One to One Mapping in Hibernate by many-to-one example

We can perform one to one mapping in hibernate by two ways:

- By many-to-one element
- By one-to-one element

Here, we are going to perform one to one mapping by many-to-one element. In such case, a foreign key is created in the primary table.

In this example, one employee can have one address and one address belongs to one employee only. Here, we are using bidirectional association. Let's look at the persistent classes.

1) Persistent classes for one to one mapping

There are two persistent classes Employee.java and Address.java. Employee class contains Address class reference and vice versa.

Employee.java

```
package com.javatpoint;

public class Employee {
private int employeeId;
private String name,email;
private Address address;
//setters and getters
}
```

Address.java

```
package com.javatpoint;

public class Address {
private int addressId;
private String addressLine1,city,state,country;
private int pincode;
private Employee employee;
//setters and getters
}
```

2) Mapping files for the persistent classes

The two mapping files are employee.hbm.xml and address.hbm.xml.

employee.hbm.xml

In this mapping file we are using **many-to-one** element with unique="true" attribute to make the one to one mapping.

```
<?xml version='1.0' encoding='UTF-8'?>
<!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.javatpoint.Employee" table="emp211">
        <id name="employeeId">
        <generator class="increment"></generator>
        </id>
        <property name="name"></property>
        <property name="email"></property>

        <many-to-one name="address" unique="true" cascade="all"></many-to-one>
        </class>

        </hibernate-mapping>
```

address.hbm.xml

This is the simple mapping file for the Address class.

```
<?xml version='1.0' encoding='UTF-8'?>
```

```
<!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.javatpoint.Address" table="address211">
        <id name="addressId">
        <generator class="increment"></generator>
        </id>
        <property name="addressLine1"></property>
        <property name="city"></property>
        <property name="state"></property>
        <property name="country"></property>

        </class>

        </hibernate-mapping>
```

3) Configuration file

This file contains information about the database and mapping 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">

<!-- Generated by MyEclipse Hibernate Tools.                   -->
<hibernate-configuration>

    <session-factory>
        <property name="hbm2ddl.auto">update</property>
        <property name="dialect">org.hibernate.dialect.Oracle9Dialect</property>
        <property name="connection.url">jdbc:oracle:thin:@localhost:1521:xe</property>
        <property name="connection.username">system</property>
        <property name="connection.password">oracle</property>
        <property name="connection.driver_class">oracle.jdbc.driver.OracleDriver</property>
    <mapping resource="employee.hbm.xml"/>
    <mapping resource="address.hbm.xml"/>
    </session-factory>

</hibernate-configuration>
```

4) User classes to store and fetch the data

Store.java

```
package com.javatpoint;
import org.hibernate.cfg.*;
import org.hibernate.*;

public class Store {
public static void main(String[] args) {
    Configuration cfg=new Configuration();
    cfg.configure("hibernate.cfg.xml");
    SessionFactory sf=cfg.buildSessionFactory();
    Session session=sf.openSession();
    Transaction tx=session.beginTransaction();

    Employee e1=new Employee();
    e1.setName("Ravi Malik");
    e1.setEmail("ravi@gmail.com");

    Address address1=new Address();
    address1.setAddressLine1("G-21,Lohia nagar");
    address1.setCity("Ghaziabad");
    address1.setState("UP");
    address1.setCountry("India");
    address1.setPincode(201301);
    e1.setAddress(address1);
    address1.setEmployee(e1);

    session.persist(e1);
    tx.commit();

    session.close();
    System.out.println("success");
}
}
```

Fetch.java

```
package com.javatpoint;
import java.util.Iterator;
import java.util.List;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class Fetch {
public static void main(String[] args) {
    Configuration cfg=new Configuration();
    cfg.configure("hibernate.cfg.xml");
    SessionFactory sf=cfg.buildSessionFactory();
    Session session=sf.openSession();

    Query query=session.createQuery("from Employee e");
    List<Employee> list=query.list();

    Iterator<Employee> itr=list.iterator();
    while(itr.hasNext()){
     Employee emp=itr.next();
     System.out.println(emp.getEmployeeId()+" "+emp.getName()+" "+emp.getEmail());
     Address address=emp.getAddress();
     System.out.println(address.getAddressLine1()+" "+address.getCity()+" "+
        address.getState()+" "+address.getCountry());
    }

    session.close();
    System.out.println("success");
}
}
```

download this hibernate example

← prev

next →