

838p Final Exam

Don't panic.

There are 12 questions. **Answer any 10 of them (i.e., pick two questions to not answer)**. Each question is worth 10 points.

1. Discuss pass-by-value vs. pass-by-reference, with regards to:
 - Web services
 - EJB Local interfaces
 - EJB Remote interfaces

Answer:

- Web services: All web services are implemented with a fairly simple pass-by-value approach.
 - EJB Local interfaces use the same mechanism as normal Java. Objects are passed by reference, primitive values are passed by value.
 - EJB Remote interfaces use the same mechanism as RMI. Primitive values are passed by value, Serializable or externalizable objects are passed by value (e.g., copy), and remote objects are passed by reference.
2. How would you decide whether something should be implemented as a stateful session bean or as an entity bean. Give several rules for deciding this (e.g., if it is colored blue, it should be an entity bean).

Answer: Both entity beans and stateful session beans have state that persists across more than a single method call. However, stateful session beans are unique to a unique session, while entity beans typically persist across many sessions and are accessed by many sessions.

If something might be created by a mechanism other than an EJB call, if it needs to be accessed by multiple sessions, or corresponds to an entry in the persistent store, it should be an entity bean.

If something shouldn't be shared across multiple session, stateful session beans are a possibility.

A shopping cart could be implemented as either. As an entity bean, it would be possible for a user to put things into their shopping cart, move to a different machines, and then update their shopping cart and complete their purchase. On the other hand, you might want to do it as a session bean, and avoid some of the complications and data storage issues associated with storing the shopping cart in the database. For example, the information collected from a sequence of web pages to complete the purchase of a plane ticket would more likely be handled with a stateful session bean.

3. Here are some methods. State which interfaces they might reasonably appear in (e.g., [remote/local] [home/component] interface of a [stateless session bean / stateful session bean / entity bean]). Some (most) methods may appear in multiple interfaces, please list concisely using wildcards as appropriate. Note: the methods don't list return types or thrown exceptions.

- create()

Answer: Local/Remote home interface

- create(String firstName, String lastName)

Answer: Local/Remote home interface for entity bean or stateful session bean

- findByPrimaryKey(String key) **Answer:** Local/Remote home interface

- getHandle()

Answer: Remote component interface

- remove()

Answer: Local/Remote component interface

- remove(Handle h)

Answer: Remote home interface

4. Suppose you had two different entity beans, X and Y. There are three different relationships between X and Y. Relationship R1 is a one-to-one bidirectional relationship from X to Y. Relationship R2 is a one-to-many relationship from X to Y. Relationship R3 is a many-to-many relationship from X to Y. Describe how these relationships could be implemented in an underlying database.

Answer: Although R1 could be implemented using a separate table, most likely it would be implemented by having the table for X contain a field that is the primary key for the corresponding element of Y, and the table for Y contain a field that is the primary key for the corresponding element of X. Although you could accomplish this with just one of these fields, but doing so would make computation of the reverse direction expensive.

R2 could also be implemented using a separate table, most likely it would be implemented by having the table for Y contain a field that contains the primary key for the corresponding element of X.

R3 would need to be implemented with a separate join table. This R3 table would contain two fields, Xkey and Ykey, containing primary keys for the corresponding elements of X and Y.

5. What are the mechanisms that a servlet container might use to track sessions? What requirements do they impose on browsers and/or servlets?

Answer: Servlets containers can track sessions using two different mechanisms. One is to use cookies, the other is to perform URL rewriting. Using cookies requires that users have cookies enabled in their browser. Using URL rewriting requires that all

URLs to pages that are to be part of the session be rewritten by the servlet container. Getting URL rewriting to work requires care, and means that even static pages need to be dynamically generated by the servlet container.

6. In Servlets/JSP, what is the difference between forwarding, including and redirecting?

Answer:

- forwarding: the response is generated by the page forwarded to, and the current servlet is not further consulted.
- including: the response generated by the included page is inserted into the response generated by the current servlet/jsp.
- redirecting: the client is told to load a different url. This requires an extra round trip with the client, and also means that the new URL will be visible to the user, and can be bookmarked or reloaded.

7. What special functionality/convenience is made available by using JavaBeans in a JSP or servlet?

Answer: There are several, including:

- The ability to set multiple properties of a bean from request parameters with a single method call.
- JSP tags that allow access to bean properties without writing Java code.

8. Given a container managed, 1-1 bidirectional relation:

`Foo <--> Bar`

and the object relations:

`f1 <--> b1`

`f2 <--> b2`

Which boolean expressions will be true after the following code runs?

```
b2.setFoo(b1.getFoo());
```

Note all that apply:

- (a) `f1.getBar() == null`
- (b) `f2.getBar() == null`
- (c) `b1.getFoo() == null`
- (d) `b2.getFoo() == null`

Answer: b, c.

9. What situations/circumstances would force you to use bean-managed transactions? Contained-managed transactions?

Answer: The primary reason for using bean-managed transactions is if you don't want the duration of a transaction to match the duration of a method call. You might want a transaction to be shorter than a method call, or even stretch over several method calls (all this is very dubious).

The primary reason for using contained managed transactions is if you want a method to be performed in the transaction context of the calling method. Guiding by this principle, entity bean methods are restricted to contained managed transactions.

10. What is the difference between a Get request and a Post request in HTTP/Servlets?

Answer: In a get request, all of the parameters are encoded in the URL. In a post request, parameters are sent separately, after the URL. This allows for arbitrarily long/big parameters in a post request, and prevents post requests from being bookmarked. Generally, post requests are used for substantial form inputs and things that cause actions to be performed (i.e., buying your plane tickets). Get requests are generally restricted to situations where performing duplicate requests would not be harmful. The servlet interface has separate methods for doGet and doPost, allowing you to handle them separately.

11. Which of the following design patterns is normally used to reduce the number of remote calls made by the clients to the server? Select the one that best applies.

- (a) Business Delegate
- (b) Model View Controller
- (c) Facade
- (d) Factory
- (e) Value Objects

12. Compare and contrast the following. In particular, give situations that would call out for or preclude the use of these technologies.

- JDBC ResultSets
- JDBC RowSets
- JDO
- EJB Entity beans

: Answer:

- JDBC ResultSets: represents a result returned from a database, and may be a live connection. ResultSets cannot be serialized or passed between machines. Some ResultSets have very limited functionality.

- JDBC RowSets: A general framework for representing tabular data. There are several different forms, but generally they offer more guaranteed functionality than ResultSets (such as scrolling), and can be serialized and passed between machines.
- JDO: Provides Object-like access to persistent data (e.g., each persistent entity is an object, each attribute of the entity has getter/setter methods). Can be used outside of EJB, in normal J2SE or J2ME applications. Doesn't directly integrate with J2EE functionality such as transactions and security. Unclear how well JDO handles persistent stores too large to fit in memory.
- EJB Entity beans: Provides Object-like access to persistent data (e.g., each persistent entity is an object, each attribute of the entity has getter/setter methods). Works well with very large persistent stores (only part of which are cached in memory at any time) and integrates well with application servers (e.g., integration of transaction and security support).