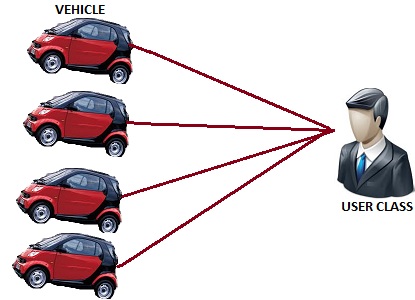
One to Many Mapping in Hibernate Example

In [previous tutorial](https://www.dineshonjava.com/p/hibernate-one-to-one-mapping-tutorial.html) we look that what is One To One Mapping and also discussed some examples about that.

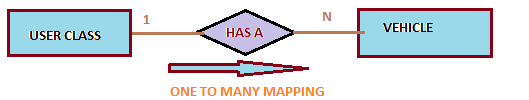
In this tutorial of one to many mapping in hibernate example we will discuss about the One To Many Mapping. A one-to-many relationship occurs when one entity is related to many occurrences in another entity.

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

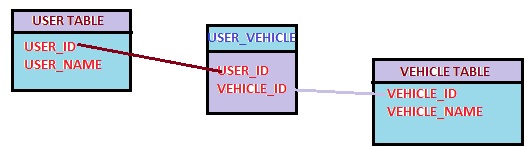
One user have the multiple vehicles.



Class diagram for that given below.



In this example **UserDetails**class has the collection of the another entity class **Vehicle**. So the given below diagram so table structure for that.



For that we will use the following annotation.

**@OneToMany:**

**Target:**  
Fields (including property get methods)Defines a many-valued association with one-to-many multiplicity. If the collection is defined using generics to specify the element type, the associated target entity type need not be specified; otherwise the target entity class must be specified. If the relationship is bidirectional, the**mappedBy**element must be used to specify the relationship field or property of the entity that is the owner of the relationship. The **OneToMany**annotation may be used within an **embeddable**class contained within an entity class to specify a relationship to a collection of entities. If the relationship is bidirectional, the **mappedBy**element must be used to specify the relationship field or property of the entity that is the owner of the relationship.

Now we look the following Example related to the **One to Many** mapping.

1. First Create Vehicle Class  
**Vehicle.java**

package com.sdnext.hibernate.tutorial.dto;

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=”VEHICLE”)

public class Vehicle

{

@Id

@GeneratedValue(strategy=GenerationType.AUTO)

@Column(name=”VEHICLE\_ID”)

private int vehicleId;

@Column(name=”VEHICLE\_NAME”)

private String vehicleName;

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;

}

}

2. Create the User Class  
**UserDetails.java**

package com.sdnext.hibernate.tutorial.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.JoinColumn;

import javax.persistence.JoinTable;

import javax.persistence.OneToMany;

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;

@OneToMany

@JoinTable( name=”USER\_VEHICLE”,

joinColumns=@JoinColumn(name=”USER\_ID”),

inverseJoinColumns=@JoinColumn(name=”VEHICLE\_ID”)) //its optional using for name configuration of the join table

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;

}

}

3.Create the hibernate configuration file.  
**hibernate.cfg.xml**:

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

<!DOCTYPE hibernate-configuration PUBLIC

“-//Hibernate/Hibernate Configuration DTD 3.0//EN”

“http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd”>

<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=*”hibernate.dialect”*>org.hibernate.dialect.MySQL5Dialect</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.sdnext.hibernate.tutorial.dto.UserDetails”*/>

<mapping class=*”com.sdnext.hibernate.tutorial.dto.Vehicle”*/>

</session-factory>

</hibernate-configuration>

4. Create Test Demo class for run this code.  
**HibernateTestDemo.java**

package com.sdnext.hibernate.tutorial;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.AnnotationConfiguration;

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

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

public class HibernateTestDemo {

/\*\*

\* @param args

\*/

public static void main(String[] args)

