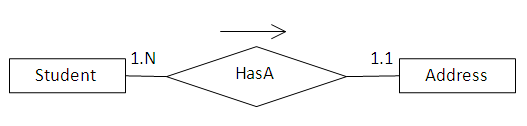
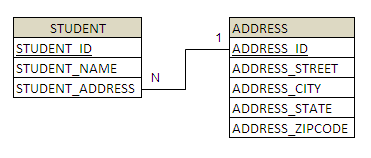
In this example you will learn how to map many-to-one relationship using Hibernate. Consider the following relationship between *Student* and *Address* entity.



According to the relationship many students can have the same address.

To create this relationship you need to have a *STUDENT* and *ADDRESS* table. The relational model is shown below.



To create the *STUDENT* and *ADDRESS* table you need to create the following hibernate mapping files.

*Student.hbm.xml* is used to create the *STUDENT* table.

01.<?xml version="1.0"?>

02.<!DOCTYPE hibernate-mapping PUBLIC

03."-//Hibernate/Hibernate Mapping DTD 3.0//EN"

04."<http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd>">

05.<hibernate-mapping>

06.<class name="com.vaannila.student.Student" table="STUDENT">

07.<meta attribute="class-description">This class contains student details.</meta>

08.<id name="studentId" type="long" column="STUDENT\_ID">

09.<generator class="native" />

10.</id>

11.<property name="studentName" type="string" length="100" not-null="true"column="STUDENT\_NAME" />

12.<many-to-one name="studentAddress" class="com.vaannila.student.Address"column="STUDENT\_ADDRESS" cascade="all" not-null="true" />

13.</class>

14.</hibernate-mapping>

The *many-to-one* element is used to create the many-to-one relationship between the *Student* and *Address*entities. The *cascade* option is used to cascade the required operations to the associated entity. If the *cascade*option is set to all then all the operations will be cascaded. For instance when you save a *Student* object, the associated *Address* object will also be saved automatically.

*Address.hbm.xml* is used to create the *ADDRESS* table.

[view source](http://www.dzone.com/tutorials/java/hibernate/hibernate-example/hibernate-mapping-many-to-one-1.html#viewSource)

[print](http://www.dzone.com/tutorials/java/hibernate/hibernate-example/hibernate-mapping-many-to-one-1.html#printSource)[?](http://www.dzone.com/tutorials/java/hibernate/hibernate-example/hibernate-mapping-many-to-one-1.html#about)

01.<?xml version="1.0"?>

02.<!DOCTYPE hibernate-mapping PUBLIC

03."-//Hibernate/Hibernate Mapping DTD 3.0//EN"

04."<http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd>">

05.<hibernate-mapping>

06.<class name="com.vaannila.student.Address" table="ADDRESS">

07.<meta attribute="class-description">This class contains the student's address

08.details.</meta>

09.<id name="addressId" type="long" column="ADDRESS\_ID">

10.<generator class="native" />

11.</id>

12.<property name="street" column="ADDRESS\_STREET" type="string"length="250" />

13.<property name="city" column="ADDRESS\_CITY" type="string" length="50" />

14.<property name="state" column="ADDRESS\_STATE" type="string" length="50"/>

15.<property name="zipcode" column="ADDRESS\_ZIPCODE" type="string"length="10" />

16.</class>

17.</hibernate-mapping>

Now create the hibernate configuration file and add all the mapping files.

[view source](http://www.dzone.com/tutorials/java/hibernate/hibernate-example/hibernate-mapping-many-to-one-1.html#viewSource)

[print](http://www.dzone.com/tutorials/java/hibernate/hibernate-example/hibernate-mapping-many-to-one-1.html#printSource)[?](http://www.dzone.com/tutorials/java/hibernate/hibernate-example/hibernate-mapping-many-to-one-1.html#about)

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

02.<!DOCTYPE hibernate-configuration PUBLIC

03."-//Hibernate/Hibernate Configuration DTD 3.0//EN"

04."<http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd>">

05.<hibernate-configuration>

06.<session-factory>

07.<property name="hibernate.connection.driver\_class"> org.hsqldb.jdbcDriver </property>

08.<property name="hibernate.connection.url"> jdbc:hsqldb:[hsql://localhost<](hsql://localhost%3c);/property>

09.<property name="hibernate.connection.username">sa</property>

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

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

12.<property name="hibernate.dialect"> org.hibernate.dialect.HSQLDialect </property>

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

14.<property name="hbm2ddl.auto">create-drop</property>

15.<mapping resource="com/vaannila/student/Student.hbm.xml"/>

16.<mapping resource="com/vaannila/student/Address.hbm.xml"/>

17.</session-factory>

18.</hibernate-configuration>

After creating the configuration file, generate java class files using Hibernate Tools.(To generate code using Hibernate Tools refer this [example](http://vaannila.com/hibernate/hibernate-example/hibernate-tools-1.html) )

The following classes will be generated.

