Spring Boot provides a powerful and convenient way to interact with databases using Java Persistence API (JPA) through its built-in repository support. Spring Data JPA offers a variety of methods that you can use to perform common database operations without writing explicit SQL queries. These methods are based on method naming conventions and are automatically implemented by Spring Data JPA.

Here are some of the commonly used Spring Data JPA methods:

1. **Save:**
   * **save(entity)**: Saves an entity to the database.
2. **Find:**
   * **findById(id)**: Retrieves an entity by its primary key.
   * **findAll()**: Retrieves all entities.
   * **findAllById(ids)**: Retrieves a list of entities by a list of primary keys.
3. **Count:**
   * **count()**: Returns the number of entities.
4. **Delete:**
   * **delete(entity)**: Deletes a specific entity.
   * **deleteById(id)**: Deletes an entity by its primary key.
   * **deleteAll()**: Deletes all entities.
   * **deleteAllById(ids)**: Deletes a list of entities by a list of primary keys.
5. **Custom Queries:**
   * You can create custom queries by following Spring Data JPA's method naming conventions. For example:
     + **findByFirstName(String firstName)**: Retrieves entities by the value of the "firstName" property.
     + **findByLastNameAndAge(String lastName, int age)**: Retrieves entities that match both the "lastName" and "age" properties.
     + **findByAgeLessThan(int age)**: Retrieves entities with an "age" property less than the specified value.
6. **Sorting and Paging:**
   * **findAll(Sort sort)**: Retrieves all entities with sorting.
   * **findAll(Pageable pageable)**: Retrieves entities with paging and sorting.
7. **Derived Queries:**
   * Spring Data JPA automatically derives queries based on method names and property names in your entity classes. This reduces the need to write custom queries for simple operations.
8. **@Query Annotation:**
   * You can also define custom queries using the **@Query** annotation and write JPQL or native SQL queries as needed.

These are just a few examples of the Spring Data JPA methods available in Spring Boot. By following naming conventions and leveraging Spring Data JPA's powerful features, you can significantly reduce the amount of boilerplate code needed for database interactions in your Spring Boot application.

Top of Form