOnly and getRollbackOnly methods of the EJBContext interface?

Please select :

A. Required

B. RequiresNew

C. Supports

D. NotSupported

E. Mandatory

F. Never

Your answer is incorrect.

Answer: A, B and E

Explanation:

As per the EJB 3.1 Specification (subsections 13.6.2.8 and 13.6.2.9), the container must handle the EJBContext.setRollbackOnly and EJBContext.getRollbackOnly methods invoked from a business method executing with the REQUIRED, REQUIRES_NEW, or MANDATORY transaction attribute. The container must throw the java.lang.IllegalStateException if those methods (setRollbackOnly and getRollbackOnly) are invoked from a business method executing with the SUPPORTS, NOT_SUPPORTED, or NEVER transaction attribute.

The correct answers are: Required, RequiresNew, Mandatory

Submit your Feedback/Queries to our Experts

QUESTION 39 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given two stateless session beans with container-managed transaction demarcation: BeanA and BeanB. BeanA defines a method named methodA with transaction attribute Required; BeanB has a method named methodB with transaction attribute Mandatory. A local client running within a transaction calls methodA, which in turn calls methodB. When executing, methodB throws an application exception. Which of the following statements is correct, given no exceptions are caught in either methodB or methodA?

Please select :

- A. BeanA instance is discarded
- B. BeanB instance is discarded
- C. The client gets the original application exception and continuing transaction is fruitless
- D. The client gets an EJBException but it may continue within the same transaction
- E. None of the above

Your answer is incorrect.

Answer: E

When an application exception is thrown, it is re-thrown all the way back to the client. Therefore option D is incorrect.

The container discards bean instances only in the case a system exception is thrown, which is not the case over here. So, options A and B are both incorrect.

By default, an application exception does not cause transaction rollback. As the result, option C is incorrect.

The correct answer is: None of the above

Submit your Feedback/Queries to our Experts

QUESTION 40 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given a method in a session bean with bean-managed transaction demarcation:

```
public void doSomething() throws Exception {
  try {
    // do something
    ejbContext.setRollbackOnly();
  } catch (RuntimeException e) {
    e.printStackTrace();
  }
}
```

What happens when the doSomething method is executed?

Please select :

- A. The method throws an exception and the transaction is marked for rollback
- B. The method throws an exception but the transaction is not marked for rollback
- C. The method does not throw an exception, but the transaction is marked for rollback
- D. The method does not throw an exception, and the transaction is not marked for rollback

Your answer is incorrect.

Answer: D

Explanation:

As per the Java documentation, invoking the EJBContext.setRollbackOnly method in a bean with bean-managed transaction demarcation will result in an IllegalStateException. This exception is a subtype of RuntimeException, thus it is caught by the catch block. The setRollbackOnly fails to be completed, the transaction is therefore not marked for rollback.

References:

http://docs.oracle.com/javaee/6/api/javax/ejb/EJBContext.html#setRollbackOnly()

The correct answer is: The method does not throw an exception, and the transaction is not marked for rollback

Submit your Feedback/Queries to our Experts

QUESTION 41 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given three enterprise bean methods involved in container-managed transactions: methodA, methodB and methodC. methodA's transaction attribute is NotSupported. When methodA invokes methodB, methodB is executed outside a transaction. Meanwhile, when methodA invokes methodC, the container throws a TransactionRequiredException. Identify the transaction attributes of methodB and methodC, respectively:

Please select :

- A. NotSupported and Never
- B. Required and Supports
- C. Supports and Mandatory
- D. RequiresNew and NotSupported
- E. None of the above

Your answer is incorrect.

Answer: C

Explanation:

The transaction of methodA is NotSupported, implying that the client of methodB and methodC is not associated with a transaction.

Option A is incorrect since methodC would run outside a transaction. Option B is incorrect since the container would start a new transaction before running methodB and methodC would run outside a transaction. Option D is incorrect since the container would start a new transaction before running methodB, while methodC would run outside a transaction.

You can see option C is the correct answer by looking at the following description of transaction attributes Supports and Mandatory:

- Supports: 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 does not start a new transaction before running the method.

- Mandatory: 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 a TransactionRequiredException.

References:

http://docs.oracle.com/javaee/6/tutorial/doc/bncij.html#bncik

The correct answer is: Supports and Mandatory

Submit your Feedback/Queries to our Experts

QUESTION 42 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given four methods in stateless session beans with container-managed transaction demarcation. These methods are declared with the following transaction attributes:
methodA: RequiresNew
methodB: Mandatory
methodC: NotSupported
methodD: Never
Which of the following invocation chains is correct if the client calling methodA does not run within a transaction?
Note: The arrow (-->) represents a method invocation; while T1, T2 denotes transaction 1, transaction 2 and No means the associated method runs outside a transaction.

