Hibernate – IV

Association Mapping

Table of Contents

Introduction

- Hibernate, provides great features to put relationship between two entities.
- The main advantage of putting relationship between objects is, we can do operation on one object, and the same operation can transfer onto the other object in the database.
- Relationship can be established by using foreign key only.
- Object means one row in hibernate terminology

Using Hibernate we can make the following types of relationships

- ✓ One-To-One
- ✓ Many-To-One
- ✓ Many-To-Many
- ✓ One-To-Many

Unidirectional associations

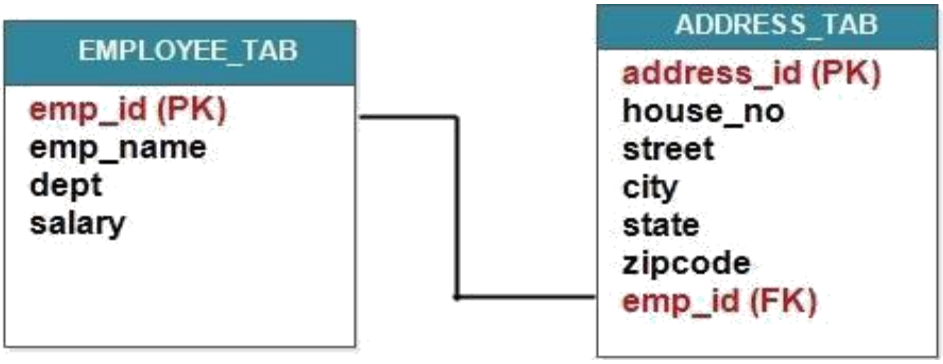
One to One Mapping

- One object is associated with one object only
- To apply one to one relationship, we copy the primary key value of parent object into primary key value of the child object. So that the relationship between two objects is one to one
- If we want to copy parent object primary key value into child object primary key, we need to use a special generator class **"foreign"**
- **Foreign** generator is only used in one to one relationship only

Let's see in below **Application1 (unidirectional-one-to-one-using-joincolumn-and-annotation)**

hibernate.cfg.xml

```
//Add DTD configuration
<hibernate-configuration>

<session-factory>

<property name="connection.driver_class">com.mysql.jdbc.Driver</property>
<property name="connection.url">jdbc:mysql://localhost:3306/hibernate</property>

<property name="connection.username">root</property>
 <property name="connection.password">admin</property>

<property name="hbm2ddl.auto">update</property>

<property name="dialect">org.hibernate.dialect.MySQLDialect</property>
 <property name="show_sql">true</property>

<mapping class="com.kalibermind.hibernate.entity.Employee" />
<mapping class="com.kalibermind.hibernate.entity.Address" />

</session-factory>
</hibernate-configuration>
```



Employee.java (POJO with annotation)

```
package com.kalibermind.hibernate.entity;

import javax.persistence.*;

@Entity
@Table(name="EMPLOYEE_TAB1")
public class Employee {

   @Id
   @GeneratedValue(strategy=GenerationType.IDENTITY)
   @Column(name="EMP_ID")
   private int empId;
   @Column(name="EMP_NAME")
   private String empName;
   private String dept;
   private double salary;

  @OneToOne(cascade=CascadeType.ALL)
  @JoinColumn(name="EMP_ID")
  private Address address;

    public Employee() {}

  public Employee(String empName, String dept, double
                                                  salary)
  {
          this.empName = empName;
          this.dept = dept;
          this.salary = salary;
  }
//corresponding setter and getter
}
```

Address.java (POJO with annotation)

```
package com.kalibermind.hibernate.entity;

import javax.persistence.*;

@Entity
@Table(name="ADDRESS_TAB1")
public class Address {

    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    @Column(name="ADDRESS_ID")
     private int addressId;

     @Column(name="HOUSE_NO")
     private String houseNo;
     private String street;
     private String city;
     private String state;
     private int zipcode;

      public Address() {}

  public Address(String houseNo, String street, String city,
                                  String state, int zipcode)
    {
          this.houseNo = houseNo;
          this.street = street;
          this.city = city;
          this.state = state;
          this.zipcode = zipcode;
     }
   //corresponding setter and getter
}
```

Note:

@OneToOne: it is use to put **one-to- one** relationship between two entity

@JoinColumn(name="EMP_ID") : is use to pass the join column **or** to make the **foreign key**

