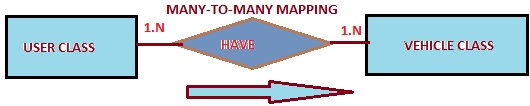
Many to Many Mapping in Hibernate

Many to many mapping in hibernate is required when each record in an entity may have many linked records in another entity and vice-versa.

In this examples you will learn how to map **many-to-many** relationship using Hibernate. Consider the following relationship between *Vehicle*and *UserDetails* entity.



According to the relationship a user can have in any number of vehicles and the vehicle can have any number of users.

**UserDetail.java**

package com.test.hibernate.examples.dto;

import java.util.ArrayList;

import java.util.Collection;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.ManyToMany;

import javax.persistence.Table;

**@Entity**

**@Table (name="USER")**

public class UserDetails

{

**@Id**

**@Column(name="USER\_ID")**

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

    private int    userId;

**@Column(name="USER\_NAME")**

    private String userName;

**@ManyToMany**

    private Collection<Vehicle> vehicle = new ArrayList<Vehicle>();

    public int getUserId() {

        return userId;

    }

    public Collection<Vehicle> getVehicle() {

        return vehicle;

    }

    public void setVehicle(Collection<Vehicle> vehicle) {

        this.vehicle = vehicle;

    }

    public void setUserId(int userId) {

        this.userId = userId;

    }

    public String getUserName() {

        return userName;

    }

    public void setUserName(String userName) {

        this.userName = userName;

    }

}

**Vehicle.java**

package com.test.hibernate.examples.dto;

import java.util.ArrayList;

import java.util.Collection;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.ManyToMany;

import javax.persistence.Table;

**@Entity**

**@Table(name="VEHICLE")**

public class Vehicle

{

**@Id**

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

**@Column(name="VEHICLE\_ID")**

    private int vehicleId;

**@Column(name="VEHICLE\_NAME")**

    private String vehicleName;

**@ManyToMany(mappedBy="vehicle")**

    private Collection<UserDetails> user = new ArrayList<UserDetails>();

    public Collection<UserDetails> getUser() {

        return user;

    }

    public void setUser(Collection<UserDetails> user) {

        this.user = user;

    }

    public int getVehicleId() {

        return vehicleId;

    }

    public void setVehicleId(int vehicleId) {

        this.vehicleId = vehicleId;

    }

    public String getVehicleName() {

        return vehicleName;

    }

    public void setVehicleName(String vehicleName) {

        this.vehicleName = vehicleName;

    }

}

**hibernate.cfg.xml**:

<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/hibernateDB</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">create</property>

<mapping class="com.test.hibernate.examples.dto.UserDetails">

<mapping class="com.test.hibernate.examples.dto.Vehicle">

</mapping></mapping></session-factory>

</hibernate-configuration>

**HibernateTestDemo.java**

**package com.test.hibernate.examples;**

**import org.hibernate.Session;**

**import org.hibernate.SessionFactory;**

**import org.hibernate.cfg.AnnotationConfiguration;**

**import com.test.hibernate.examples.dto.UserDetails;**

**import com.test.hibernate.examples.dto.Vehicle;**

**public class HibernateTestDemo {**

**/\*\***

**\* @param args**

**\*/**

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

**{**

**UserDetails user = new UserDetails();**

**UserDetails user2 = new UserDetails();**

**Vehicle vehicle = new Vehicle();**

**Vehicle vehicle2 = new Vehicle();**

**vehicle.setVehicleName("Car");**

**vehicle.getUser().add(user);**

**vehicle.getUser().add(user2);**

**vehicle2.setVehicleName("Jeep");**

**vehicle2.getUser().add(user2);**

**vehicle2.getUser().add(user);**

**user.setUserName("First User");**

**user2.setUserName("Second User");**

**user.getVehicle().add(vehicle);**

**user.getVehicle().add(vehicle2);**

**user2.getVehicle().add(vehicle);**

**user2.getVehicle().add(vehicle2);**

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

**Session session = sessionFactory.openSession();**

**session.beginTransaction();**

**session.save(vehicle);**

**session.save(vehicle2);**

**session.save(user);**

**session.save(user2);**

**session.getTransaction().commit();**

**session.close();**

**}**

**}**

**Using Mapping files instead of Annotation**  
 **userDetail.hbm.xml**

<hibernate-mapping>

<class name="com.test.hibernate.examples.dto.UserDetails" table="USER">

<id column="ID" name="userId" type="long">

<generator class="assigned">

</generator></id>

<property column="column" name="UserName">

<list cascade="all" name="vehicle" table="USER\_VEHICLE">

<key column="USER\_ID">

<many-to-many class="com.test.hibernate.examples.dto.Vehicle" column="VEHICLE\_ID">

</many-to-many></key></list>

</property></class>

</hibernate-mapping>

**vehicle.hbm.xml**

<hibernate-mapping>

<class name="com.test.hibernate.examples.dto.Vehicle" table="VEHICLE">

<id column="ID" name="userId" type="long">

<generator class="assigned">

</generator></id>

<property column="VEHICLE\_NAME" name="vehicleName">

</property></class>

</hibernate-mapping>