**Spring Boot Annotation @GeneratedValue**

The **@GeneratedValue annotation** in JPA (Java Persistence API) is used to specify the strategy used for generating primary key values for entities.

The **GenerationType.IDENTITY strategy** is commonly used with databases that support auto-incrementing columns, **such as MySQL and PostgreSQL.**

1. **GenerationType.IDENTITY:**

* This strategy relies on an auto-incremented database column for primary key generation.
* It is suitable for databases that support auto-incrementing columns (e.g., MySQL, PostgreSQL).
* The database automatically generates a unique value for the primary key when a new record is inserted.

Example:

@Id  
@GeneratedValue(strategy = GenerationType.IDENTITY)  
private Long id;

**2. GenerationType.SEQUENCE:**

* This strategy relies on a database sequence to generate primary key values.
* Not all databases support sequences, so it’s essential to check database compatibility.
* You may need to define the sequence explicitly in your database.

Example:

@Id  
@GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "expense\_sequence")  
@SequenceGenerator(name = "expense\_sequence", sequenceName = "expense\_sequence", allocationSize = 1)  
private Long id;

**3. GenerationType.TABLE:**

* This strategy uses a separate table to maintain a counter for generating primary key values.
* It can be less efficient than other strategies and is not commonly used.

Example:

@Id  
@GeneratedValue(strategy = GenerationType.TABLE, generator = "expense\_generator")  
@TableGenerator(name = "expense\_generator", table = "id\_generator", pkColumnName = "id\_key", pkColumnValue = "expense\_id", allocationSize = 1)  
private Long id;

**4. GenerationType.AUTO:**

* The AUTO strategy allows the JPA provider to choose the appropriate strategy based on the underlying database capabilities.
* It’s a portable way to generate primary keys, but the actual strategy used depends on the database.

Example:

@Id  
@GeneratedValue(strategy = GenerationType.AUTO)  
private Long id;

In most cases, GenerationType.IDENTITY is a good choice when working with databases that support auto-incrementing columns.

It simplifies the code and leverages the native capabilities of the database.

Fetch Types in Hibernate

**EAGER**

Load the associated data of the other entity, beforehand which is bit costly.

**LAZY**

Load the associated data of the other entity, only when requested. This is done on demand.

There are specified fetching types for each relationship type which is applied by Hibernate by default.

OneToMany: LAZY

ManyToOne: EAGER

ManyToMany: LAZY

OneToOne: EAGER