1. What is JPA?

As a specification, the Java Persistence API is concerned with persistence, which loosely means any mechanism by which Java objects outlive the application process that created them. Not all Java objects need to be persisted, but most applications persist key business objects. The JPA specification lets you define which objects should be persisted, and how those objects should be persisted in your Java applications. By itself, JPA is not a tool or framework; rather, it defines a set of concepts that can be implemented by any tool or framework. While JPA's object-relational mapping (ORM) model was originally based on Hibernate, it has since evolved. Likewise, while JPA was originally intended for use with relational/SQL databases, some JPA implementations have been extended for use with NoSQL datastores. A popular framework that supports JPA with NoSQL is EclipseLink, the reference implementation for JPA 2.2.

2. What is the naming convention for finder methods in the Spring data repository interface?

Spring data JPA has its own naming conventions for methods. Following these conventions, we can build sophisticated queries. These conventions are called also called as method name strategies. These strategies have defined set of keywords to use in method names. Based on the formed method name, method performs predefined operations. As a simple example: findByLastnameAndFirstname, findByFirstname, findByFirstnameIs, findByFirstnameEquals.

3. What is PagingAndSortingRepository?

PagingAndSortingRepository is an extension of CrudRepository to provide additional methods to retrieve entities using the pagination and sorting abstraction. It provides two methods:

Page findAll (Pageable pageable) – returns a Page of entities meeting the paging restriction provided in the Pageable object.

Iterable findAll (Sort sort) – returns all entities sorted by the given options. No paging is applied here.

4. Differentiate between findById() and getOne()?

Both findById() and getOne() methods are used to retrieve an object from underlying datastore. But the underlying mechanism for retrieving records is different for both these methods, infact getOne() is lazy operation which does not even hit the database. getOne() returns a reference to the entity with the given identifier. getOne internally invokes EntityManager.getReference() method. As per docs, this method will always return a proxy without hitting the database (lazily fetched). This method will throw EntityNotFoundException at the time of actual access if the requested entity does not exist in the database. findById() method will hit the database and return the real object mapping to a row in the database. It is EAGER loaded operation that returns null if no record exists in database.

5. What is @Query used for?

In order to define SQL to execute for a Spring Data repository method, we can annotate the method with the @Query annotation — its value attribute contains the JPQL or SQL to execute. The @Query annotation takes precedence over named queries, which are annotated with @NamedQuery or defined in an orm.xml file. It's a good approach to place a query definition just above the method inside the repository rather than inside our domain model as named queries. The repository is responsible for persistence, so it's a better place to store these definitions.

6. What is lazy loading in hibernate?

Lazy loading can help improve the performance significantly since often you won't need the children and so they will not be loaded. Say you have a parent, and that parent has a collection of children. Hibernate now can "lazy-load" the children, which means that it does not actually load all the children when loading the parent. Instead, it loads them when requested to do so. You can either request this explicitly or, and this is far more common, hibernate will load them automatically when you try to access a child.

7. What is SQL injection attack? Is Hibernate open to SQL injection attack?

SQL injection is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. It generally allows an attacker to view data that they are not normally able to retrieve. This might include data belonging to other users, or any other data that the application itself is able to access. In many cases, an attacker can modify or delete this data, causing persistent changes to the application's content or behavior. In some situations, an attacker can escalate an SQL injection attack to compromise the underlying server or other back-end infrastructure, or perform a denial-of-service attack. Hibernate does not grant immunity to SQL Injection, one can misuse the API as they please. There is nothing special about HQL (Hibernates subset of SQL) that makes it any more or less susceptible. Functions such as createQuery(String query) and createSQLQuery(String query) create a Query object that will be executed when the call to commit() is made. If the query string is tainted, you have SQL injection. The details of these functions are covered later.

8. What is criteria API in hibernate?

Hibernate provides alternate ways of manipulating objects and in turn data available in RDBMS tables. One of the methods is Criteria API, which allows you to build up a criteria query object programmatically where you can apply filtration rules and logical conditions. The Hibernate Session interface provides createCriteria() method, which can be used to create a Criteria object that returns instances of the persistence object's class when your application executes a criteria query. You can use add() method available for Criteria object to add restriction for a criteria query. You can create AND or OR conditions using LogicalExpression restrictions. The Criteria API provides the org.hibernate.criterion.Order class to sort your result set in either ascending or descending order, according to one of your object's properties. The Criteria API provides the org.hibernate.criterion.Projections class, which can be used to get average, maximum, or minimum of the property values. The Projections class is similar to the Restrictions class, in that it provides several static factory methods for obtaining Projection instances.

9. What Is Erlang? Why Is It Required For Rabbitmq?

Erlang is a programming language used to build massively scalable soft real-time systems with requirements on high availability. Some of its uses are in telecoms, banking, e-commerce, computer telephony and instant messaging. Erlang's runtime system has built-in support for concurrency, distribution, and fault tolerance. Written in Erlang, the RabbitMQ server is built on the Open Telecom Platform framework for clustering and failover. Client libraries to interface with the broker are available for all major programming languages. The source code is released under the Mozilla Public License.

10. What is the JPQL?

JPQL is Java Persistence Query Language defined in JPA specification. It is used to create queries against entities to store in a relational database. JPQL is developed based on SQL syntax. But it won't affect the database directly. JPQL can retrieve information or data using SELECT clause, can do bulk updates using UPDATE clause and DELETE clause. EntityManager.createQuery() API will support for querying language.

11. What are the steps to persist an entity object?

1) Creating an entity manager factory object
2) Obtaining an entity manager from factory.
3) Initializing an entity manager.
4) Persisting a data into relational database.
5) Closing the transaction
6) Releasing the factory resources.

12. What are the different types of entity mapping?

one-to-one, one-to-many or many-to-one (dependent on the direction), many-to-many. In addition, each relationship can be one-way or two-way. This is referred to as the direction of the relationship. The one-way relationship is unidirectional; the two-way relationship is bidirectional. For example, a unidirectional relationship can be from an employee to an address. With the employee information, you can retrieve an address. However, with an address, you cannot retrieve the employee. An example of a bidirectional relationship is with an employee/projects example. Given a project number, you can retrieve the employees working on the project. Given an employee number, you can retrieve all projects that the employee is working on. Thus, the relationship is valid in both directions.

13. What are the properties of an entity?

Various properties can be specified inside your application.properties file, inside your application.yml file, or as command line switches. This appendix provides a list of common Spring Boot properties and references to the underlying classes that consume them. Property contributions can come from additional jar files on your classpath, so you should not consider this an exhaustive list. Also, you can define your own properties.

14. Difference between CrudRepository and JpaRepository in Spring Data JPA?

JpaRepository extends PagingAndSortingRepository which in turn extends CrudRepository. Their main functions are CrudRepository mainly provides CRUD functions. PagingAndSortingRepository provides methods to do pagination and sorting records. JpaRepository provides some JPA-related methods such as flushing the persistence context and deleting records in a batch. Because of the inheritance mentioned above, JpaRepository will have all the functions of CrudRepository and PagingAndSortingRepository. So, if you don't need the repository to have the functions provided by JpaRepository and PagingAndSortingRepository, use CrudRepository.