* For a new object, before an object is going to be stored in a database, id for that object is mandatory because hibernate checks whether already another object with the same id is exist in the database or not(table or not) using the id of the new object.
* If already another object exist with same id in data base then hibernate throws an exception.
* In hibernate for an object we can manually assign the value for id or we can use predefined id generators technique given by hibernate for genreating id of the object.
* We need to inform the hibernate that whether a developer is going to assing the id or it should be generated by hibernate using <generator> tag under <id> tag of mapping file .
* Hibernate has provided some predefined id generators techniques for generating id, if the existing id generation techniques are not suitable for our application then we can also create our own in id generators technique. This is called custom id ggenerators.

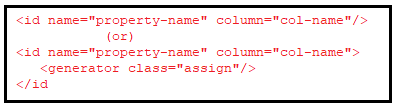
Hibernate provided the following generator classes/techniques:

1. **Assigned Generator:-** It is default id generator .

If generator is assingned then the programmer has to assing the id for an object. If not then hibernate throws exception.

Assigned is database independent generator class.

Ex:-

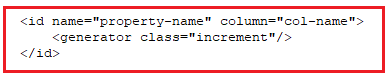


1. **Increment Generator:-**Before object is going to be stored in table, hibernate gets maximum value from id column. This generator class uses max(id)+1 formula for generating an id for new object.

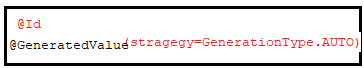
If we manually assing the id value for an object then hibernate overirdes it with value generated using increment generator.

If the database table does not exist contain any object(rows) then max(id) will be zero.

**XML Configuration:-**

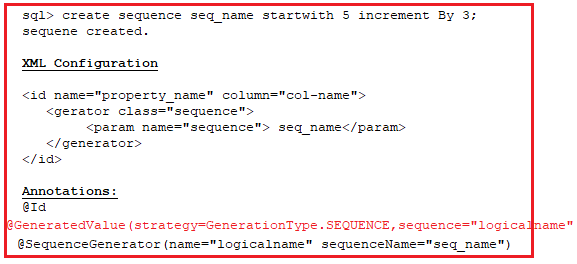


**Annotation:-**

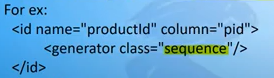


1. **Sequence generator:-** This generator class reads a next value of given sequence from data base and assign that value for id and then object will be inserted in to table.

If we add generator class as sequence then we need to configure a parameter under generator tag called sequence. For this parameter we need to pass sequence name of database as value.

 if we do not configure<param> tag then hibernate creates its own sequence in database with the name “hibernate\_sequence”, then select next val of the sequence and assign it for id of the object and then the object is inserted into database.

If hibernate want to create its own sequence in data abse then in hernate.cfg.xml file, hbm2ddl.auto property should be configure with creates of update as value.



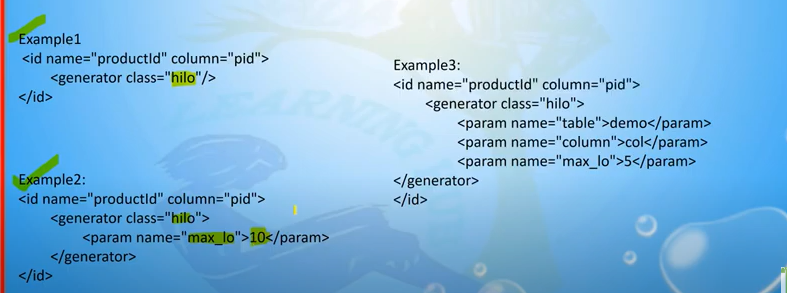
**d.Hilo Generator:-** This generator class generates is for an object when first time used as one(1) from second time onwards, the id will be generated using max\_lo\*next\_hi+next\_hi formula.

When hilo generator is configured for <id> tag then hibernate expects three parameters configures along with hilo generator

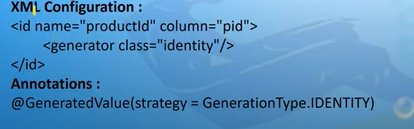
1. Table
2. Column
3. Max\_lo

If parameters are not configured then hibernate considers it “s default values. The default values of parameters of hilo generators are

1. Table:hihernate\_unique\_key
2. Column:next\_hi
3. Max\_lo:32767



**e.Identity Generator:-** This generator class is used to inform that id column of table is auto increment column of the table. When identity generator is configured then hibernate does not generate id for an object. Hibernate inserts with out id and id will be added by data base.



Example:- This application demonstrates the “incrementor “ generator.

**Domain class:Student.java**

**package** com.hib.domain;

**import** java.io.Serializable;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

@Entity

@Table(name="student")

**public** **class** Student **implements** Serializable{

@Id

@GeneratedValue(strategy=GenerationType.***AUTO***)

**private** **int** sid;

@Column(name="sname")

**private** String sname;

**public** **int** getSid() {

**return** sid;

}

**public** **void** setSid(**int** sid) {

**this**.sid = sid;

}

**public** String getSname() {

**return** sname;

}

**public** **void** setSname(String sname) {

**this**.sname = sname;

}

}

**HibernateConfiguration File**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name=*"hibernate.connection.driver\_class"*>oracle.jdbc.OracleDriver</property>

<property name=*"hibernate.connection.url"*>jdbc:oracle:thin:@localhost:1521:xe</property>

<property name=*"hibernate.connection.username"*>system</property>

<property name=*"hibernate.connection.password"*>tiger</property>

<property name=*"hibernate.dialect"*>org.hibernate.dialect.Oracle10gDialect</property>

<property name=*"hibernate.hbm2ddl.auto"*>update</property>

<mapping class=*"com.hib.domain.Student"*/>

</session-factory>

</hibernate-configuration>

**ClientApplication:Test.java**

**package** com.hib.domain;

**import** java.io.FileNotFoundException;

**import** java.sql.SQLException;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.hibernate.cfg.Configuration;

**import** org.hibernate.classic.Session;

**public** **class** Test {

**public** **static** **void** main(String[] args)**throws** SQLException,FileNotFoundException {

**try** {

Configuration cfg=**new** Configuration();

cfg.configure("com/hib/domain/hibernate.cfg.xml");

SessionFactory sf=cfg.buildSessionFactory();

Session s=sf.openSession();

Student s1=**new** Student();

Transaction t1=s.beginTransaction();

s1.setSid(1);

s1.setSname("suma");

**int** pk=(Integer)s.save(s1);

**if**(pk==1) {

System.***out***.println("Record Inserted");

}

**else**

{

System.***out***.println("Record not inserted");

}

t1.commit();

}

**catch**(Exception e) {

e.printStackTrace();

}

}

}

Output:-

