The hibernate application can manage the underlying database schema using schema generation tools. Database schema managing means

1. Creating table.
2. Dropping table
3. Update the table structure.
4. ….etc.

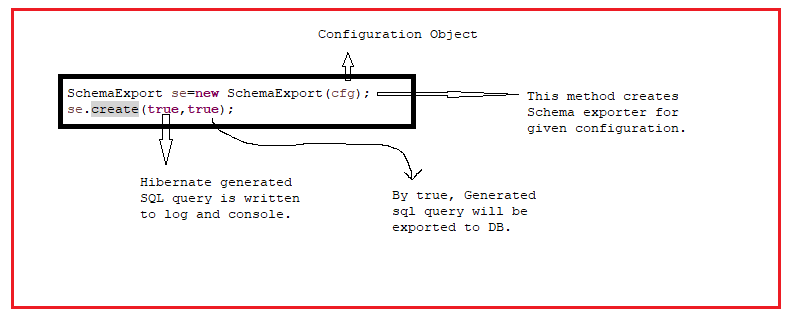
The schema generation tools uses either **mapping file or annotated class** to manage underlying database schema.The schema generation tools are

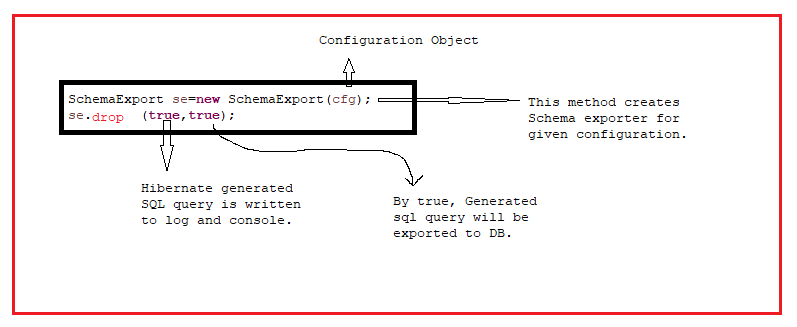
1. SchemaExport
2. SchemaUpdate
3. Hbm2ddl.auto

**1.Schema Export:-** The Basic O-R mapping details present in Configuration object. From configuration object, schema export gets required details(table name, coloumns names, column size, column data types …etc). Schema export tool checks whether tables exists with same name or not in DB.

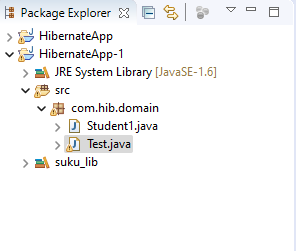
If table does not exist with same name, it will create a new table with mapping detailes.

If table exist with same name, it will drop the existing table and creates new table with mapping details.





Example:- In this table does not exist.



Persistent class: student1.java

**package** com.hib.domain;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

@Entity

@Table(name="student1")

**public** **class** Student1 {

@Id

@Column(name="sid")

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

@Column(name="sname")

**private** String sname;

@Column(name="marks")

**private** **float** marks;

**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;

}

**public** **float** getMarks() {

**return** marks;

}

**public** **void** setMarks(**float** marks) {

**this**.marks = marks;

}

}

Client Application: Test.java

**package** com.hib.domain;

**import** org.hibernate.HibernateException;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

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

**import** org.hibernate.tool.hbm2ddl.SchemaExport;

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

**public** **static** **void** main(String[] args) {

**try** {

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

cfg.setProperty("hibernate.connection.driver\_class","oracle.jdbc.OracleDriver");

cfg.setProperty("hibernate.connection.url","jdbc:oracle:thin:@localhost:1521:xe");

cfg.setProperty("hibernate.connection.username","system");

cfg.setProperty("hibernate.connection.password","tiger");

cfg.setProperty("hibernate.dialect","org.hibernate.dialect.OracleDialect");

cfg.addAnnotatedClass(Student1.**class**);

SessionFactory sf= cfg.buildSessionFactory();

Session s=sf.openSession();

**SchemaExport se=new SchemaExport(cfg);**

**se.create(true,true);**

}

**catch**(HibernateException e) {

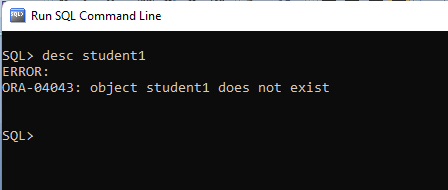
System.***out***.println(e.getMessage());

