**@DynamicInsert and @DynamicUpdate – 2024**

**@DynamicInsert:** Hibernate’s proprietary class level annotation which generates sql **insert** statement for the non-null columns or non-null fields.

**@DynamicUpdate:** Hibernate’s proprietary class level annotation which generates sql **update** statement for the modified columns or modified fields.

**Downside**

* In order to find out the changed columns, Hibernate needs to track the state of the current entity.
* So, when we change any field of an entity, it compares the current and the modified states of the entity.
* There may be performance overhead.

**When to use**

* If an entity represents a table that has a large number of columns and only a few of these columns are required to be updated frequently

Detailed Analysis and Example given below.

@Entity(name = "Product")

@Table(name = "product")

@Getter @Setter

**public** **class** Product {

@Id @GeneratedValue

**private** Long id;

**private** String name;

**private** String rating;

**private** String manufacturer;

**private** **int** price;

}

**Use Case-1:** Without the use of **@DynamicInsert**

**Hibernate formed Query 🡺** **insert into product (manufacturer, name, price, rating, id) values (?, ?, ?, ?, ?)**

**Use Case-2**: With the use of **@DynamicInsert**

**@DynamicInsert**

@Entity(name = "Product") @Table(name = "product")

@Getter @Setter

**public** **class** Product {

@Id @GeneratedValue

**private** Long id;

**private** String name;

**private** String rating;

**private** String manufacturer;

**private** **int** price;

}

**private** **void** saveProduct() {

Product p = **new** Product();

p.setName("Wahsing Machine");

p.setPrice(2300);

// p.setRating("Ok"); 🡺**Rating is null**

// p.setManufacturer("Nokia"); 🡺**Manufacturer is null**

prodRepo.save(p);

System.***out***.println("Product saved successfully ...");

}

**Hibernate formed Query 🡺 insert into product (name, price, id) values (?, ?, ?) 🡺 Only Non-Null Fields are taken**

**In case of @DynamicUpdate**

**Use Case -1**: Without the use of **@DynamicUpdate**

**Hibernate formed Query 🡺 Update product set manufacturer=?, name=?, price=?, rating=? Where id=?**

**Use Case-2:** With the use of **@DynamicUpdate**

@DynamicInsert

**@DynamicUpdate**

@Entity(name = "Product") @Table(name = "product")

@Getter @Setter

**public** **class** Product {

@Id @GeneratedValue

**private** Long id;

**private** String name;

**private** String rating;

**private** String manufacturer;

**private** **int** price;

}

private void updateProduct() {

Product p = new Product();

p.setId(202L);

p.setName("Wahsing Machine");

p.setPrice(2300);

p.setRating("Ok"); 🡺 new value to update

p.setManufacturer("LG"); 🡺 new value to update

prodRepo.save(p);

System.out.println("Product saved successfully ...");

}

**Hibernate formed Query 🡺 update product set manufacturer=?, rating=? Where id=?**