@generateValue

It is used to tell the ORM (such as Hibernate) that the ID or Primary Key column is generated automatically instead of you entering it manually.

```
import jakarta.persistence.*;

@Entity
@Table(name = "items")
public class Item {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;    // ال as Primary Key

    private String name;

    // getters & setters
}
```

Strategies (GenerationType)

There are several types that determine how the value is generated:

- IDENTITY

  Relies on Auto Increment in the database (such as MySQL, SQL Server).

  Example: A column is automatically generated as 1, 2, 3...

- SEQUENCE

  Sequence is used in databases such as Oracle and PostgreSQL.

  You need to know about sequences in the database.

  Default Format

If you define a normal sequence like this in Oracle, for example:

Copy code

```
CREATE SEQUENCE item_seq START WITH 1 INCREMENT BY 1;
```

START WITH 1 → First value starting from 1

INCREMENT BY 1 → Increments by 1 each time

➔There are also other options.

MAXVALUE: Specifies the maximum value it will reach.

CYCLE: When it reaches MAXVALUE, it starts over.

CACHE: Stores numbers in memory for speed (may cause overflows if the server crashes).

Copy code

```
CREATE SEQUENCE item_seq
START WITH 10
INCREMENT BY 10
MAXVALUE 1000
CYCLE
CACHE 20;
```

When it reaches 1000, it starts again from 10 (because of the cycle).

Binding with JPA ; In the code:

```
@Id
@GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "item_seq")
@SequenceGenerator(name = "item_seq", sequenceName = "ITEM_SEQ", allocationSize = 1)
private Long id;
```

sequenceName = "ITEM_SEQ" → The sequence name in Oracle.

allocationSize = 1 → Specifies how many numbers Hibernate allocates each time (very important to prevent overflows).

➢ TABLE

Stores the last used number in a special table, incrementing it each time.

Little used.

Hibernate/JPA doesn't rely on the Auto Increment or Sequence features of the database.

👉 Instead, it creates a special table in the database that stores the last generated number and increments this number each time a new insert is made.

In other words, it mimics the concept of sequence, but using a table.

🛠 How it works

Hibernate (if you don't specify anything) will create a table named hibernate_sequences or sequences, with two columns:

next_val	sequence_name
5	item_seq

sequence_name → The name of the entity/generator.

next_val → The number to use next time.

The table in the database

Hibernate will create a table named id_generator (if it doesn't exist):

```
CREATE TABLE id_generator (
    seq_name VARCHAR(50) NOT NULL,
    next_val NUMBER NOT NULL,
    PRIMARY KEY (seq_name)
);
```

```
INSERT INTO id_generator (seq_name, next_val) VALUES ('item_seq', 1);
```

Code

```
import jakarta.persistence.*;

@Entity
@Table(name = "items")
public class Item {

    @Id
    @GeneratedValue(strategy = GenerationType.TABLE, generator = "item_gen")
    @TableGenerator(
        name = "item_gen",              // The generator name used in the entity
        table = "id_generator",         // The table that stores the sequence values
        pkColumnName = "seq_name",      // The column that holds the sequence key (identifier)
        valueColumnName = "next_val",   // The column that holds the current sequence value
        pkColumnValue = "item_seq",     // The specific key value for this entity sequence
        allocationSize = 1              // Increment size (how much the value increases each time
    )
    private Long id;                    // Primary Key generated using TABLE strategy

    private String name;                // Example field
}
```

What happens when you use persist?

1. Hibernate reads next_val from the table.
2. It assigns the value to the new entity.
3. It adds next_val and stores it back in the table.

✅ Summary

TABLE strategy is useful when the database doesn't support Auto Increment or Sequence (like some older ones, or if you want to control it yourself).

It's a little slower than SEQUENCE or IDENTITY, because each Insert operation requires updating the table.

- AUTO

  Hibernate automatically decides based on the database type.

⚡ How does it actually work?

- If Oracle/PostgreSQL, use SEQUENCE.
- If MySQL/SQL Server, use IDENTITY (auto increment).
- If the database is outdated or doesn't support one, use TABLE.

This means that AUTO makes the code portable (it can run on more than one database without changing it).

✅ AUTO Advantages

You don't need to specify a strategy. Hibernate adjusts it according to the database.

It's suitable if your project may move between more than one database.

❌ Disadvantages

It doesn't always choose the best one (it can be a bit slow if using a table).

Sometimes you might prefer manual control (for example, Oracle - you want a specific SEQUENCE instead of the default).

Notes

In Oracle, it is normal to use SEQUENCE and not IDENTITY, because historically (before version 12c) Oracle did not have an AUTO INCREMENT / IDENTITY column like MySQL or SQL Server.

So why is it working for you now?

From Oracle 12c (2014) onwards, support for Identity Columns has been added.

⚡ Quick Comparison

Before 12c → You must use SEQUENCE + @SequenceGenerator.

From 12c onwards → You can use IDENTITY normally.