Please select :

- A. methodA (T1) --> methodB (T1) --> methodC (T1) --> methodD (No)
- B. methodA (T1) --> methodB (T1) --> methodC (No) --> methodD (No)
- C. methodA (T1) --> methodB (T2) --> methodC (No) --> methodD (No)
- D. methodA (T1) --> methodB (No) --> methodC (T2) --> methodD (T2)

Your answer is incorrect.

Answer: B

Explanation:

The transaction attribute of methodB is Mandatory, implying methodB must execute in the same transaction as methodA. Thus, options C and D are incorrect.

The transaction attribute of methodC is NotSupported, therefore methodC must run without a transaction. So, option A is incorrect as well.

References:

http://docs.oracle.com/javaee/6/tutorial/doc/bncij.html#bncik

The correct answer is: methodA (T1) --> methodB (T1) --> methodC (No) --> methodD (No)

Submit your Feedback/Queries to our Experts

QUESTION 43 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Which of the following APIs may NOT be supported in the Java EE platform (Full Profile)?

Please select :

- A. JAAS
- B. JDBC
- C. JMS
- D. JTA
- E. None of the above

Your answer is incorrect.

Answer: E

Explanation:

All the APIs shown above are supported in the Java EE platform. Among them, JAAS and JDBC are included in the Java SE platform.

References:

http://docs.oracle.com/javaee/6/tutorial/doc/bnacj.html

The correct answer is: None of the above

Submit your Feedback/Queries to our Experts

QUESTION 44 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

When looking up objects in the JNDI component namespace, in which situation a new instance of the requested object rather than an existing one is returned?

Please select :

- A. It always happens
- B. It never happens, as the same instance is returned for all requests for the same object type
- C. It happens only if the container knows the object is immutable
- D. It happens only if the object is defined to be a singleton

E. None of the above

Your answer is incorrect.

Answer: E

As per the EJB 3.1 Specification (subsection 16.2.1), in general, lookups of objects in the JNDI java: namespace are required to return a new instance of the requested object every time. Exceptions are allowed for the following:

- The container knows the object is immutable (for example, objects of type java.lang.String), or knows that the application can't change the state of the object.

- The object is defined to be a singleton, such that only one instance of the object may exist in the JVM.

- The name used for the lookup is defined to return an instance of the object that might be shared. The name java:comp/ORB is such a name.

The correct answer is: None of the above

Submit your Feedback/Queries to our Experts

QUESTION 45 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

What happens when the following code fragment is executed in an enterprise bean method:

```
Context initCtx = new InitialContext();
initCtx.bind("java:comp/env/ejb/Examiner", "Whizlabs1"); // Line 1
initCtx.rebind("java:comp/env/ejb/Examiner", "Whizlabs2"); // Line 2
```

Please select :

A. The String "Whizlabs1" is bound to the name ejb/Examiner

B. The String "Whizlabs2" is bound to the name ejb/Examiner

C. A NamingException is thrown when the statement at Line 1 is executed

D. A NamingException is thrown when the statement at Line 2 is executed

Your answer is incorrect.

Answer: C

Explanation:

As per the EJB 3.1 Specification (subsection 16.3.4), the container must ensure that the enterprise bean instances have only read access to their environment variables. The container must throw the javax.naming.OperationNotSupportedException (a subclass of NamingException) from all the methods of the javax.naming.Context interface that modify the environment naming context and its subcontexts.

The correct answer is: A NamingException is thrown when the statement at Line 1 is executed

Submit your Feedback/Queries to our Experts

QUESTION 46 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Which of the following is NOT a standard initialization property for an embeddable container?

Please select :

A. javax.ejb.embeddable.beans

B. javax.ejb.embeddable.initial

C. javax.ejb.embeddable.modules

D. javax.ejb.embeddable.appName

Your answer is incorrect.

Answer: A

Explanation:

All embeddable containers must recognize three initialization properties, namely javax.ejb.embeddable.initial, javax.ejb.embeddable.modules and javax.ejb.embeddable.appName. javax.ejb.embeddable.initial.

Property javax.ejb.embeddable.initial holds a String value that specifies the fully-qualified name of an embeddable container provider class corresponding to the embeddable container implementation that should be used.

Property javax.ejb.embeddable.modules is used to explicitly specify the module(s) to be initialized. It can refer to modules that are included in the JVM classpath or to modules outside the JVM classpath.

