# Enterprise Programming using JAVA

## Chapter-4: Hibernet (ORM)

**Prof. ARNIKA PATEL**
**Assistant Professor**
**Department of CSE**

Parul® University

NAAC A++

## Content

1. Generator Class…………………………………… 3
2. Dialects…………………………………………… 4
3. Mapping…………………………………………. 16
4. Annotation……………………………………….
18

INDEX

## Generator Class

- A **generator** class is used to generate an **ID** for an object, which is going to be inserted in the database as a primary key.
- The **ID** is a unique identifier for the objects of the persistent class.
- We can use any **generator** classes in our application as per our requirements.
- Hibernate provides a **<generator>** tag to represent the generator.
- However, it is a sub-element of <id>.
- There is a shortcut name for each predefined **generator** class so that the **<generator>** tag has to configure with this shortcut name.

## Generator Class

**Hibernate** provides many predefined **generator** classes. Some of the important predefined generator classes in hibernate are:

1. assigned
2. foreign
3. hilo
4. identity
5. increment
6. native
7. sequence
8. uuid

## Generator Class

### 1. assigned generator

The **assigned generator** is the default generator. By default hibernate consider **assigned** as a generator and **assigned** is a shortcut name given for the **Assigned class**.

This class returns the same id set by us to the hibernate which in turn store an object with that id in the database.

## Generator Class

### 1. assigned generator

```
<hibernate-mapping>
        <class …>
                <id name=”studentId” column=”sid”>
                        <generator class=”assigned”></generator>
                </id>
        </class>
</hibernate-mapping>
```

## Generator Class

### 2. foreign generator

**foreign** is a shortcut name given for the **ForeignGenerator** class.

The **ForeignGenerator** is only suitable for one-one relationships.

It returns the id of the **parent object** as the id for the **child object.**

## Generator Class

**2. foreign generator**

```
<hibernate-mapping>
      <class …>
              <id name=”studentId” column=”sid”>
                    <generator class=”foreign”></generator>
              </id>
      </class>
</hibernate-mapping>
```

## Generator Class

### 3. hilo generator

**hilo** is a shortcut name for the **TableHiloGenerator** class.

While configuring **hilo** generator class in the **hbm.xml** file, we pass **“table”, “column”**, and **“max_lo”** as parameters.

The default value of above three parameters given by **Hibernate** are:

table = **hibernate_unique_key**
column = **next_hi**
max_lo = **32767**

## Generator Class

**3. hilo generator**

```
<hibernate-mapping>
     <class …>
            <id name=”studentId” column=”sid”>
                 <generator class=”hilo”>
                        <param
                        name=”table”>document</param>
                        <param name=”column”>column</param>
                       <param name=”max_lo”>13210</param>
                 </generator>
            </id>
     </class>
</hibernate-mapping>
```

## Generator Class

**4. identity generator**

**identity** is a shortcut name given for the **IdentityGenerator** class.

It reads an auto-incremented column algorithm of the database and takes that value and returns it as id to hibernate.

**Identity** generator supports databases like **Sybase, My SQL, MS SQL Server, DB2**, **and HypersonicSQL** but doesn't support **Oracle** database because there is no autoincrement functionality in **Oracle** database.

Hence identity generator can be considered as a database-dependent generator.

## Generator Class

### 4. identity generator

```
<hibernate-mapping>
      <class …>
               <id name=”studentId” column=”sid”>
                      <generator class=”identity”></generator>
              </id>
      </class>
</hibernate-mapping>
```

## Generator Class

**5. increment generator**

An **increment** is a shortcut name given for the **IncrementGenerator** class.

The **IncrementGenerator** class reads the max value of the existing ID in the database table and then increments it by one and returns the ID value to hibernate.

## Generator Class

### 5. increment generator

```
<hibernate-mapping>
    <class …>
           <id name=”studentId” column=”sid”>
                <generator class=”increment”></generator>
           </id>
    </class>
</hibernate-mapping>
```

## Generator Class

**6. native generator**

**native** is the shortcut name given for the **NativeGenerator** class.

If the database supports an **identity** generator, then it acts as **identity** otherwise it will check for the database support of other generators.

It selects **identity, sequence**, or **hilo** depending upon the capabilities of the underlying database.

## Generator Class

**6. native generator**

```
<hibernate-mapping>
    <class …>
            <id name=”studentId” column=”sid”>
                  <generator class=”native”></generator>
            </id>
    </class>
</hibernate-mapping>
```

## Generator Class

### 7. sequence generator

**sequence** is a shortcut name given for the **SequenceGenerator** class.

It reads the next value of a database sequence and then returns that value as **id** to the hibernate.
**Syntax to create a sequence in the database:**

*create sequence <sequence_name> start with <random_number> increment by <random_number>*

## Generator Class

### 7. sequence generator

```
<hibernate-mapping>
   <class …>
          <id name=”studentId” column=”sid”>
                <generator class=”sequence”>
                      <param 
name=”sequence”>sequential_datasource</param>
                </generator>
          </id>
   </class>
</hibernate-mapping>
```

## Generator Class

**8. uuid generator**

**uuid** is a shortcut name given for the **AbstractUUIDGenerator** class.

It generates a unique string Identifier and returns to hibernate based on the **IP Address of the machine**, **Start-up time of JVM, System time**, and **Counter value in JVM.**

Parul® University NAAC A++

## Generator Class

### 8. uuid generator

```
<hibernate-mapping>
  <class …>
         <id name=”studentId” column=”sid”>
              <generator class=”uuid”>    </generator>
         </id>
  </class>
</hibernate-mapping>
```

## PPT Content Resources Reference Sample:

1. **Book Reference**

   Jim Farley, William Crawford, David Flanagan. Java Enterprise in a Nutshell, O'Reilly

2. **Book Reference**

   Rocha, R., Purificação, J. (2018). Java EE 8 Design Patterns and Best Practices: Build Enterprise-ready Scalable Applications with Architectural Design Patterns. Germany: Packt Publishing..

3. **Website Reference**

   https://www.scribd.com/document/268349254/Java-8-Programming-Black-Book .

4. **Sources**

   https://developers.redhat.com/topics/enterprise-java

5. **Article**

   https://www.researchgate.net/publication/276412369_Advanced_Java_Programming

Parul® University | NAAC GRADE A++

https://paruluniversity.ac.in/