Repository pattern and query methods

The Repository Pattern is a design pattern that abstracts the data layer, providing a collection-like interface for accessing domain objects. It acts as a bridge between the domain and data mapping layers, ensuring that data access logic is decoupled from business logic.

Benefits of Using the Repository Pattern

- **Separation of Concerns**: Keeps the data access logic separate from business logic.
- **Increased Testability**: Facilitates easier unit testing of business logic without involving data access code.
- **Simplified Data Access**: Provides a consistent API for CRUD operations and custom queries.

Spring Data JPA

What is Spring Data JPA?

Spring Data JPA is a part of the Spring Data project, aimed at simplifying data access using JPA (Java Persistence API). It provides an abstraction layer over JPA, reducing boilerplate code and offering easy integration with various databases.

Key Concepts

- **Repositories**: Interfaces for CRUD operations.
- **Entities**: Java objects representing database tables.
- **JPA Providers**: Implementations like Hibernate that manage entity persistence.

Implementing the Repository Pattern in Spring Data JPA

Defining Repository Interfaces

In Spring Data JPA, repositories are interfaces that extend `JpaRepository`, `CrudRepository`, or `PagingAndSortingRepository`. These interfaces provide methods for common CRUD operations.

CRUD Operations

Spring Data JPA repositories come with built-in methods like `save()`, `findById()`, `delete()`, and more, which simplify CRUD operations.

Custom Query Methods

Repositories can also include custom query methods derived from method names or annotated with `@Query`.

Query Methods in Spring Data JPA

Derived Query Methods

Spring Data JPA can automatically generate queries based on method names. For example, `findBySurname` generates a query to find doctors by their surname.