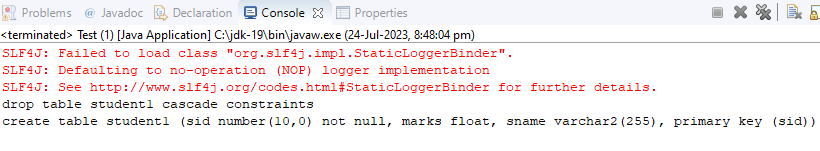
}

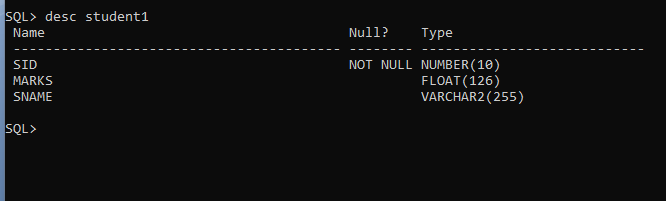
}

}

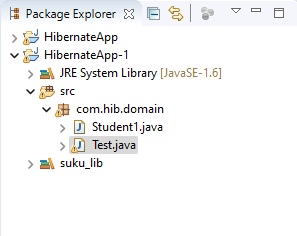
**Output:-**







Example:2 In this Table existed.



**Domaiclass:Student1.class**

**package** com.hib.domain;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

@Entity

@Table(name="student1")

**public** **class** Student1 {

@Id

@Column(name="sid" **,length=7)**

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

@Column(name="sname" **,length=15)**

**private** String sname;

@Column(name="marks")

**private** **float** marks;

**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;

}

**public** **float** getMarks() {

**return** marks;

}

**public** **void** setMarks(**float** marks) {

**this**.marks = marks;

}

}

**Client App:test.java**

**package** com.hib.domain;

**import** org.hibernate.HibernateException;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

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

**import** org.hibernate.tool.hbm2ddl.SchemaExport;

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

**public** **static** **void** main(String[] args) {

**try** {

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

cfg.setProperty("hibernate.connection.driver\_class","oracle.jdbc.OracleDriver");

cfg.setProperty("hibernate.connection.url","jdbc:oracle:thin:@localhost:1521:xe");

cfg.setProperty("hibernate.connection.username","system");

cfg.setProperty("hibernate.connection.password","tiger");

cfg.setProperty("hibernate.dialect","org.hibernate.dialect.OracleDialect");

cfg.addAnnotatedClass(Student1.**class**);

SessionFactory sf= cfg.buildSessionFactory();

Session s=sf.openSession();

SchemaExport se=**new** SchemaExport(cfg);

se.create(**true**,**true**);

}

**catch**(HibernateException e) {

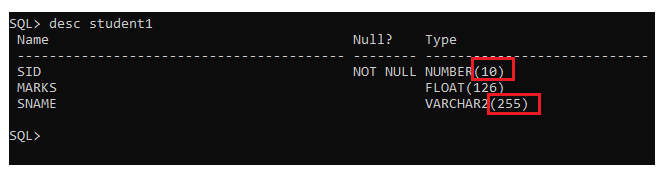
System.***out***.println(e.getMessage());

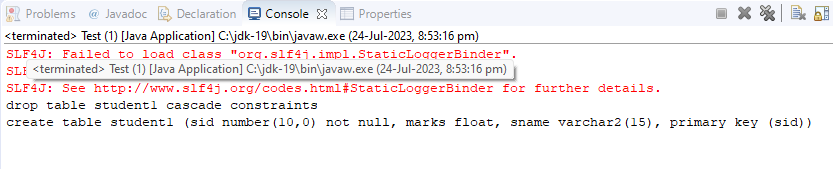
}

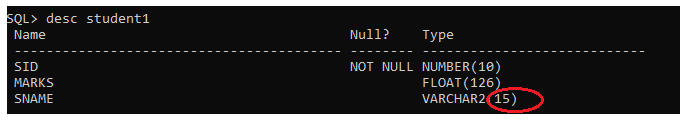
}

}

Output:-







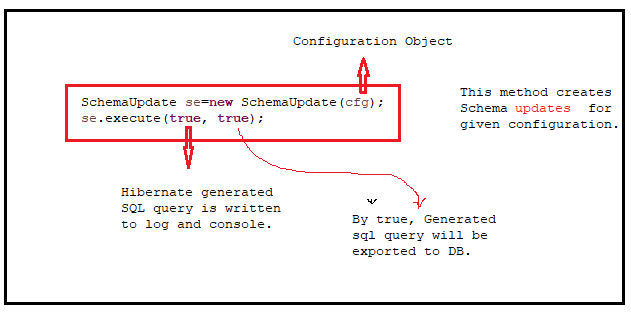
**2.Schema Update:-** The Basic O-R mapping details present in Configuration object. From configuration object, schema update tool gets required details(table name, coloumns names, column size, column data types …etc). Schema update tool checks whether tables exists with same name or not in DB.

