## Hibernate Annotation-

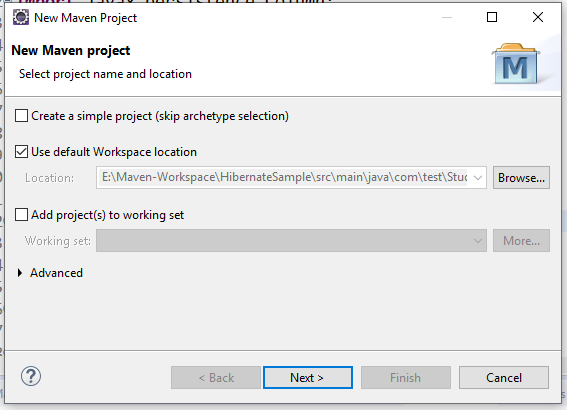
We can create hibernate application with annotation. There are many annotation that can be used to create the hibernate application such as @Entity, @Id, @Table, etc.

Package for annotation is javax.persistence.\*; Why?

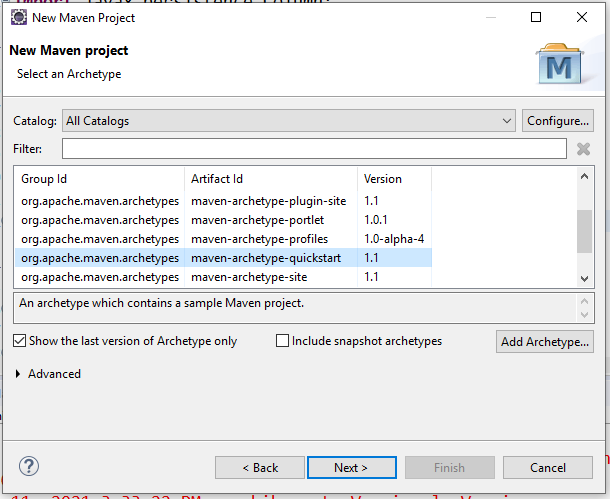
You don’t need to create mapping (hbm) file.

CRUD operation in hibernate **Hibernate- Insert operation** Maven project

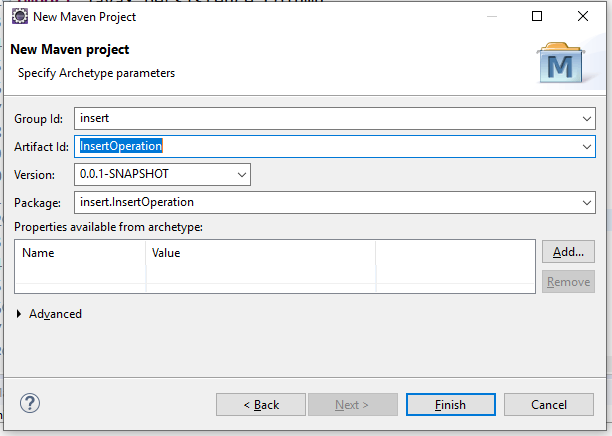
File->New->Maven Project->



Click on create simple project Click on Next button



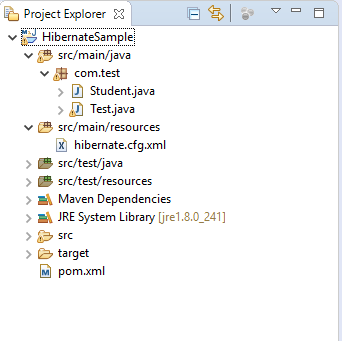
Click on Next button



Mention the group id as any name, artifact id as any name. Where group id is the project id and artifact id is project name.

Click on finish button.

Maven project structure looks like as



Go to pom.xml add hibernate and MySQL jar dependencies in that file

# <dependencies>

<dependency>

# <groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

# <version>4.1.4.Final</version>

</dependency>

# <dependency>

<groupId>mysql</groupId>

# <artifactId>mysql-connector-java</artifactId>

<version>8.0.21</version>

# </dependency>

</dependencies>

Create the student class

**package** com.test;

**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 {

# @Id

@Column(name = "id")

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

**private int** id; @Column(name = "name") **private** String name; @Column(name = "city") **private** String city; @Column(name = "mobile") **private** String mobile;

**public int** getId() {

**return** id;

# }

**public void** setId(**int** id) {

**this**.id = id;

# }

**public** String getName() {

**return** name;

# }

**public void** setName(String name) {

**this**.name = name;

# }

**public** String getCity() {

**return** city;

# }

**public void** setCity(String city) {

**this**.city = city;

# }

**public** String getMobile() {

**return** mobile;

# }

**public void** setMobile(String mobile) {

# **this**.mobile = mobile;

}

# }

Create the hibernate.cfg.xml file

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

# <!DOCTYPE hibernate-configuration SYSTEM "classpath://org/hibernate/hibernate-configuration-

3.0.dtd">

# <hibernate-configuration>

<session-factory>

<property name=*"hibernate.connection.driver\_class"*>com.mysql.jdbc.Driv er</property>

