

SECTION: 1: 1: EJB 3.0 Overview

OBJECTIVE: 1.1: Identify the uses, benefits, and characteristics of Enterprise JavaBeans technology, for version 3.0 of the EJB specification.

1) Which statement correctly identifies the goals of the designers of the EJB architecture?

- a) The EJB architecture requires that users be aware of the low-level transaction details of the Java EE platform.
- b) The EJB architecture is the standard component architecture for building distributed business applications in the Java programming language. (*)
- c) Although the EJB architecture defines the contracts that enable tools to develop and deploy components, each application server vendor must produce its own set of additional tooling.
- d) The EJB architecture provides a standard way of persisting state data using the Java Persistence API, Java Data Objects, and JDBC.

REFERENCE:

Option B is correct. See section 2.1 "Overall Goals" in the Core Specification. None of the other options are listed as goals here.

Option A is incorrect because the specification makes it clear that these low-level details are not required.

Option C is incorrect because you do not need tooling from a particular application server vendor.

Option D is incorrect because the EJB architecture only standardizes on the Java Persistence API.

SECTION: 11: 11: Security Management

OBJECTIVE: 11.1: Match security behaviors to declarative security specifications (default behavior, security roles, security role references, and method permissions)

2) Given an excerpt from a stateless session bean:

```
10. @Stateless
11. @RolesAllowed("JAMES")
12. public class YourBean implements MyI {
13.     @DenyAll
14.     public void spy() { /* ... */}
15.     public void eye() { /* ... */}
14. }
```

The methods spy and eye are defined in the business interface for this bean. The bean has an accompanying descriptor, and here is an excerpt of the relevant security portion:

```
<method-permission>
<role-name>BOND</role-name>
<method>
<ejb-name>YourBean</ejb-name>
<method-name>eye</method-name>
</method>
```

Which security roles can access which methods?

- a) JAMES can access `spy` and `eye`.
- b) BOND can access `spy` and `eye`.
- c) JAMES and BOND can access `spy` and `eye`.
- d) JAMES can access `eye`, and BOND can access `spy`.
- e) JAMES can access `eye`, no role can access `spy`.
- f) BOND can access `eye`, no role can access `spy`. (*)

REFERENCE:

Option F is correct. The `RolesAllowed` annotation tells you that the security role JAMES can access the bean (all methods) (See 17.3.2.1 of the Core Specification), while `DenyAll` retracts that permission for the `spy` method (so nobody can access it) The deployment descriptor overrides the annotation (See 17.3.2.2 of the Core Specification) and so only BOND can access `eye`, NOT JAMES.

Options A,B,C,D,E are incorrect because F is correct. See the reasoning for Option F.

SECTION: 2: 2: General EJB 3.0 Enterprise Bean Knowledge

OBJECTIVE: 2.1: Identify correct and incorrect statements or examples about the life cycle of all 3.0 enterprise bean instances, including the use of the `@PostConstruct` and `@PreDestroy` callback methods.

3) Which statement about life-cycle callback methods is correct?

- a) Life-cycle callback methods must be implemented in the bean class.
- b) Life-cycle callback methods can have public, private, protected, or package-level access. (*)
- c) A life-cycle callback method can only have a single callback annotation. In other words, you cannot define a single method and give it two different callback annotations.
- d) Life-cycle callback methods can be declared as static.

REFERENCE:

See 3.4.1 Simplified Specification for all of these.

Option A is incorrect. They can be implemented in an interceptor class as well.

Option B is correct.

Option C is incorrect. It can have multiple callback annotations.

Option D is incorrect. They cannot be static (or final)

SECTION: 3: 3: EJB 3.0 Session Bean Component Contract and Life Cycle

OBJECTIVE: 3.1: Identify correct and incorrect statements or examples that compare the purpose and use of stateful and stateless session beans.

4) Which statement characterizes stateless session beans?

- a) They allow the `PostConstruct`, `PreDestroy`, and `PrePassivate` life-cycle callbacks. (*)
- b) They require home interfaces.
- c) When a client looks up a stateful session bean in the JNDI, the same bean is returned every time.
- d) They are asynchronous message consumers.

REFERENCE:

Option A is correct. See Specification 5.1.4.

Option B is incorrect. Session beans do not require home interfaces. (See Simplified Specification 3.5.)

Option C is incorrect. See 4.4 of the Core Specification. A different bean is returned every time.

Option D is incorrect. Message-driven beans are asynchronous message consumers. (See 2.4.2 of the Core Specification.)

SECTION: 4: 4: EJB 3.0 Message-Driven Bean Component Contract

OBJECTIVE: 4.1: Develop code that implements a message-driven bean class.

5) A message-driven bean can have an instance of a `MessageDrivenContext` injected. Which method can be successfully invoked on this interface from a message-driven bean?

- a) `getEJBHome`
- b) `getCallerPrincipal (*)`
- c) `isCallerInRole`
- d) `getEJBLocalHome`

REFERENCE:

Option B is correct. See 5.4.4 of the Core Specification.

Options A, C, and D should NOT be called. See 5.4.4 of the Core Specification.

SECTION: 5: 5: Java Persistence API Entities

OBJECTIVE: 5.1: Identify correct and incorrect statements or examples about the characteristics of Java Persistence entities.

6) Which statement about entities is correct?

- a) Entities must be annotated with the `@Entity` annotation.
- b) Entities can be final classes.
- c) Entities can have a single no-arg constructor with protected visibility. (*)
- d) Instance variables of an entity can have private, protected, public, or package visibility.

