Java Persistence API (JPA) in Spring Boot

1. Introduction to JPA

JPA (Java Persistence API) is a specification that provides a way to manage relational data in Java applications. It simplifies database interactions using Object-Relational Mapping (ORM) techniques.

JPA vs Hibernate

- **JPA** is just a specification; it does not provide an implementation.
- **Hibernate** is a popular implementation of JPA that provides additional features beyond the JPA standard.

2. Spring Data JPA

Spring Data JPA is a part of the Spring ecosystem that provides an abstraction over JPA. It reduces boilerplate code and simplifies database access.

Key Features of Spring Data JPA:

- Reduces **boilerplate code** by providing built-in CRUD operations.
- Supports **query methods** using naming conventions.
- Enables **pagination and sorting**.
- Allows **custom queries** using JPQL or native SQL.
- Provides **transaction management** with `@Transactional`.
- Supports **auditing and caching**.

3. Difference Between CrudRepository and JpaRepository

Feature	CrudRepository	JpaRepository
Definition	Basic CRUD operations.	Extends `CrudRepository` and provides additional JPA-specific features.
Pagination & Sorting	❌ Not available	✅ Available with `findAll(Pageable pageable)` and `findAll(Sort sort)`.
Flush Support	❌ Not available	✅ Supports `flush()` to synchronize persistence context with the database.

Feature	CrudRepository	JpaRepository
Batch Operations	❌ Not available	✅ Supports `saveAllAndFlush()`, `deleteAllInBatch()`, etc.
Custom Queries	✅ Supported with `@Query`	✅ Provides advanced features.

4. Key JPA Concepts

4.1 Flush in JPA

Flush is the process of synchronizing in-memory changes with the database.

```
userRepository.save(user);  // Change is stored in memory
userRepository.flush();  // Forces Hibernate to execute the SQL
immediately
```

✅ Useful when you need immediate database updates without committing transactions.

4.2 Pagination in JPA

Pagination helps in fetching large datasets efficiently by retrieving small subsets at a time.

```
Pageable pageable = PageRequest.of(0, 5); // Page number = 0,
Page size = 5
Page users = userRepository.findAll(pageable);
```

✅ Reduces database load and improves performance.

4.3 Batch Processing in JPA

Batch processing allows executing multiple database operations as a single batch.

```
List users = List.of(new User("Alice"), new User("Bob"));
userRepository.saveAllAndFlush(users);  // Saves all users in a
single batch operation
```

✅ Reduces database calls, improving performance.

5. Custom Queries in Spring Data JPA

Spring Data JPA allows writing custom queries using JPQL and native SQL with `@Query` annotation.

```
@Query("SELECT u FROM User u WHERE u.name = ?1")
User findByName(String name);
```

✅ Provides flexibility when default methods are not sufficient.

6. Entity Relationships in JPA

JPA supports entity relationships such as **One-To-One, One-To-Many, Many-To-One, and Many-To-Many** using annotations like `@OneToMany`, `@ManyToOne`, etc.

```
@Entity
public class Educator {
    @Id @GeneratedValue
    private Long id;

    @OneToMany(mappedBy = "educator")
    private List students;
}
```

✅ Helps in structuring data efficiently in relational databases.

7. JPA Annotations

Some commonly used JPA annotations are:

- `@Entity` - Defines a JPA entity.

- `@Table(name = "table_name")` - Specifies the table name.
- `@Id` - Marks the primary key.
- `@GeneratedValue(strategy = GenerationType.IDENTITY)` - Auto-generates primary key values.
- `@Column(name = "column_name")` - Specifies column details.
- `@OneToMany`, `@ManyToOne` - Defines relationships between entities.

8. Interview Questions and Answers

1. What is JPA?

JPA (Java Persistence API) is a specification that allows Java applications to manage relational data using ORM.

2. How is JPA different from Hibernate?

JPA is just a specification, while Hibernate is an implementation of JPA with additional features.

3. What are the main advantages of using JPA?

- Reduces boilerplate code.
- Provides built-in transaction management.
- Supports entity relationships and caching.
- Enables JPQL for complex queries.

4. What is the difference between `save()` and `saveAndFlush()` in JPA?

- `save()` stores an entity but does not immediately sync with the database.
- `saveAndFlush()` forces an immediate flush, writing changes to the database.

5. What is the use of `@Transactional` annotation in JPA?

`@Transactional` ensures that operations are executed within a transaction, ensuring consistency.

6. Explain the difference between `findById()` and `getOne()`.

- `findById()` returns an optional entity and queries the database immediately.
- `getOne()` returns a proxy and fetches data lazily when accessed.

7. What are different types of entity relationships in JPA?

- **One-To-One** (`@OneToOne`)
- **One-To-Many** (`@OneToMany`)
- **Many-To-One** (`@ManyToOne`)
- **Many-To-Many** (`@ManyToMany`)

8. What is the purpose of `EntityManager` in JPA?

`EntityManager` is used to manage database operations like persisting, updating, and deleting entities.

9. What is the difference between JPQL and Native SQL?

- **JPQL** is an object-oriented query language that works with entity names and fields.
- **Native SQL** allows writing direct SQL queries.

10. How can you enable caching in JPA?

Caching can be enabled using `@Cacheable` annotation or Hibernate's second-level cache.