Property javax.ejb.embeddable.appName specifies an application name for the EJB modules executing within the embeddable container. If specified, the property value applies to the <app-name> portion of the portable global JNDI name syntax. If this property is not specified, the <app-name> portion of the portable global JNDI name syntax does not apply.

The correct answer is: javax.ejb.embeddable.beans

Submit your Feedback/Queries to our Experts

QUESTION 47 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Which of the following statements is correct about metadata annotations and the deployment descriptor?

Please select :

A. The deployment descriptor must always be present

B. The deployment descriptor may be absent; when it is present, metadata annotations is ignored

C. The deployment descriptor can override some of the metadata annotations

D. The deployment descriptor can override all metadata annotations

Your answer is incorrect.

Answer: C

Explanation:

There are two basic kinds of metadata information: enterprise beans' structural information and application assembly information. The structural information cannot, in general, be changed because doing so could break the enterprise bean's function. On the other hand, assembly level information can be changed without breaking the enterprise bean's function. As such, only some, not all, metadata annotations can be overridden by the deployment descriptor.

The correct answer is: The deployment descriptor can override some of the metadata annotations

Submit your Feedback/Queries to our Experts

QUESTION 48 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given a session bean named MyBean with a single client view: a no-interface view. The class file of this bean is packaged within an EJB-JAR file named myBean.jar.
Which of the following code fragments can be used to obtain a reference to MyBean within an embeddable container?

Please select :

A. Context context = new Context();
MyBean myBean = (MyBean) context.lookup("java:global/myBean/MyBean");

B. Context context = new InitialContext();
MyBean myBean = (MyBean) context.lookup("java:myBean/MyBean");

C. Context context = ejbContainer.getContext();
MyBean myBean = (MyBean) context.lookup("java:global/myBean.jar/MyBean");

D. Context context = ejbContainer.newContext();
MyBean myBean = (MyBean) context.lookup("java:global/myBean/MyBean");

E. None of the above

Your answer is incorrect.

Answer: E

Explanation:

Within an embeddable container, the naming context must be retrieved using the EJBContainer.getContext method. Therefore, options A, B and D are incorrect in the first place.

Session bean references are identified using the portable global JNDI name syntax, which is:

java:global[/application name]/module name/enterprise bean name[/interface name]

In the above syntax, module name is the unqualified name of an archive file excluding the extension part. Thus, option C is incorrect as well.

The correct statements to obtain a reference to MyBean is as follows:

Context context = ejbContainer.getContext();

MyBean myBean = (MyBean) context.lookup("java:global/myBean/MyBean");

References:

http://docs.oracle.com/javaee/6/tutorial/doc/gkcrr.html#glhur

The correct answer is: None of the above

Submit your Feedback/Queries to our Experts

QUESTION 49 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given the declaration of a method:

```
public void invokeBeanMethod() {
  try {
    // do something
  } catch (EJBException e) {
    e.printStackTrace();
  }
}
```

The above method calls a local business interface business method of a stateless session bean with container-managed transaction demarcation from within the try block. This invocation is made in the context of a transaction, and the invoked bean method is declared with transaction attribute Mandatory. After the calling, the client prints an exception and its backtrace to the error stream. Which of the following statements is NOT correct?

Please select :

A. The transaction has been marked for rollback

B. The enterprise bean has been destroyed

C. The exception has been logged in the container

D. None of the above

Your answer is incorrect.

Answer: D

Explanation:

The exception is caught by the given catch block, meaning that it is a system exception. Operations described in options A, B and C are what the container must do when

facing a system exception.

The correct answer is: None of the above

Submit your Feedback/Queries to our Experts

QUESTION 50 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given a scenario where the message listener method of a message-driven bean with bean-managed transaction demarcation throws a system exception. Which of the following is NOT a responsibility of the container?

Please select :

- A. Log the exception
- B. Rollback the current transaction, if any
- C. Discard the bean instance
- D. Throw the original exception

Your answer is incorrect.

Answer: D

With the given scenario, the container will log the exception, rollback the current transaction (if any), discard the bean instance, and throw an EJBException that wraps the original exception to resource adapter.

Please see subsection 14.3.4 of the EJB 3.1 Specification for more details.

The correct answer is: Throw the original exception

Submit your Feedback/Queries to our Experts

QUESTION 51 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

What does a container do if it fails to start or commit a container-managed transaction?

Please select :

- A. Retry the failed transaction
- B. Throw an EJBException
- C. Throw a RemoteException
- D. Throw either an EJBException or a RemoteException

Your answer is incorrect.

Answer: D

Explanation:

As per the EJB 3.1 Specification (subsection 14.3.10), if the container fails to start or commit a container-managed transaction, the container must throw the javax.ejb.EJBException. If the business interface is a remote business interface that extends java.rmi.Remote, the java.rmi.RemoteException is thrown to the client instead.