<property name=*"hibernate.connection.url"*>jdbc:mysql://localhost:3306/ test</property>

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

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

<property name=*"hibernate.dialect"*>org.hibernate.dialect.MySQLDialect<

# /property>

<property name=*"hbm2ddl.auto"*>create</property>

<property name=*"show\_sql"*>true</property>

<mapping class=*"com.test.Student"*></mapping>

# </session-factory>

</hibernate-configuration>

Create the main class as Test file- package com.test;

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) {

Configuration cfg = new Configuration(); cfg.configure("hibernate.cfg.xml");

SessionFactory sessionFactory = cfg.buildSessionFactory(); Session session = sessionFactory.openSession(); Transaction t = session.beginTransaction();

// insert data into database Student student = new Student(); student.setName("ram"); student.setCity("pune"); student.setMobile("9595972678"); session.save(student); t.commit();

session.close();

System.out.println("Record saved successfully.");

}

}

Run the application

Below message is displayed on screen is Hibernate: drop table if exists student

Hibernate: create table student (id integer not null auto\_increment, city varchar(255), mobile varchar(255), name varchar(255), primary key (id))

INFO: HHH000230: Schema export complete

Hibernate: insert into student (city, mobile, name) values (?, ?, ?) Record saved successfully.

Go to database and check the results whether record is inserted or not.

## Hibernate- Update operation

Create the student class

**package** com.test;

**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 {

# @Id

@Column(name = "id")

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

**private int** id; @Column(name = "name") **private** String name; @Column(name = "city") **private** String city;

# @Column(name = "mobile")

**private** String mobile;

**public int** getId() {

**return** id;

# }

**public void** setId(**int** id) {

**this**.id = id;

# }

**public** String getName() {

**return** name;

# }

**public void** setName(String name) {

**this**.name = name;

# }

**public** String getCity() {

**return** city;

# }

**public void** setCity(String city) {

**this**.city = city;

# }

**public** String getMobile() {

**return** mobile;

# }

**public void** setMobile(String mobile) {

# **this**.mobile = mobile;

}

# }

Create the hibernate.cfg.xml file

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

# <!DOCTYPE hibernate-configuration SYSTEM "classpath://org/hibernate/hibernate-configuration-

3.0.dtd">

# <hibernate-configuration>

<session-factory>

<property name=*"hibernate.connection.driver\_class"*>com.mysql.jdbc.Driv er</property>

<property name=*"hibernate.connection.url"*>jdbc:mysql://localhost:3306/ test</property>

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

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

<property name=*"hibernate.dialect"*>org.hibernate.dialect.MySQLDialect<

# /property>

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

<property name=*"show\_sql"*>true</property>

<mapping class=*"com.test.Student"*></mapping>

# </session-factory>

</hibernate-configuration>

Create the test class as main class. package com.test;

import org.hibernate.Session;

import org.hibernate.SessionFactory; import org.hibernate.Transaction; import org.hibernate.cfg.Configuration; public class Test { //update the record

public static void main(String[] args) { Configuration cfg = new Configuration(); cfg.configure("hibernate.cfg.xml");

SessionFactory sessionFactory = cfg.buildSessionFactory(); Session session = sessionFactory.openSession();

session.beginTransaction();

//pass the class name and id for updating record

Student student = (Student)session.get(Student.class, 1); student.setName("jack");

session.update(student); session.getTransaction().commit(); session.close(); sessionFactory.close();

System.out.println("Record updated successfully.");

}

}

Run the application

Hibernate: select student0\_.id as id0\_0\_, student0\_.city as city0\_0\_, student0\_.mobile as mobile0\_0\_, student0\_.name as name0\_0\_ from student student0\_ where student0\_.id=?

Hibernate: update student set city=?, mobile=?, name=? where id=?

org.hibernate.service.jdbc.connections.internal.DriverManagerConnectionProviderI mpl stop

INFO: HHH000030: Cleaning up connection pool [jdbc:mysql://localhost:3306/test] Record updated successfully.

Go to database and check the results. **Hibernate- Delete operation** Create the student class

**package** com.test;

**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 {

# @Id

@Column(name = "id")

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

**private int** id; @Column(name = "name") **private** String name; @Column(name = "city") **private** String city; @Column(name = "mobile") **private** String mobile;

**public int** getId() {

**return** id;

# }

**public void** setId(**int** id) {

**this**.id = id;

# }

**public** String getName() {

**return** name;

# }

**public void** setName(String name) {

**this**.name = name;

# }

**public** String getCity() {

**return** city;

# }

**public void** setCity(String city) {

**this**.city = city;

# }

**public** String getMobile() {

**return** mobile;

# }

**public void** setMobile(String mobile) {

# **this**.mobile = mobile;

}

# }

Create the hibernate.cfg.xml file

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

# <!DOCTYPE hibernate-configuration SYSTEM "classpath://org/hibernate/hibernate-configuration-

3.0.dtd">

# <hibernate-configuration>

<session-factory>

<property name=*"hibernate.connection.driver\_class"*>com.mysql.jdbc.Driv er</property>

