« Prev Page (https://www.sanfoundry.com/spring-mcqs-questions-answers/)
Next Page (https://www.sanfoundry.com/spring-questions-answers-setting-transaction-attribute/) »

Spring Questions and Answers – Transaction Management

This set of Java Spring Multiple Choice Questions & Answers (MCQs) focuses on “Transaction Management”.

1. Transactions can be described with key properties:-
a) Atomicity
b) Consistency
c) Isolation
d) All of the mentioned
View Answer

Answer: d
Explanation: The concept of transactions can be described with four key properties: atomicity, consistency, isolation, and durability (ACID).
• Atomicity: A transaction is an atomic operation that consists of a series of actions. The atomicity of a transaction ensures that the actions either complete entirely or take no effect at all.
• Consistency: Once all actions of a transaction have completed, the transaction is committed. Then your data and resources will be in a consistent state that conforms to business rules.
• Isolation: Because there may be many transactions processing with the same data set at the same time, each transaction should be isolated from others to prevent

data corruption.
• Durability: Once a transaction has completed, its result should be durable to survive any system failure (imagine if the power to your machine was cut right in the middle of a transaction commit). Usually, the result of a transaction is written to persistent storage.

2. To access a database running on the Derby server, you have to add:-
a) Derby client library
b) Tomcat client library
c) All of the mentioned
d) None of the mentioned
View Answer

Answer: a
Explanation: To access a database running on the Derby server, you have to the Derby client library to your CLASSPATH.

3. Spring's transaction support offers a set of technology-independent facilities, including transaction managers.
a) org.springframework.transaction.PlatformTransactionManager
b) org.springframework.transaction.support.TransactionTemplate
c) all of the mentioned
d) none of the mentioned
View Answer

Answer: c
Explanation: Spring's transaction support offers a set of technology-independent facilities, a transaction template (e.g., org.springframework.transaction.support.TransactionTemplate), and transaction declaration support to simplify your transaction management tasks.

4. Spring's core transaction management abstraction is based on the interface:-
a) PlatformTransaction
b) PlatformTransactionManager
c) TransactionManager
d) PlatformManager
View Answer

Answer: b
Explanation: It encapsulates a set of technology-independent methods for transaction management.

5. The PlatformTransactionManager interface provides methods for working with transactions:
a) getTransaction(TransactionDefinition definition)
b) commit(TransactionStatus status)
c) rollback(TransactionStatus status)
d) all of the mentioned
View Answer

Answer: d
Explanation:

• TransactionStatus getTransaction(TransactionDefinition definition) throws TransactionException
• void commit(TransactionStatus status) throws TransactionException;
• void rollback(TransactionStatus status) throws TransactionException;

6. Spring has several built-in implementations of PlatformTransactionManager interface for use with different transaction management APIs.

a) True

b) False

View Answer

Answer: a

Explanation:

• If you have to deal with only a single data source in your application and access it with JDBC, DataSourceTransactionManager should meet your needs.
• If you are using JTA for transaction management on a Java EE application server, you should use JtaTransactionManager to look up a transaction from the application server. Additionally, JtaTransactionManager is appropriate for distributed transactions (transactions that span multiple resources). Note that while it's common to use a JTA transaction manager to integrate the application servers' transaction manager, there's nothing stopping you from using a stand-alone JTA transaction manager such as Atomikos.
• If you are using an object/relational mapping framework to access a database, you should choose a corresponding transaction manager for this framework, such as HibernateTransactionManager and JpaTransactionManager.

7. A transaction manager is declared in the Spring IoC container as a normal bean.

a) True

b) False

View Answer

Answer: a

Explanation: For example, the following bean configuration declares a DataSourceTransactionManager instance. It requires the dataSource property to be set so that it can manage transactions for connections made by this data source.

8. Method that allows you to start a new transaction (or obtain the currently active transaction).

a) getTransaction()

b) commit()

c) rollback()

d) all of the mentioned

View Answer

Answer: a

Explanation: Spring's transaction manager provides a technology-independent API that allows you to start a new transaction (or obtain the currently active transaction) by calling the getTransaction() method.

9. PlatformTransactionManager is an abstract unit for transaction management.

a) True

b) False

View Answer

Answer: a

Explanation: Because PlatformTransactionManager is an abstract unit for transaction management, the methods you called for transaction management are guaranteed to be technology independent.

10. Method to start a new transaction with that definition:-

a) getTransaction()

b) commit()

c) rollback()

d) none of the mentioned

View Answer

Answer: a

Explanation: Once you have a transaction definition, you can ask the transaction manager to start a new transaction with that definition by calling the getTransaction() method.

11. To help you control the overall transaction management process and transaction exception handling.

a) SpringTransactionTemplate

b) TransactionTemplate

c) Transaction

d) None of the mentioned

View Answer

Answer: b

Explanation: Spring also provides a TransactionTemplate to help you control the overall transaction management process and transaction exception handling.

12. You just have to encapsulate your code block in a callback class that implements the TransactionCallback interface and pass it to the TransactionTemplate execute method for execution. In this way, you don’t need to repeat the boilerplate transaction management code for this block.

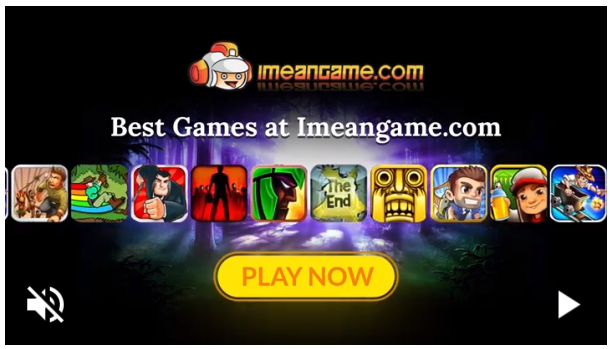
a) True

b) False

View Answer

Answer: a

Explanation: In this way, you don’t need to repeat the boilerplate transaction management code for this block.



13. A TransactionTemplate can accept a transaction callback object that implements:-
a) TransactionCallback
b) TransactionCallbackWithoutResult class
c) All of the mentioned
d) None of the mentioned
View Answer

Answer: c
Explanation: A TransactionTemplate can accept a transaction callback object that implements either the TransactionCallback or an instance of the one implementer of that interface provided by the framework, the TransactionCallbackWithoutResult class.

14. Spring (since version 2.0) offers a transaction advice that can be easily configured via the:-
a) rx:advice
b) bx:advice
c) tx:advice
d) none of the mentioned
View Answer

Answer: c
Explanation: This advice can be enabled with the AOP configuration facilities defined in the aop saop schema.

15. You can omit the transaction-manager attribute in the element if your transaction manager has the name transactionManager.
a) True
b) False
View Answer

Answer: a
Explanation: This element will automatically detect a transaction manager with this name. You have to specify a transaction manager only when it has a different name.

Sanfoundry Global Education & Learning Series – Java Spring.
To practice all areas of Java Spring, here is complete set of 1000+ Multiple Choice Questions and Answers (https://www.sanfoundry.com/1000-spring-questions-answers/).