Automatic Table Creation Using Hibernate

Last Updated : 23 Jul, 2025

Hibernate is a Java framework that implements **ORM**(Object Relational Mapping) design pattern. It is used to map java objects into a relational database. It internally uses **JDBC**(Java Database Connectivity), **JTA**(Java Transaction API), and **JNDI**(Java Naming and Directory Interface). It helps to make java objects persist in the database without losing their state, thus, named Hibernate. It can be used to perform all the CRUD operations without having to write SQL queries. Hibernate framework can be used to create tables in the database automatically. Below property is added in the configuration file to create tables automatically.

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

"hibernate.hbm2ddl.auto" property accepts the following values:

- **create**: If the value is created, Hibernate creates a new table in the database when the SessionFactory object is created. In case, a table exists in the database with the same name, it deletes the table along with data and creates a new table.
- **update**: If the value is updated then hibernate first validates whether the table is present in the database or not , if present alters that table as per the changes, if not creates a new one.
- **validate**: If the value is validated then hibernate only verifies whether the table is present or not if the table does not exist then throws an exception.
- **create-drop**: If the value is create-drop then hibernate creates a new table when SessionFactory is created, performs the operation, and

deletes the table when SessionFactory is destroyed. This value is used for testing the hibernate code.

- **none**: It does not make any changes to the schema.

Below is an example demonstrating the automatic table creation in hibernate using the MySQL database.

Example Project

hibernate.cfg.xml file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate Configuration DTD 3.0//EN"
             "https://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="hibernate.hbm2ddl.auto">create</property>
        <property name="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</property>
        <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/demo</property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password">root</property>
        <mapping resource="Student.hbm.xml"></mapping>
    </session-factory>
</hibernate-configuration>
```

Student.hbm.xml file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate Mapping DTD 3.0//EN"
            "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="p1.Student" table="student">
    <id name="id"></id>
    <property name="name"></property>
    </class>
</hibernate-mapping>
```

Student.java file:

```
package p1;
```

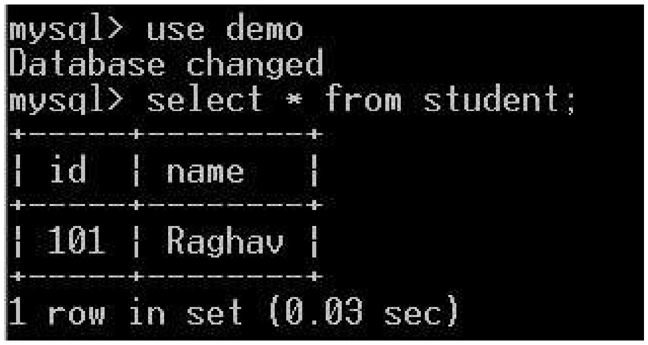
```
public class Student {

    private int id;
    private String name;

    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;
    }

}
```

Test.java file:

```
package p1;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

import beans.Student;

public class Test {
    public static void main(String... args) {
        try
        {
            Configuration config=new Configuration();
            config.configure();
            System.out.println(config);
            SessionFactory sessionFactory=config.buildSessionFactory();
            Session session=sessionFactory.openSession();
            System.out.println(session);
            Student s=new Student();
            s.setId(101);
            s.setName("Raghav");
            session.save(s);
            Transaction t=session.beginTransaction();
            t.commit();
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
}
```

Output:

A new table 'student' is created and data from the Student class object is mapped into the table.



```
mysql> use demo
Database changed
mysql> select * from student;
+-----+--------+
| id  | name   |
+-----+--------+
| 101 | Raghav |
+-----+--------+
1 row in set (0.03 sec)
```