The correct answer is: Throw either an EJBException or a RemoteException

Submit your Feedback/Queries to our Experts

QUESTION 52 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given a session bean with container-managed transaction demarcation. This bean contains a method that invokes another bean method. In which of the following case the calling bean method recognizes that the transaction has been marked for rollback?

Please select :

- A. It receives an EJBException
- B. It receives a RemoteException
- C. The invocation of method EJBContext.getRollbackOnly returns true
- D. The invocation of method UserTransaction.getStatus returns

Your answer is incorrect.

Answer: C

Explanation:

Only EJBTransactionRolledbackException, TransactionRolledbackException or TransactionRolledbackLocalException implies that the transaction has been rolled back or marked for rollback. All other exceptions do not tell anything about the rollback status of a transaction. Thus, options A and B are incorrect.

The UserTransaction.getStatus method can only be invoked on a bean with bean-managed transaction demarcation. Option D is incorrect, then.

The correct answer is: The invocation of method EJBContext.getRollbackOnly returns true

Submit your Feedback/Queries to our Experts

QUESTION 53 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Which of the following statements is correct about application exception?

Please select :

- A. An application exception must be a checked exception

- B. Any checked exception is considered an application exception if it is listed in the throws clause of a bean business method
- C. An application exception never results in the transaction being marked for rollback
- D. None of the above

Your answer is incorrect.

Answer: D

Explanation:

Any RuntimeException subclass can be declared as an application exception by using the @ApplicationException. Thus option A is incorrect.

The RemoteException and its subclasses are system exceptions, even when listed in the throws clause of a business method. Hence, option B is incorrect.

An application exception may result in marking the transaction for rollback if it is decorated with @ApplicationException(rollback = true). Therefore, option C is incorrect.

The correct answer is: None of the above

Submit your Feedback/Queries to our Experts

QUESTION 54 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given a singleton session bean:

```
@Singleton
public class Whizlabs {
   private int number = 0;
   public void increment() {
      number++;
      number++;
   }
   public void printOut() {
      System.out.println(number);
   }
   // other declarations
}
```

What number can be printed on the console when the above bean is accessed by two clients at the same time, one invokes increment and the other calls printOut?

Please select :

- A. 0
- B. 2
- C. 0 or 2
- D. 1 or 2
- E. 0, 1, or 2

Your answer is incorrect.

Answer: C

Explanation:

By default, a business method of a singleton bean is associated with a Write lock. Therefore, the number field cannot be accessed while the increment method is running. As a result, either 0 or 2 will be printed, depending on which method is executed first.

The correct answer is: 0 or 2

Submit your Feedback/Queries to our Experts

QUESTION 55 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given a singleton session bean:

```
@Singleton
@Local(A.class)
@Remote(B.class)
public class Whizlabs implements A, B {
   private int number;
   private String text;
   public void methodA() { ... }
   public void methodB() { ... }
   // other declarations
}
```

The above bean exposes many business interface client views. What is the correct way to make the bean thread-safe, provided there is no associated deployment descriptor?

Please select :

- A. Annotate the bean class with @ConcurrencyManagement(BEAN)
- B. Annotate the bean class with @Lock(READ)
- C. Specify all methods as synchronized
- D. Specify all fields as volatile
- E. Nothing needs to be done

Your answer is incorrect.

Answer: E

Explanation:

By default, singleton session beans use container-managed concurrency with an exclusive lock for every business method. This default configuration ensures thread-safety for bean instances.

The solutions in options C and D are unnecessary, while the ones in options A and B makes the bean no longer thread-safe.

The correct answer is: Nothing needs to be done

Submit your Feedback/Queries to our Experts

QUESTION 56 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given two singleton session beans:

```
@Singleton
@Startup
public class BeanA {
   @EJB
   BeanB myBean;
   @PostConstruct
   void initialize() {
      myBean.doSomething();
   }
   // other declarations
}
@Singleton
public class BeanB {
   public void doSomething() { ... }
   // other declarations
}
```

What needs to be done to ensure a BeanB instance is always available when the initialize method of BeanA is executed, provided there is no associated deployment descriptor?

Please select :

- A. Annotate BeanA with @DependsOn("BeanB")
- B. Annotate BeanB with @DependsOn("BeanA")
- C. The goal cannot be guaranteed as BeanA is initialized during the application startup, while BeanB may not
- D. The existing declarations already ensure the availability of a BeanB instance for the initialization of BeanA

Your answer is incorrect.

