


# Hibernate ID Generation Strategies

In Hibernate (and JPA), every entity must have a **primary key** (identifier). Hibernate provides several strategies to generate unique IDs for entities, ensuring data integrity and persistence across different databases.

---

## 1. Manual (Assigned) Strategy

- Developer is responsible for setting the ID before persisting the entity.
- Hibernate will not generate the value automatically.

A code snippet in a dark-themed editor with three colored window control buttons (red, yellow, green) at the top left. The code is: 

```
@Id
private Long id;
```

```
@Id
private Long id;
```

**Use case:** When IDs come from external systems (e.g., employee code, national ID).

---

## 2. GenerationType.AUTO

- Hibernate chooses the strategy automatically based on the underlying database dialect.
- Can result in IDENTITY, SEQUENCE, or TABLE depending on the DB.

A code snippet in a dark-themed editor with three colored window control buttons (red, yellow, green) at the top left. The code is: 

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

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

**Use case:** General purpose when you don't want to hardcode the strategy.

---

## 3. GenerationType.IDENTITY

- Uses the database's auto-increment/identity column.
- Common in MySQL, SQL Server, PostgreSQL.

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

**Advantages:** Simple, fast, relies on database.

**Disadvantages:** Not suitable for batch inserts (because ID is generated only after insert).

---

## 4. GenerationType.SEQUENCE

- Uses a database sequence object to generate unique IDs.
- Supported in Oracle, PostgreSQL, H2.

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

**Advantages:** Efficient, can fetch IDs in advance. **Disadvantages:** Requires sequence support in the database.

---

## 5. GenerationType.TABLE

- Uses a database table to generate IDs by storing and incrementing values.
- Works in all databases.

```
@Id
@GeneratedValue(strategy = GenerationType.TABLE, generator = "player_tab")
@TableGenerator(name = "player_tab", table = "id_gen", pkColumnName = "gen_name",
valueColumnName = "gen_val", allocationSize = 1)
private Long id;
```

**Advantages:** Database-independent. **Disadvantages:** Slower due to extra table lookup.

---

## 6. UUID Strategy (Hibernate-specific)

- Generates Universally Unique Identifiers (UUIDs).
- Useful for distributed systems.



```
@Id
@GeneratedValue(generator = "uuid")
@GenericGenerator(name = "uuid", strategy = "uuid2")
private String id;
```

**Advantages:** Globally unique, no collisions. **Disadvantages:** Larger storage size (string-based IDs).