{

UserDetails user = new UserDetails(); //create the user entity object

Vehicle vehicle = new Vehicle(); //create the first vehicle entity object

Vehicle vehicle2 = new Vehicle(); //create the second vehicle entity

vehicle.setVehicleName(“BMW Car”); //set the value to the vehicle entity

vehicle2.setVehicleName(“AUDI Car”);

user.setUserName(“Dinesh Rajput”); //Set the value to the user entity

user.getVehicle().add(vehicle); //add vehicle to the list of the vehicle

user.getVehicle().add(vehicle2);

SessionFactory sessionFactory = new AnnotationConfiguration().

configure().buildSessionFactory(); //create session factory object

Session session = sessionFactory.openSession(); //create the session object

session.beginTransaction(); //start the transaction of the session object

session.save(vehicle); //saving the vehicle to the database

session.save(vehicle2);

session.save(user); //save the user to the database

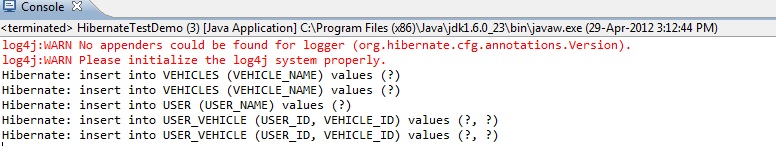
session.getTransaction().commit(); //close the transaction

session.close(); //close the session

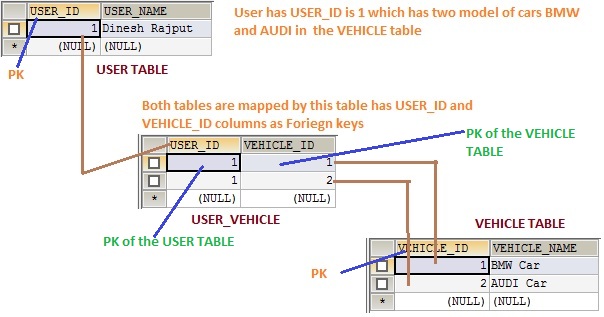
}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



Now we look the table structure about this example.



Now how can implement this mapping through mapping file(**.hbm.xml**) instead of the **annotations**.

For user class..  
**UserDetails.hbm.xml**

<?xml version=*”1.0″?*>

<!DOCTYPE hibernate-mapping PUBLIC

“-//Hibernate/Hibernate Mapping DTD 3.0//EN”

“http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd”>

<hibernate-mapping>

<class name=*”com.sdnext.hibernate.tutorial.dto.UserDetails”* table=*”USER”*>

<id name=*”userId”* type=*”long”* column=*”ID”* >

<generator class=*”assigned”*/>

</id>

<property name=*”userName”*>

<column name=*”UserName”* />

</property>

<list name=*”vehicle”* table=*”STUDENT\_VEHICLE”* cascade=*”all”*>

<key column=*”USER\_ID”* />

<many-to-many column=*”VEHICLE\_ID”* unique=*”true”* class=*”com.sdnext.hibernate.tutorial.dto.Vehicle”* />

</list>

</class>

</hibernate-mapping>

Mapping File For Vehicle  Class…  
**vehicle.hbm.xml**

<?xml version=*”1.0″?*>

<!DOCTYPE hibernate-mapping PUBLIC

“-//Hibernate/Hibernate Mapping DTD 3.0//EN”

“http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd”>

<hibernate-mapping>

<class name=*”com.sdnext.hibernate.tutorial.dto.Vehicle”* table=*”VEHICLE”*>

<id name=*”userId”* type=*”long”* column=*”ID”* >

<generator class=*”assigned”*/>

</id>

<property name=*”vehicleName”* column=*”VEHICLE\_NAME”*> </property>

</class>

</hibernate-mapping>

So it is brief description about the One To Many Mapping with using **annotation**and also using**.hbm.xml** files for entity class.