REFERENCE:

Option A is incorrect. The developer can use the XML descriptor instead. (See 2.1 of the JPA Specification.)

Option B is incorrect. They cannot be final. (See 2.1 of the JPA Specification.)

Option C is correct. That no-arg constructor can be protected, it does NOT have to be public. (See 2.1 of the JPA Specification.)

Option D is incorrect. They must NOT be public. (See 2.1 of the JPA Specification.)

SECTION: 6: 6: Java Persistence Entity Operations

OBJECTIVE: 6.1: Describe how to manage entities, including using the `EntityManager` API and the cascade option.

7) Which statement is correct about the `EntityManager` API?

- a) The `merge`, `persist`, `remove`, and `getReference` methods must be invoked within a transaction context.
- b) It is safe (no exception is thrown) to call `merge` or `getTransaction` on a JTA `EntityManager` instance.
- c) The `getReference` method can throw an `EntityNotFoundException`. (*)
- d) Runtime exceptions thrown by the `refresh` and `createQuery` methods of the `EntityManager` interface do NOT cause the transaction to be rolled back.

REFERENCE:

Option A is incorrect. `getReference` does NOT need to be invoked within a transaction context. (See JPA Specification 3.1.1.)

Option B is incorrect. An exception is thrown if you call `getTransaction` on a JTA transaction manager. (See the JPA Specification 3.1.1 interface listing.)

Option C is correct. Even though in general this behaves as a lazy proxy to an entity, an `EntityNotFoundException` can be thrown. (See the JPA Specification 3.1.1 interface documentation.)

Option D is incorrect. It causes the exception to roll back. (See JPA Specification 3.1.1.)

SECTION: 7: 7: Persistence Units and Persistence Contexts

OBJECTIVE: 7.1: Identify correct and incorrect statements or examples about JTA and resource-local entity managers.

8) Which statement about JTA and resource-local entity managers is correct?

- a) The default transaction type for an entity manager is JTA in both Java EE and Java SE environments.
- b) You cannot use a resource-local entity manager in a Java EE environment.
- c) The `EntityTransaction` interface must be used when using a resource-local entity manager. (*)
- d) Application-managed entity managers can be only a JTA transaction type.

REFERENCE:

Option A is incorrect. The default is resource-local for Java SE. (See 6.2.1.2 of the JPA Specification.)

Option B is incorrect. You can. (See 6.2.1.2 of the JPA Specification for an example.)

Option C is correct. The developer needs to use this to explicitly start/commit the transactions. (See 5.5.2 and 5.5.2.1 of the JPA Specification.)

Option D is incorrect. They can be resource-local as well. (See 5.5 of the JPA Specification.)

SECTION: 8: 8: Java Persistence Query Language

OBJECTIVE: 8.1: Develop queries that use the `SELECT` clause to determine query results, including the use of entity types, use of aggregates, and returning multiple values.

9) Given an excerpt from an entity:

```
10. @Entity
11. public class Koala {
12.     int children;
13.     @Id
14.     private Integer id;
15.     /* ... */
16. }
```

The following code was written to find the sum and average number of children across all Koala entities (assume the variable `em` is bound to a valid `EntityManager` instance):

```
20. String query = "SELECT SUM(k.children), AVG(k.children) from Koala k";
21. Query q = em.createQuery(query);
22. Object[] res = (Object []) q.getSingleResult();
```

Which two statements are correct? (Choose two.)

- a) There is a syntax error in the JPQL on line 20.
- b) If the variable `o` is NOT null, the types of the two values in the array are `Integer` and `Double` respectively. (*)
- c) There is an error in the typecasting on line 22, which results in a runtime or compile-time exception.
- d) If the variable `o` is NOT null, the types of the two values in the array are `Double` and `Double` respectively.
- e) The syntax of the JPQL statement on line 20 is valid and the code executes without an error. (*)
- f) A runtime exception is generated on line 22.

REFERENCE:

Option E and Option B are correct. This is a valid JPQL query. (See 4.8.4 of the JPA Specification.) The Type of the return value is `(Object[])` (see 3.6.1 of the JPA Specification) and in this case, the `MAX` aggregate function returns `Integer` and `AVG` a `Double` (See 4.8.4 of the JPA Specification.)

Option A, C, D and F are incorrect because there is no syntax error in the query or the code.

SECTION: 9: 9: Transactions

OBJECTIVE: 9.1: Identify correct and incorrect statements or examples about bean-managed transaction demarcation.

10) Which statement about transaction demarcation is correct?

- a) A session bean can be designed with bean-managed and container-managed transaction demarcation, at the same time.
- b) A message-driven bean can be designed with bean-managed (BM) and container-managed (CM) transaction demarcation, at the same time.
- c) Entities can be designed with bean-managed or container-managed transaction demarcation, but not with both at the same time.
- d) Both stateless and stateful session beans can designed with bean-managed transactions. . (*)
- e) Only stateless session beans can be designed with bean-managed transactions.

REFERENCE:

Option D is correct. (See 13.3.1 of the Core Specification.)

Option A and B are incorrect because the developer cannot use both BM and CM transaction demarcation at the same time.

Option C is incorrect because D is correct. The developer cannot specify the management type.

Option E is incorrect. The developer can have bean-managed transaction with stateful beans as well. (See 13.3.1 of the Core Specification.)