# What is JPA ?

JPA is a standard that emerged in the Java programming language for associating Java classes with relational database tables. The purpose here is to match the class variables with the columns of the table and to perform database operations directly on the objects without writing SQL. At this point, JPA only sets a standard and does not take any action on the data itself. For this, a separate tool that implements these standards is needed. Hibernate, TopLink, EclipseLink and OpenJPA can be counted as the main tools that implement JPA.

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

Spring uses lower snake case by default, which means it uses only lower case letters and separates words with underscores. For example *findByStudentId().*

# What is PagingAndSortingRepository ?

The PagingAndSortingRepository interface provided by Spring Data Commons is used.

*public interface PagingAndSortingRepository<T, ID> extends CrudRepository<T, ID> {*

*Iterable<T> findAll(Sort sort);*

*Page<T> findAll(Pageable pageable);*

*}*

Methods counted by the Sort class are used in the sorting process.

Below is an example usage of the sorting feature.

*personRepository.findAll(Sort.by("id"))*

*.forEach(System.out::println);*

*personRepository.findAll(Sort.by(Sort.Direction.DESC, "firstName"))*

*.forEach(System.out::println);*

*personRepository.findAll(Sort.by("firstName").descending())*

*.forEach(System.out::println);*

*personRepository.findAll(Sort.by("firstName", "LastName"))*

*.forEach(System.out::println);*

*personRepository.findAll(Sort.by("firstName").and(Sort.by("lastName")))*

*.forEach(System.out::println);*

# Differentiate between findById() and getOne() ?

**getOne()**

* Brings the reference of the target with Lazy Load
* Very useful when you don't need to access object's properties
* When the object is accessed, it throws EntityNotFoundException if the object is not in the DB
* better performance

**findById()**

* Really returns the given id (not a reference)
* All its properties are accessible as the object is fully fetch
* Returns null if the object is not registered in the DB
* An additional DB query is required

# What is @Query used for ?

It is used to create queries over objects created from classes called Entity that are mapped to database tables.

# What is lazy loading in hibernate ?

In Hibernate, the relation between objects differs in pulling the data in the database.

Let's say we have a child entity. Let the child entity be the associated toy entity and the relationship between them be OneToMany. So a child can have more than one toy. When we call the child from the database, we should do Lazy Loading here if we want him to bring the toys he has, not just him. Here, we say that only he should come, and I will call his toys when necessary. When necessary, we call the toys using the "get" method of the object.

# What is SQL injection attack ? Is Hibernate open to SQL injection attack ?

A type of cyber attack called SQL Injection or SQLi is a hacking technique that makes it possible to bypass web page and web application security measures. Using this attack technique, hackers can hijack, modify and delete the contents of an SQL database.

* 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.

*String query1 = "select \* from MyBean where id = "+ id;//Not secure*

*String query2 = "select \* from MyBean where id = :id";//Secure*

# What is criteria API in hibernate ?

The Criteria API can be used to get the lowest, highest, and average values with the *org.hibernate.criterion.Projections* class. The Projections class is similar to the Restrictions class. It can be used to fetch the fields we want while fetching records in SQL.

# What Is Erlang? Why Is It Required For Rabbitmq ?

Erlang is a concurent and functional programming language designed for programming large industrial and real-time systems.

Erlang is dynamically typed and has a pattern matching syntax.

Functions are used and defined recursively in this language. Erlang provides clear concurrency, and is a language free of side effects.

Erlang has It is an Open Source message queue system. In other words, it is the system that receives a message from one application and forwards it to another application when it is their turn. We can think of this system as a post office or cargo company. Just as a letter is delivered to the post office or a product is delivered to the relevant address when the time comes, RabbitMQ performs the same process and sends the message given to it to the relevant consumer at the right time

If we want to provide a scalable environment in software applications, RabbitMQ should be used. We can explain this as follows; Let's say we have a website or mobile application. If we cannot respond instantly as a result of a request from the user, or if we need to activate non-instantaneous/time-consuming processes, we should process such a process asynchronously and reduce the application density instead of distracting the user. Otherwise, the user will be exposed to an unnecessary Response Time and there will be a situation against our software.

