# Difference between @JPARepository and @CrudRepository in Spring Data

A popular Spring Data Interview question for Java developers

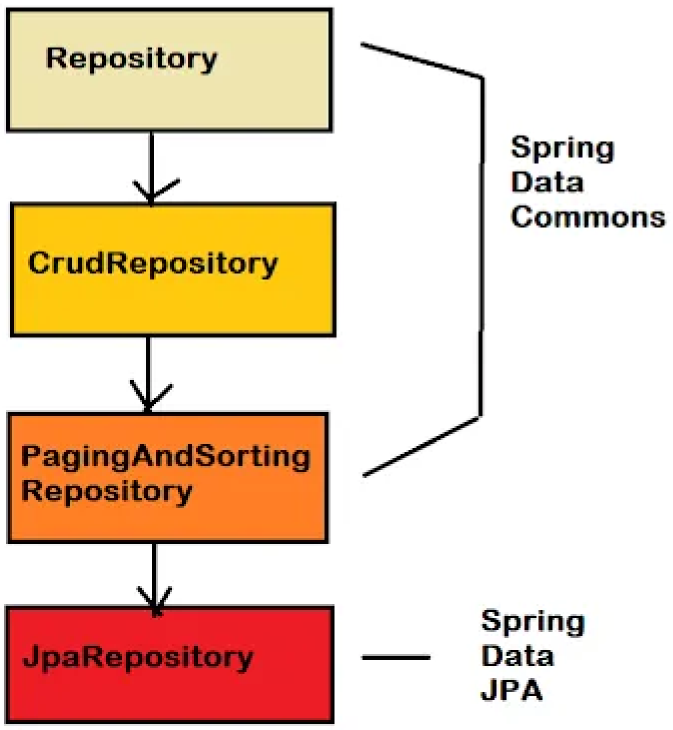
JAVINPAUL
JAN 25, 2024 · PAID

Hello guys, today, I Am going to share a popular Spring Data Interview question that was recently asked to one of my friends, yes, you guessed it correctly. It's about the Repository in JPA.

In the realm of Spring Data, two commonly used interfaces, namely **@JPARepository** and **@CrudRepository**, play pivotal roles in simplifying database interactions for Java developers.

These interfaces are essential components of the Spring Data repository abstraction, offering convenient methods for CRUD (Create, Read, Update, Delete) operations.

However, despite their similarities, @JPARepository and @CrudRepository exhibit distinct characteristics that cater to specific use cases. This article aims to unravel the differences between these two interfaces, providing clarity for developers navigating the Spring Data landscape.



## What is @JPARepository annotation in Spring Data JPA?

The @JPARepository interface is an extension of the base repository functionality provided by Spring Data. Specifically tailored for Java Persistence API (JPA), @JPARepository is designed to work seamlessly with JPA entities and the underlying database. It incorporates the capabilities of @CrudRepository while introducing additional methods that align with JPA-specific functionalities.

**Key Characteristics of @JPARepository:**

1. **Query Methods:** @JPARepository inherits the query methods from @CrudRepository, allowing developers to define queries based on method names conventionally. Additionally, it supports JPA-specific query creation, utilizing the power of JPA's query derivation mechanism.
2. **Entity Manager Integration:** @JPARepository integrates with the JPA entity manager, enabling efficient management of JPA entities and their lifecycle.
3. **Transparent JPA Features:** Developers can leverage advanced JPA features seamlessly with @JPARepository, such as entity listeners, custom query hints, and JPA-specific annotations.

## What is @CrudRepository in Spring Data JPA?

On the other hand, @CrudRepository represents the fundamental repository interface in Spring Data, providing basic CRUD operations for entities. This interface is not tied to any particular persistence mechanism, making it versatile and suitable for a wide range of data sources beyond JPA.

**Key Characteristics of @CrudRepository:**

1. **CRUD Operations:** @CrudRepository offers standard CRUD methods, including save, findById, findAll, update, and delete. These methods serve as the backbone for interacting with the underlying data source.
2. **Persistence-Agnostic:** Unlike @JPARepository, @CrudRepository is not tied to a specific persistence mechanism. It can be used with various data stores, such as MongoDB, Cassandra, and others, making it a more generic option.
3. **Basic Query Methods:** While @CrudRepository supports query methods, it lacks the JPA-specific query derivation capabilities present in @JPARepository.

## When to use @JPARepository and @CrudRepository?

The decision between @JPARepository and @CrudRepository largely depends on the project requirements and the underlying data store. If the application uses JPA for persistence and demands advanced JPA features, **@JPARepository** is the more suitable choice. On the other hand, if versatility and compatibility with multiple data sources are essential, @CrudRepository provides a more generic solution.

## An Example of @JPARepository and @CrudRepository in Spring Data JPA

Let's consider a simplified example to illustrate the usage of both `@JPARepository` and `@CrudRepository` in a Spring Data context.

Assume we have a simple entity class named `Book`:

```
import javax.persistence.Entity;

import javax.persistence.Id;

@Entity

public class Book {

@Id

private Long id;

private String title;

private String author;

// Constructors, getters, and setters

}
```

Now, let's explore how you might use `@JPARepository` and `@CrudRepository` to perform basic CRUD operations.

## Using @JPARepository:

```
import org.springframework.data.jpa.repository.JpaRepository;

public interface BookJPARepository extends JpaRepository<Book, Long> {

// Additional JPA-specific query methods can be declared here

}
```

With `@JPARepository`, you inherit basic CRUD methods (`save`, `findById`, `findAll`, etc.) from `JpaRepository`, and you can also declare additional methods tailored to JPA-specific queries if needed.

## Using @CrudRepository:

```
import org.springframework.data.repository.CrudRepository;

public interface BookCrudRepository extends CrudRepository<Book, Long>
{

// No additional JPA-specific methods can be declared here

}
```

With `@CrudRepository`, you get the standard CRUD methods (`save`, `findById`, `findAll`, etc.), but you can't declare additional JPA-specific query methods. This interface is more generic and can be used with various data sources beyond JPA.

## Example Usage in a Service

```
import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

@Service

public class BookService {

@Autowired

private BookJPARepository bookJPARepository;

@Autowired

private BookCrudRepository bookCrudRepository;

public void saveBookWithJPA(Book book) {

bookJPARepository.save(book);

}

public void saveBookWithCrudRepository(Book book) {

bookCrudRepository.save(book);

}
// Other service methods for retrieving and manipulating books

}
```

In the BookService class, you can inject either BookJPARepository or BookCrudRepository based on your project's requirements.

If you need to leverage JPA-specific features or query methods, you might opt for @JPARepository. If you want a more generic approach that can work with various data sources, @CrudRepository is a suitable choice.

**Conclusion:**

In the Spring Data ecosystem, **@JPARepository and @CrudRepository** serve as indispensable tools for simplifying data access layer development. Understanding the differences between these interfaces empowers developers to make informed choices based on their project's specific needs, ensuring efficient and tailored database interactions within the Spring framework.

And, if you are preparing for Java interviews you can also check my Java + Spring Interview + SQL Bundle on GUmroad, use code **friends20** to get a 20% discount also

## 1 Comment

Soma Soma's Substack Jan 25

good explanation