If table does not exist with same name, it will create a new table with mapping detailes.

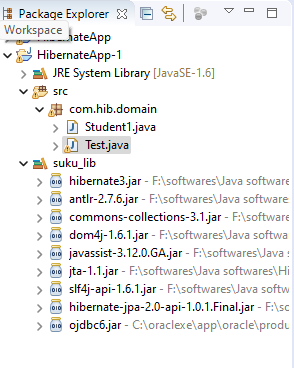
If table exist with same name, it will modify the only table structure in following case:

Case1: If Mapping details have more no.of columns when compared with database table existed in DB then only schema update tool modifies the table structure and it does not remove table records.

Case2:If mapping details have less no.of columns when compared with database table existed in DB then schema update tool will not modifies the table structure.



Example to Case-1:



Domain Class: Student1.class

**package** com.hib.domain;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

@Entity

@Table(name="student1")

**public** **class** Student1 {

@Id

@Column(name="sid" ,length=7)

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

@Column(name="sname" ,length=15)

**private** String sname;

/\*

\* @Column(name="marks") private float marks;

\*/

**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;

}

/\*

\* public float getMarks() { return marks; } public void setMarks(float marks) {

\* this.marks = marks; }

\*/

}

**Client Application:Test.java**

**package** com.hib.domain;

**import** org.hibernate.HibernateException;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

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

**import** org.hibernate.tool.hbm2ddl.SchemaUpdate;

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

**public** **static** **void** main(String[] args) {

**try** {

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

cfg.setProperty("hibernate.connection.driver\_class","oracle.jdbc.OracleDriver");

cfg.setProperty("hibernate.connection.url","jdbc:oracle:thin:@localhost:1521:xe");

cfg.setProperty("hibernate.connection.username","system");

cfg.setProperty("hibernate.connection.password","tiger");

cfg.setProperty("hibernate.dialect","org.hibernate.dialect.OracleDialect");

cfg.addAnnotatedClass(Student1.**class**);

SessionFactory sf= cfg.buildSessionFactory();

Session s=sf.openSession();

**SchemaUpdate se=new SchemaUpdate(cfg);**

**se.execute(true, true);**

}

**catch**(HibernateException e) {

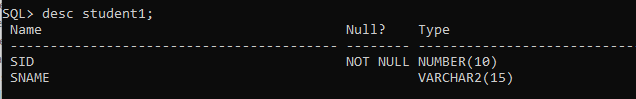
System.***out***.println(e.getMessage());

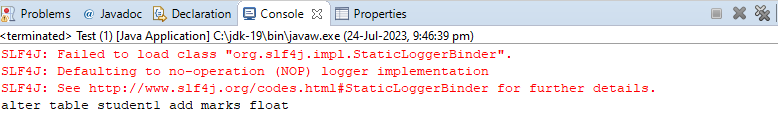
}

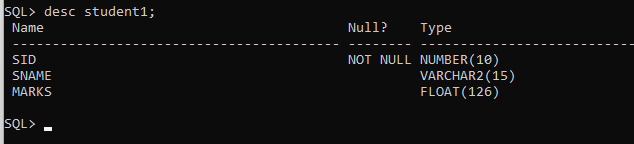
}

}

Output:-





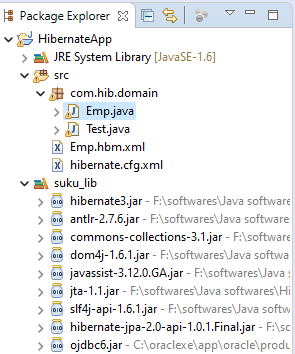


**3.hbm2ddl.auto:-**

The hibernate.hbm2ddl.auto configuration property is used to customize the Hibernate database schema generation process, and it can take the following values:

* none – This option disables the hbm2ddl.auto tool, so Hibernate is not going to take any action for managing the underlying database schema.
* create-only – This option instructs Hibernate to generate the database schema from the entity model.
* drop – This option instructs Hibernate to drop the database schema using the entity model as a reference for the DDL DROP statements.
* create – This option instructs Hibernate to drop the database schema and recreate it afterward using the entity model as a reference.
* create-drop – This option instructs Hibernate to drop the database schema and recreate it afterward using the entity model as a reference. When sessionfactory object created ,immediately table will be created. **When sessionfactory object is close, immediately table will be deleted from DB.**
* validate – This option instructs Hibernate to validate the underlying database schema against the entity mappings.
* update – This option instructs Hibernate to update the database schema by comparing the existing schema with the entity mappings and generate the appropriate schema migration scripts.