SaveData.java

```
package com.kalibermind.hibernate.test;

import org.hibernate.Session;
 import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;
import com.kalibermind.hibernate.entity.Address;
 import com.kalibermind.hibernate.entity.Employee;

public class SaveData {
   public static void main(String[] args)
   {
Configuration cfg = new Configuration().configure();
SessionFactory factory = cfg.buildSessionFactory();

Address  address = new
              Address("H10","Silk
     Board","Bangalore","KA",560068);
Employee emp = new
Employee("CPVERMA","IT",95000.00);
emp.setAddress(address);

    Session session = factory.openSession();
    Transaction txn = session.beginTransaction();
    session.save(emp);
     txn.commit();
     session.close();
    System.out.println("Record has been saved
    successfully!");
   }

}
```

FetchData.java

```
 package com.kalibermind.hibernate.test;

 import org.hibernate.Session;
import org.hibernate.SessionFactory;
 import org.hibernate.cfg.Configuration;

 import com.kalibermind.hibernate.entity.Employee;

 public class FetchData {

public static void main(String[] args)
{
Configuration cfg = new Configuration().configure();
SessionFactory factory = cfg.buildSessionFactory();

Session session = factory.openSession();
Employee emp = session.get(Employee.class,1);
session.close();

System.out.println("Employee Name: "+emp.getEmpName());


System.out.println("Employee City:
"+emp.getAddress().getCity());

    }

 }
```

Application2 -unidirectional one-to-one **using joinable** and annotation

It is same as per the using **join column** application; there is only **small change** we have to do

We have to pass @**JoinTable (name="EMPLOYEE_ADDRESS")**

Instead of

@**JoinColumn** (name="EMP_ID")

Application3 -Bidirectional one-to-one using joincolumn and annotation

Same application as per the previous but in **Bidirectional** mapping is done from both entities only two changes we have to map, in **parent** and **child** entity we have to map @**OneToOne**

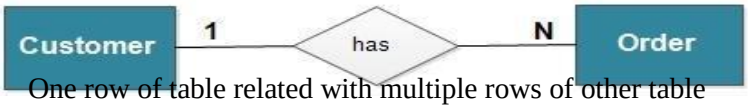
Employee.java (POJO with annotation)

```
package com.kalibermind.hibernate.entity;
import javax.persistence.*;

@Entity
@Table(name="EMPLOYEE_TAB3")
public class Employee {

    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    @Column(name="EMP_ID")
    private int empId;

     @Column(name="EMP_NAME")
     private String empName;
     private String dept;
     private double salary;

     @OneToOne(cascade=CascadeType.ALL)
     @JoinColumn(name="ADDRESS_ID")
     private Address address;

//Constructor and corresponding setter and getter

}
```

Address.java (POJO with annotation)

```
package com.kalibermind.hibernate.entity;
import javax.persistence.*;

@Entity
@Table(name="ADDRESS_TAB3")
public class Address {

    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    @Column(name="ADDRESS_ID")
    private int addressId;

     @Column(name="HOUSE_NO")
     private String houseNo;
     private String street;
     private String city;
     private String state;
    private int zipcode;

    @OneToOne
    @JoinColumn(name="ADDRESS_ID")
    private Employee employee;

//Constructor and corresponding setter and
 getter }
```

One to Many Mapping



- One row of table related with multiple rows of other table
- To achieve one-to-many between two pojo classes in the hibernate, then the following two changes are required

I. In the **parent** pojo class, we need to take a collection property; the collection can be either **Set**, **List, Map**.
II. In the **mapping file** of that parent pojo class, we need to configure the collection

Application4 (unidirectional one-to-many using joincolumn and annotation)