[view source](http://www.dzone.com/tutorials/java/hibernate/hibernate-example/hibernate-mapping-many-to-one-1.html#viewSource)

[print](http://www.dzone.com/tutorials/java/hibernate/hibernate-example/hibernate-mapping-many-to-one-1.html#printSource)[?](http://www.dzone.com/tutorials/java/hibernate/hibernate-example/hibernate-mapping-many-to-one-1.html#about)

01.package com.vaannila.student;

02.

03.// Generated Sep 3, 2009 7:20:37 PM by Hibernate Tools 3.2.4.GA

04.

05./\*\*

06.\* This class contains student details.

07.\*/

08.public class Student implements java.io.Serializable {

09.

10.private long studentId;

11.private String studentName;

12.private Address studentAddress;

13.

14.public Student() {

15.}

16.

17.public Student(String studentName, Address studentAddress) {

18.this.studentName = studentName;

19.this.studentAddress = studentAddress;

20.}

21.

22.public long getStudentId() {

23.return this.studentId;

24.}

25.

26.public void setStudentId(long studentId) {

27.this.studentId = studentId;

28.}

29.

30.public String getStudentName() {

31.return this.studentName;

32.}

33.

34.public void setStudentName(String studentName) {

35.this.studentName = studentName;

36.}

37.

38.public Address getStudentAddress() {

39.return this.studentAddress;

40.}

41.

42.public void setStudentAddress(Address studentAddress) {

43.this.studentAddress = studentAddress;

44.}

45.

46.}

[view source](http://www.dzone.com/tutorials/java/hibernate/hibernate-example/hibernate-mapping-many-to-one-1.html#viewSource)

[print](http://www.dzone.com/tutorials/java/hibernate/hibernate-example/hibernate-mapping-many-to-one-1.html#printSource)[?](http://www.dzone.com/tutorials/java/hibernate/hibernate-example/hibernate-mapping-many-to-one-1.html#about)

01.package com.vaannila.student;

02.

03.// Generated Sep 3, 2009 7:20:37 PM by Hibernate Tools 3.2.4.GA

04.

05./\*\*

06.\* This class contains the student's address

07.\*          details.

08.\*/

09.public class Address implements java.io.Serializable {

10.

11.private long addressId;

12.private String street;

13.private String city;

14.private String state;

15.private String zipcode;

16.

17.public Address() {

18.}

19.

20.public Address(String street, String city, String state, String zipcode) {

21.this.street = street;

22.this.city = city;

23.this.state = state;

24.this.zipcode = zipcode;

25.}

26.

27.public long getAddressId() {

28.return this.addressId;

29.}

30.

31.public void setAddressId(long addressId) {

32.this.addressId = addressId;

33.}

34.

35.public String getStreet() {

36.return this.street;

37.}

38.

39.public void setStreet(String street) {

40.this.street = street;

41.}

42.

43.public String getCity() {

44.return this.city;

45.}

46.

47.public void setCity(String city) {

48.this.city = city;

49.}

50.

51.public String getState() {

52.return this.state;

53.}

54.

55.public void setState(String state) {

56.this.state = state;

57.}

58.

59.public String getZipcode() {

60.return this.zipcode;

61.}

62.

63.public void setZipcode(String zipcode) {

64.this.zipcode = zipcode;

65.}

66.

67.}

[view source](http://www.dzone.com/tutorials/java/hibernate/hibernate-example/hibernate-mapping-many-to-one-1.html#viewSource)

[print](http://www.dzone.com/tutorials/java/hibernate/hibernate-example/hibernate-mapping-many-to-one-1.html#printSource)[?](http://www.dzone.com/tutorials/java/hibernate/hibernate-example/hibernate-mapping-many-to-one-1.html#about)

01.package com.vaannila.student;

02.

03.import org.hibernate.HibernateException;

04.import org.hibernate.Session;

05.import org.hibernate.Transaction;

06.

07.import com.vaannila.util.HibernateUtil;

08.

09.public class Main {

10.

11.public static void main(String[] args) {

12.Session session = HibernateUtil.getSessionFactory().openSession();

13.Transaction transaction = null;

14.try {

15.transaction = session.beginTransaction();

16.Address address = new Address("OMR Road", "Chennai", "TN", "600097");

17.//By using cascade=all option the address need not be saved explicitly when the student object is persisted the address will be automatically saved.

18.//session.save(address);

19.Student student1 = new Student("Eswar", address);

20.Student student2 = new Student("Joe", address);

21.session.save(student1);

22.session.save(student2);

23.transaction.commit();

24.} catch (HibernateException e) {

25.transaction.rollback();

26.e.printStackTrace();

27.} finally {

28.session.close();

29.}

30.

31.}

32.

33.}

On executing the *Main* class you will see the following output.

The Student table has two records.

http://cdn.dzone.com/static/images/vaannila/hibernate/hibernateManyToOnePic3.gif

The Address table has one record.

http://cdn.dzone.com/static/images/vaannila/hibernate/hibernateManyToOnePic4.gif

 Source without lib