HW#4 – İbrahim Çelik

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.

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

JPA finder methods are the most powerful methods, we can create finder methods to select the records from the database without writing SQL queries. Behind the scenes, Data JPA will create SQL queries based on the finder method and execute the query for us.

To create finder methods in Data JPA, we need to follow a certain naming convention. To create finder methods for the entity class field name, we need to create a method starting with findBy followed by field name.

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() ?

getOne()	**findById()**
Lazily loaded reference to target entity	Actually loads the entity for the given id
Useful only when access to properties of object is not required	Object is eagerly loaded so all attributes can be accessed
Throws EntityNotFoundException if actual object does not exist at the time of access invocation	Returns null if actual object corresponding to given Id does not exist

getOne()	**findById()**
Better performance	An additional round-trip to database is required

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 is a fetching technique used for all the entities in Hibernate. It decides whether to load a child class object while loading the parent class object. When we use association mapping in Hibernate, it is required to define the fetching technique. The main purpose of lazy loading is to fetch the needed objects from the database.

For example, we have a parent class, and that parent has a collection of child classes. Now, Hibernate can use lazy loading, which means it will load only the required classes, not all classes. It prevents a huge load since the entity is loaded only once when necessary. Lazy loading improves performance by avoiding unnecessary computation and reduce memory requirements.

Lazy loading can be used with all types of Hibernate mapping, i.e., one-to-one, one-to-many, many-to-one, and many-to-many.

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

SQL injection vulnerabilities arise when you construct database queries unsafely, and untrusted data gets interpreted as a part of the SQL query structure.

The best way to prevent SQL injection vulnerabilities is to use a framework that allows you to construct and parameterize queries safely. An ORM (Object Relational Mapper) is a good option. For additional security layers, validate all input and use a WAF (Web Application Firewall) product.

Hibernate does not grant immunity to SQL Injection, one can misuse the api as they please.

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.

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

Erlang is a concurrent programming language designed for programming fault-tolerant distributed systems at Ericsson and has been (since 2000) freely available subject to an open-source license.

Erlang is used for programming fault-tolerant, distributed, real-time applications. What differentiates it from most other languages is that it's a concurrent programming language; concurrency belongs to the language, not to the operating system. Its programs are collections of parallel processes cooperating to solve a particular problem that can be created quickly and have only limited memory overhead; programmers can create large numbers of Erlang processes yet ignore any preconceived ideas they might have about limiting the number of processes in their solutions.

Erlang is a general-purpose programming language and runtime environment for which RabbitMQ is built upon.

If you're doing erlang to erlang communication, you don't need RabbitMQ.

If you need robustness in the form of crash resilience, knowing whether or not a job was started or completed, being able to let a back-end process go down and not worry too much about losing work, then need RabbitMQ.

10 – What is the JPQL ?

JPQL is a powerful query language that allows you to define database queries based on your entity model. Its structure and syntax are very similar to SQL. But there is an important difference that I want to point out before I walk you through the different parts of a JPQL query. JPQL uses the entity object model instead of database tables to define a query.

That makes it very comfortable for us Java developers, but you have to keep in mind that the database still uses SQL. Hibernate, or any other JPA implementation, has to transform the JPQL query into SQL. It is, therefore, a good practice to activate the logging of the SQL statements during development to check the generated SQL statements.

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

- Creating an entity manager factory object
- Obtaining an entity manager from factory.
- Intializing an entity manager.
- Persisting a data into relational database.
- Closing the transaction.
- Releasing the factory resources.

12 – What are the different types of entity mapping ?

Hibernate simplifies data handling between SQL and JDBC by mapping the Object Oriented model in Java with the Relational model in Databases. Although mapping of basic Java classes is in-built in Hibernate, mapping of custom types is often complex.

Hibernate uses mapping types for converting Java objects into SQL queries for storing data. Similarly, it uses mapping types for converting SQL ResultSet into Java objects while retrieving data.

Generally, Hibernate categorizes the types into Entity Types and Value Types. Specifically, Entity types are used to map domain specific Java entities and hence, exist independently of other types in the application. In contrast, Value Types are used to map data objects instead and are almost always owned by the Entities.

In this tutorial, we will focus on the mapping of Value types which are further classified into:

- Basic Types – Mapping for basic Java types
- Embeddable – Mapping for composite java types/POJO's
- Collections – Mapping for a collection of basic and composite java type

13 - What are the properties of an entity ?

These are the properties of an entity that an object must have: -

- **Persistability -** An object is called persistent if it is stored in the database and can be accessed anytime.
- **Persistent Identity -** In Java, each entity is unique and represents as an object identity. Similarly, when the object identity is stored in a database then it is represented as persistence identity. This object identity is equivalent to primary key in database.
- **Transactionality -** Entity can perform various operations such as create, delete, update. Each operation makes some changes in the database. It ensures that whatever changes made in the database either be succeed or failed atomically.
- **Granuality -** Entities should not be primitives, primitive wrappers or built-in objects with single dimensional state.

14 - 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.

CrudRepository	**JpaRepository**
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>