Common Annotations in a Spring Data JPA App

@ivanfranchin

@Entity

Marks a Java class as an object that represents a table in the database.

```
@Entity
public class Product {

    // Class fields
}
```

@Id

Specifies the primary key of an entity, which uniquely identifies each record in the database.

```
@Entity
public class Product {

    @Id
    private Long id;
    // Other fields
}
```

@GeneratedValue

Works with **@Id** to automatically generate a unique value for the primary key.

```
@Entity
public class Product {

    @Id
    @GeneratedValue
    private Long id;

    // Other fields
}
```

@Table

Provides information about the database table associated with an entity, allowing customization of table properties like name and schema.

```
@Entity
@Table(name = "products")
public class Product {


    // Class fields
}
```

@Column

Specifies details about a column in a database table, such as its name, length, and other attributes.

```
@Entity
public class Product {
    @Id
    private Long id;

    @Column(name = "product_name", length = 50)
    private String name;

    // Other fields
}
```

@OneToOne

Establishes a one-to-one relationship between two entities, meaning each record in one entity corresponds to exactly one record in another entity.

```
@Entity
public class Person {

    @OneToOne
    private Address address;

    // Other fields
}
```

@OneToMany

Defines a one-to-many relationship between two entities, where each record in the owning entity can be related to multiple records in the referenced entity.

```
@Entity
public class Team {

    @OneToMany(mappedBy = "team")
    private List<Player> players;

    // Other fields
}
```

@ManyToOne

Represents a many-to-one relationship between two entities, indicating that multiple records in the owning entity can refer to a single record in the referenced entity.

```
@Entity
public class Player {

    @ManyToOne
    private Team team;

    // Other fields
}
```

@ManyToMany

Establishes a many-to-many relationship between two entities, where each record in one entity can be associated with multiple records in another entity, and vice versa.

```
@Entity
public class Student {

    @ManyToMany
    private List<Course> courses;

    // Other fields
}
```

@JoinColumn

Specifies the foreign key column when customizing the mapping of a relationship, often used with **@OneToOne** and **@ManyToOne** to define the foreign key column name and attributes.

```
@Entity
public class Order {

    @ManyToOne
    @JoinColumn(name = "customer_id")
    private Customer customer;

    // Other fields
}
```

@JoinTable

Defines a join table for a many-to-many relationship, specifying the intermediate table that manages the relationship between entities.

```
@Entity
public class Student {
    @ManyToMany
    @JoinTable(
        name = "student_course",
        joinColumns = @JoinColumn(name = "student_id"),
        inverseJoinColumns = @JoinColumn(name = "course_id")
    )
    private List<Course> courses;
    // Other fields
}
```

@Repository

Indicates that a Java class or interface is responsible for handling database operations. It helps in handling errors and makes accessing data easier.

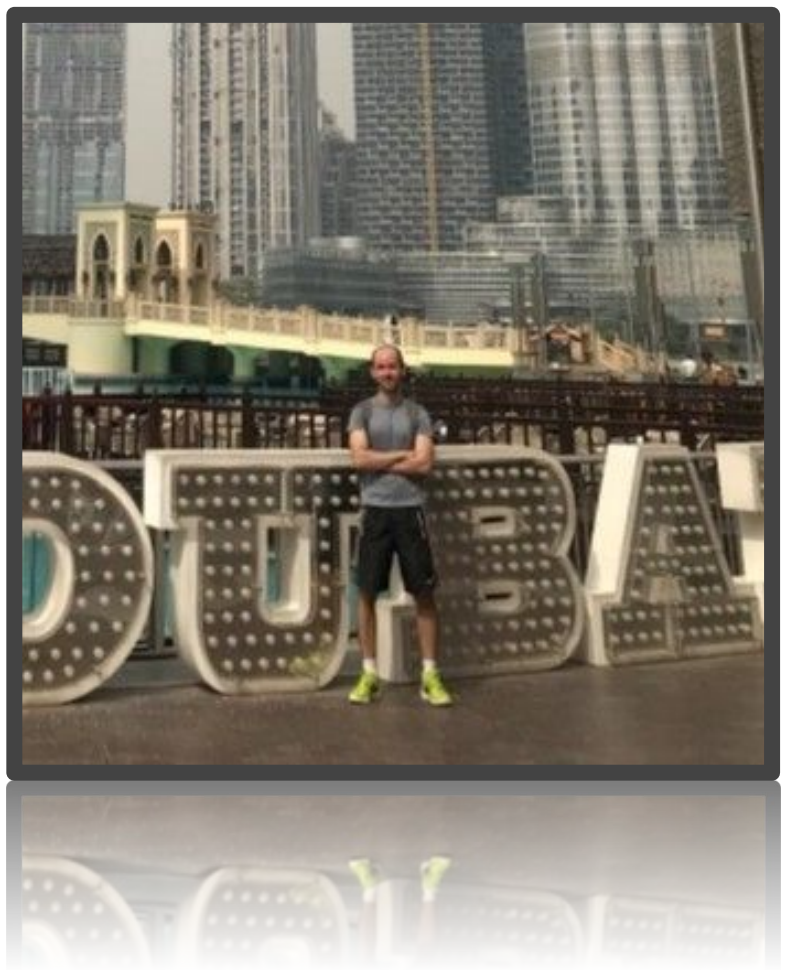
```
@Repository
public interface ProductRepository
    extends JpaRepository<Product, Long> {

    // Interface methods for database operations
}
```

@Query

Allows custom queries to be declared directly on the repository interface, enabling the definition of complex queries beyond the ones provided by default.

```
@Repository
public interface ProductRepository
    extends JpaRepository<Product, Long> {
    @Query("SELECT p FROM Product p WHERE p.price > :price")
    List<Product> findProductsByPriceGreaterThan(
        @Param("price") BigDecimal price);
}
```

That’s all



I hope you enjoy it!

Let’s connect: ***@ivanfranchin***

Follow me: ***@ivangfr***