CRUD Operations Using Hibernate (Annotation and Configuration)

In this section, you will learn how to develop a CRUD application using hibernate annotation.  
**Follows the following steps for developing the CRUD application in hibernate annotation.**  
**Step 1: Create Domain Entity Class**

**Student.java**

package com.sdnext.hibernate.tutorial.dto;

import java.io.Serializable;

import javax.persistence.Column;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Entity

@Table(name="STUDENT")

public class Student implements Serializable

{

/\*\*

\* serialVersionUID

\*/

private static final long serialVersionUID = 8633415090390966715L;

@Id

@Column(name="ID")

@GeneratedValue(strategy=GenerationType.AUTO)

private int id;

@Column(name="STUDENT\_NAME")

private String studentName;

@Column(name="ROLL\_NUMBER")

private int rollNumber;

@Column(name="COURSE")

private String course;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getStudentName() {

return studentName;

}

public void setStudentName(String studentName) {

this.studentName = studentName;

}

public int getRollNumber() {

return rollNumber;

}

public void setRollNumber(int rollNumber) {

this.rollNumber = rollNumber;

}

public String getCourse() {

return course;

}

public void setCourse(String course) {

this.course = course;

}

}

**Step 2: Create Hibernate Configuration file**  
**hibernate.cfg.xml**  
file contains  
(a.) **database connection setting** (database driver (com.mysql.jdbc.Driver), url (jdbc:mysql://localhost:3306/hibernateDB2), username (root) and password (root)),  
(b.) **SQL dialect** (dialect – org.hibernate.dialect.MySQLDialect),  
(c.) **enable hibernate’s automatic session context management**(current\_session\_context\_class – thread),  
(d.) **disable the second level cache**(cache.provider\_class – org.hibernate.cache.NoCacheProvider),  
(e.) **print all executed SQL to stdout** (show\_sql – true) and  
(f.) **drop and re-create the database schema on startup** (hbm2ddl.auto – none).

<hibernate-configuration>

<session-factory>

<!-- Database connection settings -->

<property name="connection.driver\_class">com.mysql.jdbc.Driver</property>

<property name="connection.url">jdbc:mysql://localhost:3306/hibernateDB2</property>

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

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

<!-- JDBC connection pool (use the built-in) -->

<property name="connection.pool\_size">1</property>

<!-- SQL dialect -->

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

<!-- Enable Hibernate's automatic session context management -->

<property name="current\_session\_context\_class">thread</property>

<!-- Disable the second-level cache -->

<property name="cache.provider\_class">org.hibernate.cache.NoCacheProvider</property>

<!-- Echo all executed SQL to stdout -->

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

<!-- Drop and re-create the database schema on startup -->

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

<mapping class="com.sdnext.hibernate.tutorial.dto.Student">

</mapping></session-factory>

</hibernate-configuration>

**Step 3: Create Hibernate Utility Class**  
**HibernateUtil.java**

package com.sdnext.hibernate.tutorial.utility;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.AnnotationConfiguration;

public class HibernateUtil

{

private static final SessionFactory sessionFactory;

static

{

try

{

sessionFactory = new AnnotationConfiguration().configure().buildSessionFactory();

}

catch(Throwable th){

System.err.println("Enitial SessionFactory creation failed"+th);

throw new ExceptionInInitializerError(th);

}

}

public static SessionFactory getSessionFactory(){

return sessionFactory;

}

}

**Step 4: Create Student on the database.**  
**CreateStudent.java**

package com.sdnext.hibernate.tutorial;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import com.sdnext.hibernate.tutorial.dto.Student;

import com.sdnext.hibernate.tutorial.utility.HibernateUtil;

public class CreateStudent {

/\*\*

\* @param args

\*/

public static void main(String[] args)

{

//Create student entity object

Student student = new Student();

student.setStudentName("Dinesh Rajput");

student.setRollNumber(01);

student.setCourse("MCA");

//Create session factory object

SessionFactory sessionFactory = HibernateUtil.getSessionFactory();

//getting session object from session factory

Session session = sessionFactory.openSession();

//getting transaction object from session object

session.beginTransaction();

session.save(student);

System.out.println("Inserted Successfully");

session.getTransaction().commit();

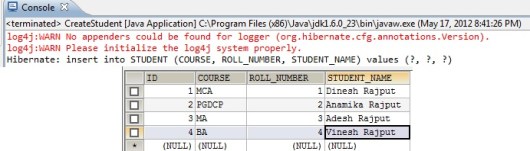
session.close();

sessionFactory.close();

}

}

**OUTPUT:**  
log4j:WARN No appenders could be found for logger (org.hibernate.cfg.annotations.Version).  
log4j:WARN Please initialize the log4j system properly.  
Hibernate: insert into STUDENT (COURSE, ROLL\_NUMBER, STUDENT\_NAME) values (?, ?, ?)  
**Inserted Successfully**