Answer: D

Explanation:

As per the EJB 3.1 Specification (subsection 4.8.1), the container ensures that all Singleton beans with which a Singleton has a DependsOn relationship have been initialized before PostConstruct is called. If one Singleton merely needs to invoke another Singleton from its PostConstruct method, no explicit ordering metadata is needed. In that case, the Singleton would merely use an ejb reference to invoke the target Singleton.

The correct answer is: The existing declarations already ensure the availability of a BeanB instance for the initialization of BeanA

Submit your Feedback/Queries to our Experts

QUESTION 57 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Which of the following declarations of a singleton session bean instructs the container to print "Welcome to Whizlabs" when the application is deployed?

Please select :

- A. @Stateless

```
@Startup
public class Whizlabs {
@PostConstruct
public void initialize() {
System.out.println("Welcome to Whizlabs");
}
}
```

- B. @Singleton

```
@Startup
public class Whizlabs {
@PostConstruct
private void initialize() {
System.out.println("Welcome to Whizlabs");
}
}
```

- C. @Singleton

```
@Startup(method = "initialize")
public class Whizlabs {
public void initialize() {
```

```
System.out.println("Welcome to Whizlabs");
}
}
```

D. None of the above

Your answer is incorrect.

Answer: B

Explanation:

Option A is incorrect as the @Startup annotation does not apply to stateless session beans. It can be declared on singleton beans only.

Option C is incorrect as there is no element named method for the @Startup annotation.

Note that a @PostConstruct method may be assigned any access modifier, including private. Therefore, there is nothing wrong with the declaration in option B.

References:

http://docs.oracle.com/javaee/6/api/javax/ejb/Startup.html

The correct answer is: @Singleton

```
@Startup
public class Whizlabs {
@PostConstruct
private void initialize() {
System.out.println("Welcome to Whizlabs");
}
}
```

Submit your Feedback/Queries to our Experts

QUESTION 58 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Given a scenario where you want the container to invoke a method that send a notification email to all employees at 8 a.m. every weekday. Which of the following statements is correct?

Please select :

- A. This should be done with programmatic timer creation
- B. This should be done with automatic timer creation
- C. There is no positive or negative effect when using either programmatic or automatic timer creation
- D. The goal cannot be achieved using a timer service

Your answer is incorrect.

Answer: B

Explanation:

When using programmatic timers, the method that creates the timer must be executed for the timer to be available. If for some reason this method is not invoked, the timer will not be created and the timeout callback will never run. The programmatic method is, therefore, only suitable for timers that must be created based on a certain condition.

On the other hand, automatic timers are created upon application deployment. As a result, these timers are always in effect until they expire or are explicitly canceled. Automatic timers are thus more reliable than programmatic timers.

The correct answer is: This should be done with automatic timer creation

Submit your Feedback/Queries to our Experts

QUESTION 59 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Which of the following statements is NOT correct about the TimerHandle interface?

Please select :

- A. TimerHandle is only available for persistent timers
- B. TimerHandle must not be passed through a bean's remote business interface
- C. A TimerHandle instance is serializable
- D. None of the above

Your answer is incorrect.

Answer: D

As per the EJB 3.1 Specification (subsection 18.2.6), the TimerHandle interface allows the bean provider to obtain a serializable timer handle that may be persisted. Timer handles are only available for persistent timers. Since timers are local objects, a TimerHandle must not be passed through a bean's remote business interface, remote interface or web service interface.

In addition, the TimerHandle interface is a subinterface of Serializable, thus TimerHandle instances are serializable.

References:

http://docs.oracle.com/javaee/6/api/javax/ejb/TimerHandle.html

The correct answer is: None of the above

Submit your Feedback/Queries to our Experts

QUESTION 60 | NOT ANSWERED | MARK FOR REVIEW | Ask our Experts

Which of the following statements can be used to cancel all active timers associated with a TimerService instance referenced by variable timerService?

Please select :

- A. timerService.cancelTimers();
- B. timerService.revokeAll();
- C. Map<Integer, Timer> timerMap = timerService.getTimers();
  for(Timer timer : timerMap.values()) {
  timer.cancel();
  }
- D. Collection<Timer> timers = timerService.getTimers();
  for(Timer timer : timers) {
  timer.cancel();
  }

Your answer is incorrect.

Answer: D

Explanation:

The TimerService interface does not define cancelTimers or revokeAll method. Thus, options A and B are both incorrect.

The TimerService.getTimers returns a Collection, not a Map. Therefore, option C is incorrect.

The correct answer is: Collection<Timer> timers = timerService.getTimers();
for(Timer timer : timers) {
timer.cancel();
}

Submit your Feedback/Queries to our Experts

Finish review