**@DynamicInsert and @DynamicUpdate – 2024**

These annotations enable the generation of SQL queries that include only the modified fields, reducing the amount of data transmitted and decreasing the load on the database. 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. This means that **@DynamicUpdate has a performance overhead associated with it**. Therefore, we should only use it when it’s actually required. Certainly, there are a few scenarios where we should use this annotation — for example, 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. Also, when we use version-less optimistic locking, we need to use @DynamicUpdate.

An example is given below.

@Entity  
@Getter  
@Setter  
@ToString  
@DynamicInsert  
@DynamicUpdate  
@NoArgsConstructor  
@AllArgsConstructor  
@Table(name = "persons")  
public class Person {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long id;  
  
 @Column(name = "name")  
 private String name;  
   
 @Column(name = "age")  
 private Integer age;  
  
 // equals and hashCode  
}