**Spring Data Repositories**

Question 1

What are the key JPA Classes for interactions with a database?

The key JPA Classes for interactions with a database include the EntityManager and Entity classes. The EntityManager is the interface that lets us interact with the persistence context, which is a set of entity instances. An Entity is a lightweight persistence domain object. Typically, an entity represents a table in a relational database, and each entity instance corresponds to a row in that table.

Correct

EntityManagerFactory and EntityManager control all interactions trough their API to the DataSource

Question 2

Where do you configure a DataSource in a Spring Boot application?

You configure a DataSource in a Spring Boot application in the application.properties file. Here, you can specify properties such as the URL, username, and password for your database. Alternatively, you can also configure a DataSource programmatically by defining a DataSource factory method within a class annotated with the @Configuration annotation.

Correct

application.properties or you can define a DataSource in a Java Config file i.e. class annotated with @Configuration

Question 3

What do we mean by an extended Persistence Context?

An extended Persistence Context in JPA can span across multiple transactions. We can persist the entity without the transaction but cannot flush it without a transaction. This means that entities remain managed over multiple transactions, and changes to entities are not immediately synchronized with the database unless within a transaction.

Correct

An EntityManager Context injected with @PersistenceContext is transactionally scoped. In order to “live” beyond the commit or rollback of a transaction you have to create the EntityManager from the EntityManager yourself, and close it yourself.

Question 1

What are the key JPA Classes for interactions with a database?

The key JPA Classes for interactions with a database include the EntityManager and Entity classes. The EntityManager is the interface that lets us interact with the persistence context, which is a set of entity instances. An Entity is a lightweight persistence domain object. Typically, an entity represents a table in a relational database, and each entity instance corresponds to a row in that table.

Correct

EntityManagerFactory and EntityManager control all interactions trough their API to the DataSource

Question 2

Where do you configure a DataSource in a Spring Boot application?

You configure a DataSource in a Spring Boot application in the application.properties file. Here, you can specify properties such as the URL, username, and password for your database. Alternatively, you can also configure a DataSource programmatically by defining a DataSource factory method within a class annotated with the @Configuration annotation.

Correct

application.properties or you can define a DataSource in a Java Config file i.e. class annotated with @Configuration

Question 3

What do we mean by an extended Persistence Context?

An extended Persistence Context in JPA can span across multiple transactions. We can persist the entity without the transaction but cannot flush it without a transaction. This means that entities remain managed over multiple transactions, and changes to entities are not immediately synchronized with the database unless within a transaction.

Correct

An EntityManager Context injected with @PersistenceContext is transactionally scoped. In order to “live” beyond the commit or rollback of a transaction you have to create the EntityManager from the EntityManager yourself, and close it yourself.

Question 1

What is a JoinPoint?

A JoinPoint in Spring AOP represents a point during the execution of a program, such as the execution of a method or the handling of an exception. In Spring AOP, a JoinPoint always represents a method execution.

Correct

Describes an entry point into the application such as a method invocation. It is tied to a PointCut which gets associated with one or many Advices

Question 2

What do we mean by being proxied?

Being proxied in Spring AOP means that an object is wrapped by another object (the proxy) that maintains its interface and optionally provides additional features. Proxies usually delegate behavior to the real object they are proxying but can execute code around the call to the wrapped object. Spring AOP uses either JDK dynamic proxies or CGLIB to create the proxies for your target objects.

Correct

A proxy wraps the target bean to control access to it. IT has the same API as the target bean and therefore can act as a surrogate to take the place of the target bean. It may block requests or control access through to the target bean and is our basis for Spring AOP.

Question 3

Which advice only executes upon returning normally from a target bean method invocation?

The advice that only executes upon returning normally from a target bean method invocation is known as @AfterReturning advice. This type of advice is triggered only after a matched method’s execution completes normally, without throwing an exception.

Correct

@AfterReturning will only be triggered on normal return from the target bean, @After always executes even if the target bean method threw an Exception. It also provides a means to get hold of the returned object form the target bean.

Question 1

Why do we prefer to use Declarative Transactional Support and how is it implemented?

Declarative Transactional Support is preferred because it provides a consistent programming model across different transaction APIs such as Java Transaction API (JTA), JDBC, Hibernate, Java Persistence API (JPA), and Java Data Objects (JDO). It supports declarative transaction management and offers a simpler API for programmatic transaction management. It is implemented using Spring’s aspect-oriented programming (AOP), but the transactional aspects code comes with the Spring Framework distribution and can be used in a boilerplate fashion.

Correct

Declarative Transaction management is non evasive to your code, it begins and commits or rollbacks the transaction, without the need for coding logic. A method becomes operational in a transaction by annotating it with @Transactional

Question 2

What is transaction propagation?

Transaction propagation defines the boundaries of a business logic’s transaction. Spring manages to start and pause a transaction according to the propagation setting. It determines whether a method call should be run within a transaction context or not.

Correct

This will scope the boundaries of a transaction when it starts and when it commits or rollbacks. It can also state it requires to be suspended because of a new transaction has started and resumed once this second transaction has completed.

Question 3

What types of exception does Spring automatically rollback a transaction?

In its default configuration, Spring automatically rolls back a transaction in the case of runtime, unchecked exceptions. That is, when the thrown exception is an instance or subclass of RuntimeExceptionError instances also result in a rollback by default.

Correct

Runtime exceptions, for Checked exceptions you must express a rollBackFor attribute with an array of CheckedExceptions that you wish to trigger rollback for.