The structure that will control the asynchronous process here is RabbitMQ. RabbitMQ provides a mechanism that will allow a different application to take the responsibility here by detaching the operations that may take a long time in Response Time from the application. This mechanism; will send long/costly transactions to the queue via RabbitMQ, and the transactions in this queue will be processed by a different application, and the result will be obtained asynchronously independently of the main application, thus minimizing the density in the main application as much as possible. a very powerful distributed architecture.

# What is the JPQL ?

JPQL (Java Persistence Query Language) is a language defined by the JPA standard for querying Entity objects. JPQL is very similar to HQL (Hibernate Query Language). Although these languages are almost similar to SQL (Structured Query Language), the arguments they use are Entity objects instead of database tables.

# What are the steps to persist an entity object ?

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

# What are the different types of entity mapping ?

**One-To-One Relationship Overview**

A one-to-one relationship is the simplest relationship between two beans. One entity bean relates only to one other entity bean. If our company office contains only cubicles, and only a single employee can sit in each cubicle, then you have a one-to-one relationship: one employee in one designated cubicle. You define a unidirectional definition for this relationship as follows:

employee --> cubicle

However, if you have a cubicle number and want to determine who is assigned to it, you can assign a bidirectional relationship. This would enable you to retrieve the employee and find what cubicle he/she sits in. In addition, you could retrieve the cubicle number and determine who sits there. You define this bidirectional one-to-one relationship as follows:

employee <--> cubicle

**One-To-Many or Many-To-One Relationship Overview**

In a one-to-many relationship, one object can reference several instances of another. A many-to-one relationship is when many objects reference a single object. For example, an employee can have multiple addresses: a home address and an office address. If you define these relationships as unidirectional from the perspective of the employee, then you can look up the employee and see all of his/her addresses, but you cannot look up an address to see who lives there. However, if you define this relationship as bidirectional, then you can look up any address and see who lives there.

**Many-To-Many Relationship Overview**

A many-to-many relationship is complex. For example, each employee can be working on several projects. And each projects has multiple employees working on it. Thus, you have a many-to-many cardinality. The direction does not matter in this instance. You have the following cardinality:

employees <--> projects

# What are the properties of an entity ?

Persistable: Must be saveable and accessible. (public no-arg constructor, getter/setter methods)

Idetity : Each Entity must have a unique identity (can be expressed as a Primary Key)

Transactionality: Adding / deleting / updating the database is a transaction. If the process is not fully completed, rollback should be done and it should be returned to the beginning.

# Difference between CrudRepository and JpaRepository in Spring Data JPA?

|  |  |
| --- | --- |
| **CrudRepository** | **JpaRepository** |
| It is a base interface and extends Repository Interface. | It extends PagingAndSortingRepository that extends CrudRepository. |
| It contains methods for CRUD operations. For example save(), saveAll(), findById(), findAll(), etc. | It contains the full API of CrudRepository and PagingAndSortingRepository. For example, it contains flush(), saveAndFlush(), saveAllAndFlush(), deleteInBatch(), etc along with the methods that are available in CrudRepository. |
| It doesn’t provide methods for implementing pagination and sorting | It provides all the methods for which are useful for implementing pagination. |
| It works as a marker interface. | It extends both CrudRepository and PagingAndSortingRepository. |
| To perform CRUD operations, define repository extending CrudRepository. | To perform CRUD as well as batch operations, define repository extends JpaRepository. |
| **Syntax:**  public interface CrudRepository<T, ID> extends Repository<T, ID> | **Syntax:**  public interface JpaRepository<T,ID> extends PagingAndSortingRepository<T,ID>, QueryByExampleExecutor<T> |