Repository Pattern and Query Methods in Spring Data JPA

The Repository pattern is a design pattern that abstracts the data access layer, providing a clean and organized way to interact with the data source. In the context of Spring Data JPA, repositories are interfaces that handle persistence logic and data access, allowing for easy implementation of CRUD operations and custom queries.

Repository Pattern

The Repository pattern separates the business logic from the data access logic by providing a standardized interface for data operations. Spring Data JPA simplifies the implementation of the Repository pattern by offering built-in interfaces like JpaRepository, which provides generic methods for CRUD operations.

Example:

```
@Repository
public interface PatientRepository extends JpaRepository<Patient, Long> {
 List<Patient> findBySurname(String surname);
}
```

CRUD Operations

CRUD operations (Create, Read, Update, Delete) are essential for any application interacting with a database.

Spring Data JPA provides built-in methods like save(), findById(), findAll(), deleteById(), etc., which can be used to perform these operations.

Example:

```
public Patient createPatient(Patient patient) {
 return patientRepository.save(patient);
}
public void deletePatient(Long id) {
 patientRepository.deleteById(id);
}
```

Custom Query Methods

Custom query methods in Spring Data JPA can be created by following specific naming conventions. These methods can perform more complex queries without requiring explicit JPQL or SQL.

Example:

List<Patient> findBySurnameAndDiagnosis(String surname, String diagnosis);

The above method will automatically generate a query that fetches patients based on their surname and diagnosis