<property name=*"hibernate.connection.url"*>jdbc:mysql://localhost:3306/ test</property>

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

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

<property name=*"hibernate.dialect"*>org.hibernate.dialect.MySQLDialect<

# /property>

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

<property name=*"show\_sql"*>true</property>

<mapping class=*"com.test.Student"*></mapping>

# </session-factory>

</hibernate-configuration>

Create the test class as main class. package com.test;

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) {

Configuration cfg = new Configuration(); cfg.configure("hibernate.cfg.xml");

SessionFactory sessionFactory = cfg.buildSessionFactory(); Session session = sessionFactory.openSession(); session.beginTransaction();

//pass the class name and id for delete record

//we use load or get () method to get the id from database. Student student = (Student)session.load(Student.class, 1); session.delete(student); session.getTransaction().commit();

session.close(); sessionFactory.close();

System.out.println("Record deleted successfully.");

}

}

Run the application

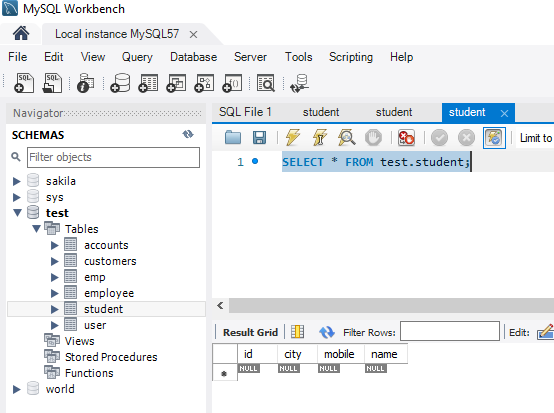
Hibernate: select student0\_.id as id0\_0\_, student0\_.city as city0\_0\_, student0\_.mobile as mobile0\_0\_, student0\_.name as name0\_0\_ from student student0\_ where student0\_.id=?

Hibernate: delete from student where id=?

org.hibernate.service.jdbc.connections.internal.DriverManagerConnectionProviderI mpl stop

INFO: HHH000030: Cleaning up connection pool [jdbc:mysql://localhost:3306/test] Record deleted successfully.

Go to database and check the results.



## Hibernate- Select operation

Create the student class

**package** com.test;

**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 {

# @Id

@Column(name = "id")

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

**private int** id; @Column(name = "name") **private** String name; @Column(name = "city") **private** String city; @Column(name = "mobile") **private** String mobile;

**public int** getId() {

**return** id;

# }

**public void** setId(**int** id) {

**this**.id = id;

# }

**public** String getName() {

**return** name;

# }

**public void** setName(String name) {

**this**.name = name;

# }

**public** String getCity() {

**return** city;

# }

**public void** setCity(String city) {

**this**.city = city;

# }

**public** String getMobile() {

**return** mobile;

# }

**public void** setMobile(String mobile) {

# **this**.mobile = mobile;

}

# }

Create the hibernate.cfg.xml file

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

# <!DOCTYPE hibernate-configuration SYSTEM "classpath://org/hibernate/hibernate-configuration-

3.0.dtd">

# <hibernate-configuration>

<session-factory>

<property name=*"hibernate.connection.driver\_class"*>com.mysql.jdbc.Driv er</property>

<property name=*"hibernate.connection.url"*>jdbc:mysql://localhost:3306/ test</property>

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

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

<property name=*"hibernate.dialect"*>org.hibernate.dialect.MySQLDialect<

# /property>

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

<property name=*"show\_sql"*>true</property>

<mapping class=*"com.test.Student"*></mapping>

# </session-factory>

</hibernate-configuration>

Create the test class as main class.

**package** com.test;

**import** java.util.\*;

**import** org.hibernate.Query;

**import** org.hibernate.Session;

# **import** org.hibernate.SessionFactory;

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

**public class** Test {

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

# Configuration cfg = **new** Configuration(); cfg.configure("hibernate.cfg.xml"); SessionFactory sessionFactory =

cfg.~~buildSessionFactory~~();

# Session session = sessionFactory.openSession(); session.beginTransaction();

// pass the class name

# Query query = session.createQuery("from Student"); List<Student>students = query.list();

**for** (Student student : students) { System.***out***.println("ID="+student.getId()); System.***out***.println("City="+student.getCity());

# System.***out***.println("Mobile="+student.getMobile()); System.***out***.println("Name="+student.getMobile());

}

# session.getTransaction().commit(); session.close(); sessionFactory.close();

System.***out***.println("Record retrieved successfully.");

# }

}

Run the application Output

Hibernate: select student0\_.id as id0\_, student0\_.city as city0\_, student0\_.mobile as mobile0\_, student0\_.name as name0\_ from student student0\_