Hibernate One to One Example(Foreign key)

Here, we are going to perform one to one mapping by one-to-one element. In such case, no 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.

**public** **class** Employee {

**private** **int** employeeId;

**private** String name,email;

**private** Address address;

//setters and getters

}

------------------------------------------

**public** **class** Address {

**private** **int** addressId;

**private** String addressLine1,city,state,country;

**private** **int** pincode;

//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 **one-to-one** element in both the mapping files to make the one to one mapping.

|  |
| --- |
| <?xml version='1.0' encoding='UTF-8'?>  <!DOCTYPE hibernate-mapping PUBLIC            "-//Hibernate/Hibernate Mapping DTD 5.3//EN"            "http://hibernate.sourceforge.net/hibernate-mapping-5.3.dtd">               <hibernate-mapping>            <**class** name="com.abc.Employee" table="emp212">            <id name="employeeId">            <generator **class**="increment"></generator>            </id>            <property name="name"></property>            <property name="email"></property>              <one-to-one name="address" cascade="all"></one-to-one>            </**class**>              </hibernate-mapping> |

#### **address.hbm.xml**

This is the simple mapping file for the Address class. But the important thing is generator class. Here, we are using **foreign** generator class that depends on the Employee class primary key.

|  |
| --- |
| <?xml version='1.0' encoding='UTF-8'?>  <!DOCTYPE hibernate-mapping PUBLIC            "-//Hibernate/Hibernate Mapping DTD 5.3//EN"            "http://hibernate.sourceforge.net/hibernate-mapping-5.3.dtd">               <hibernate-mapping>            <**class** name="com.abc.Address" table="address212">            <id name="addressId">            <generator **class**="foreign">            <param name="property">employee</param>            </generator>            </id>            <property name="addressLine1"></property>            <property name="city"></property>            <property name="state"></property>            <property name="country"></property>            <property name="pincode"></property>              <one-to-one name="employee"></one-to-one>            </**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 5.3//EN"            "http://hibernate.sourceforge.net/hibernate-configuration-5.3.dtd">    <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">jtp</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> |

Test Class

|  |
| --- |
| Transaction t=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);      t.commit();        session.close();      System.out.println("success"); |

### **3) Create the Persistence class using Annotations**

Here, we are creating the same persistent class which we have created in the previous topic. But here, we are using annotation.

**@Entity** annotation marks this class as an entity.

**@Table** annotation specifies the table name where data of this entity is to be persisted. If you don't use @Table annotation, hibernate will use the class name as the table name by default.

**@Id** annotation marks the identifier for this entity.

**@Column** annotation specifies the details of the column for this property or field. If @Column annotation is not specified, property name will be used as the column name by default.

1. **import** javax.persistence.Entity;
2. **import** javax.persistence.Id;
3. **import** javax.persistence.Table;
5. @Entity
6. @Table(name= "emp500")
7. **public** **class** Employee {
9. @Id
10. **private** **int** id;
11. **private** String firstName,lastName;
13. **public** **int** getId() {
14. **return** id;
15. }
16. **public** **void** setId(**int** id) {
17. **this**.id = id;
18. }
19. **public** String getFirstName() {
20. **return** firstName;
21. }
22. **public** **void** setFirstName(String firstName) {
23. **this**.firstName = firstName;
24. }
25. **public** String getLastName() {
26. **return** lastName;
27. }
28. **public** **void** setLastName(String lastName) {
29. **this**.lastName = lastName;
30. }
31. }
32. @Column(name="DESC", nullable=false, length=512)