Now the following code the reading the student data from database.  
**Step 5: Reading the Student data from the database table STUDENT**  
**ReadStudent.java**

package com.sdnext.hibernate.tutorial;

import java.util.List;

import org.hibernate.Query;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import com.sdnext.hibernate.tutorial.dto.Student;

import com.sdnext.hibernate.tutorial.utility.HibernateUtil;

public class ReadStudent {

/\*\*

\* @param args

\*/

public static void main(String[] args)

{

//Create session factory object

SessionFactory sessionFactory = HibernateUtil.getSessionFactory();

//getting session object from session factory

Session session = sessionFactory.openSession();

//getting transaction object from session object

session.beginTransaction();

Query query = session.createQuery("from Student");

List students = query.list();

for(Student student : students)

{

System.out.println("Roll Number: "+student.getRollNumber()+", Student Name: "+student.getStudentName()+", Course: "+student.getCourse());

}

session.getTransaction().commit();

sessionFactory.close();

}

}

**Output:**

log4j:WARN No appenders could be found for logger (org.hibernate.cfg.annotations.Version).  
log4j:WARN Please initialize the log4j system properly.  
Hibernate: select student0\_.ID as ID0\_, student0\_.COURSE as COURSE0\_, student0\_.ROLL\_NUMBER as ROLL3\_0\_, student0\_.STUDENT\_NAME as STUDENT4\_0\_ from STUDENT student0\_  
Roll Number: 1, Student Name: Dinesh Rajput, Course: MCA  
Roll Number: 2, Student Name: Anamika Rajput, Course: PGDCP  
Roll Number: 3, Student Name: Adesh Rajput, Course: MA  
Roll Number: 4, Student Name: Vinesh Rajput, Course: BA



The following code is for updating the data into the database table “STUDENT”.  
**Step 6: Update the Student Record in the Database.**  
**UpdateStudent.java**

package com.sdnext.hibernate.tutorial;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import com.sdnext.hibernate.tutorial.dto.Student;

import com.sdnext.hibernate.tutorial.utility.HibernateUtil;

public class UpdateStudent {

/\*\*

\* @param args

\*/

public static void main(String[] args)

{

//Create session factory object

SessionFactory sessionFactory = HibernateUtil.getSessionFactory();

//getting session object from session factory

Session session = sessionFactory.openSession();

//getting transaction object from session object

session.beginTransaction();

Student student = (Student)session.get(Student.class, 2);

student.setStudentName("Sweety Rajput");

System.out.println("Updated Successfully");

session.getTransaction().commit();

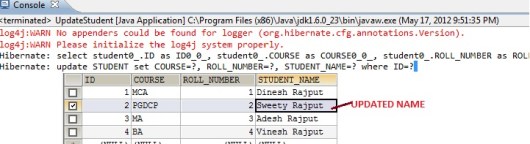
sessionFactory.close();

}

}

**Output:**

log4j:WARN No appenders could be found for logger (org.hibernate.cfg.annotations.Version).  
log4j:WARN Please initialize the log4j system properly.  
Hibernate: select student0\_.ID as ID0\_0\_, student0\_.COURSE as COURSE0\_0\_, student0\_.ROLL\_NUMBER as ROLL3\_0\_0\_, student0\_.STUDENT\_NAME as STUDENT4\_0\_0\_ from STUDENT student0\_ where student0\_.ID=?  
Hibernate: update STUDENT set COURSE=?, ROLL\_NUMBER=?, STUDENT\_NAME=? where ID=?  
**Updated Successfully**



**Step 7: Delete the student data from the database.**  
**DeleteStudent.java**

package com.sdnext.hibernate.tutorial;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import com.sdnext.hibernate.tutorial.dto.Student;

import com.sdnext.hibernate.tutorial.utility.HibernateUtil;

public class DeleteStudent {

/\*\*

\* @param args

\*/

public static void main(String[] args)

{

//Create session factory object

SessionFactory sessionFactory = HibernateUtil.getSessionFactory();

//getting session object from session factory

Session session = sessionFactory.openSession();

//getting transaction object from session object

session.beginTransaction();

Student student = (Student)session.load(Student.class, 4);

session.delete(student);

System.out.println("Deleted Successfully");

session.getTransaction().commit();

sessionFactory.close();

}

}

**Output:**

log4j:WARN No appenders could be found for logger (org.hibernate.cfg.annotations.Version).  
log4j:WARN Please initialize the log4j system properly.  
Hibernate: select student0\_.ID as ID0\_0\_, student0\_.COURSE as COURSE0\_0\_, student0\_.ROLL\_NUMBER as ROLL3\_0\_0\_, student0\_.STUDENT\_NAME as STUDENT4\_0\_0\_ from STUDENT student0\_ where student0\_.ID=?  
**Deleted Successfully**  
Hibernate: delete from STUDENT where ID=?