Example :1 create a table(Emp)



Configuration file:hibernate.cfg.xml

<?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"*>create</property>

<mapping resource=*"Emp.hbm.xml"*/>

</session-factory>

</hibernate-configuration>

Mapping File:Emp.hbm.xml

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

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 3.0//EN"

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

<hibernate-mapping>

<class name=*"com.hib.domain.Emp"* table=*"emp"*>

<id name=*"eid"* column=*"eid"* type=*"int"*/>

<property name=*"ename"* column=*"ename"* type=*"string"*/>

<property name=*"sal"* column=*"sal"* type=*"float"*/>

</class>

</hibernate-mapping>

Domain Class: Emp.java

**package** com.hib.domain;

**import** java.io.Serializable;

**public** **class** Emp **implements** Serializable {

**private** **int** eid;

**private** String ename;

**private** **float** sal;

**public** Emp() {

}

**public** **int** getEid() {

**return** eid;

}

**public** **void** setEid(**int** eid) {

**this**.eid = eid;

}

**public** String getEname() {

**return** ename;

}

**public** **void** setEname(String ename) {

**this**.ename = ename;

}

**public** **float** getSal() {

**return** sal;

}

**public** **void** setSal(**float** sal) {

**this**.sal = sal;

}

}

**ClientAPP: Test.java**

package com.hib.domain;

import java.util.Scanner;

import org.hibernate.HibernateException;

import org.hibernate.MappingException;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;

public class Test {

public static void main(String[] args) {

try {

Configuration cfg=new Configuration();

cfg.configure("hibernate.cfg.xml");

SessionFactory sf=cfg.buildSessionFactory();

Session sv=sf.openSession();

}

catch(HibernateException e) {

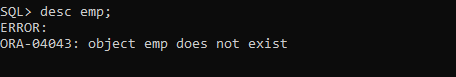
System.out.println(e.getMessage());

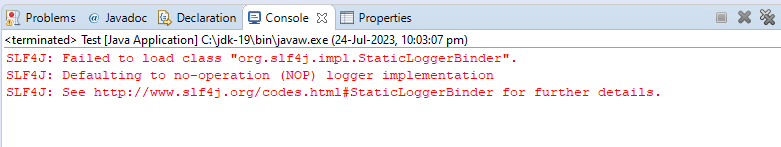
}

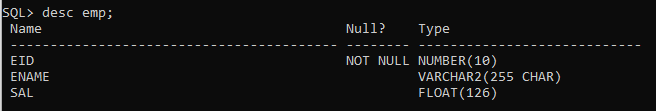
}

}

**Output:**

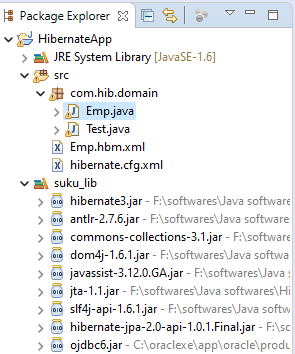
****

****

****

**Example:2 Drop the table**

Example :1 create a table(Emp)



Configuration file:hibernate.cfg.xml

<?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"*>drop</property>**

**<mapping resource=*"Emp.hbm.xml"*/>**

</session-factory>

</hibernate-configuration>

Mapping File:Emp.hbm.xml

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

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 3.0//EN"

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

<hibernate-mapping>

<class name=*"com.hib.domain.Emp"* table=*"emp"*>

<id name=*"eid"* column=*"eid"* type=*"int"*/>

<property name=*"ename"* column=*"ename"* type=*"string"*/>

<property name=*"sal"* column=*"sal"* type=*"float"*/>

</class>

</hibernate-mapping>

Domain Class: Emp.java

**package** com.hib.domain;

**import** java.io.Serializable;

**public** **class** Emp **implements** Serializable {

**private** **int** eid;

**private** String ename;

**private** **float** sal;

**public** Emp() {

}

**public** **int** getEid() {

**return** eid;

}

**public** **void** setEid(**int** eid) {

**this**.eid = eid;

}

**public** String getEname() {

**return** ename;

}

**public** **void** setEname(String ename) {

**this**.ename = ename;

}

**public** **float** getSal() {

**return** sal;

}

**public** **void** setSal(**float** sal) {

**this**.sal = sal;

}

}

**ClientAPP: Test.java**

package com.hib.domain;

import java.util.Scanner;

import org.hibernate.HibernateException;

import org.hibernate.MappingException;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;

public class Test {

public static void main(String[] args) {

try {

Configuration cfg=new Configuration();

cfg.configure("hibernate.cfg.xml");

SessionFactory sf=cfg.buildSessionFactory();

Session sv=sf.openSession();

}

catch(HibernateException e) {

System.out.println(e.getMessage());

}

}

}

**Output:**