Required files

hibernate.cfg.xml

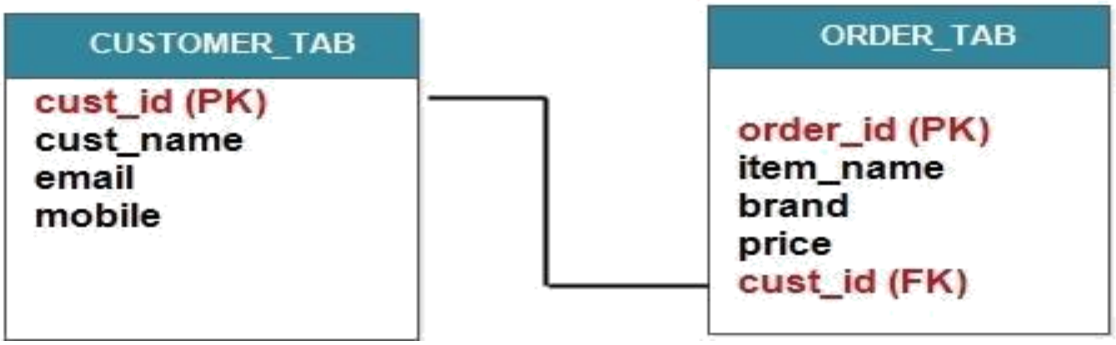
```
//Add DTD Configuration
<hibernate-configuration>
   <session-factory>
     <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
      <property    name="connection.url">jdbc:mysql://localhost:3306/hibernate
                                                                    </property>
      <property name="connection.username">root</property>
       <property name="connection.password">admin</property>

         <property name="hbm2ddl.auto">update</property>
         <property name="dialect">org.hibernate.dialect.MySQLDialect</property>
         <property name="show_sql">true</property>

<mapping class="com.kalibermind.hibernate.entity.Customer"/>
        <mapping class="com.kalibermind.hibernate.entity.Order" />
 </session-factory>
</hibernate-configuration>
```

Customer.java (POJO)

```
package com.kalibermind.hibernate.entity;
import java.util.HashSet;
import java.util.Set;
import javax.persistence.*;
@Entity
@Table(name="CUSTOMER_TAB2")
public class Customer {
    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    @Column(name="CUST_ID")
    private long custId;
    @Column(name="CUST_NAME")
    private String custName;
    private String email;
    private long mobile;
   @OneToMany(cascade=CascadeType.ALL)
   @JoinColumn(name="CUST_ID")

   private Set<Order> orders = new HashSet<>(0);
//Constructor and corresponding setter and getter
 }
```

Order.java(POJO with annotation)

```
package com.kalibermind.hibernate.entity;
import javax.persistence.*;

@Entity
@Table(name="ORDER_TAB2")
public class Order {
    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    @Column(name="ORDER_ID")
    private long orderId;

    @Column(name="ITEM_NAME")
     private String itemName;
     private String brand;
     private double price;

//Constructor and corresponding setter and getter


}
```



SaveData.java

```
package com.kalibermind.hibernate.test;
import java.util.HashSet;
import java.util.Set;
import org.hibernate.Session;
 import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;
import com.kalibermind.hibernate.entity.Customer;
import com. kalibermind.hibernate.entity.Order;

public class SaveData {

    public static void main(String[] args)
    {
      Configuration cfg = new Configuration().configure();
      SessionFactory factory = cfg.buildSessionFactory();

    Set<Order> orders = new HashSet<>();
    Order o1 = new Order ("Laptop","Lenovo",55000.00);
    Order o2 = new Order ("Mobile","Moto-G",5000.00);
    Order o3 = new Order ("LED","Micromax",45000.00);

  orders.add(o1);
  orders.add(o2);
  orders.add(o3);

 Customer cust = new Customer("CPVerma",
                "cp.verma@gmail.com",9999999900L);
cust.setOrders(orders);

Session session = factory.openSession();
Transaction txn = session.beginTransaction();
 session.save(cust);
 txn.commit();
session.close();
System.out.println("Record has been saved ");
  }
}
```

FetchData.java

```
package com.kalibermind.hibernate.test;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

import com. kalibermind.hibernate.entity.Customer;

public class FetchData {
     public static void main(String[] args)
    {
      Configuration cfg = new Configuration().configure();
      SessionFactory factory = cfg.buildSessionFactory();

     Session session = factory.openSession();
   Customer cust = session.get(Customer.class,new Long(1));

  System.out.println("Customer Name:
                                        "+cust.getCustName());
   System.out.println("Customer Order:
                                    "+cust.getOrders().toString());
       session.close();
  }
}
```

Application5 (unidirectional one-to-many using-joinTable and annotation)

It is same as per the using **join column** application; there is only **small change** we have to do

We have to pass @**JoinTable(name="CUSTOMER_ORDER")**

Instead of

@JoinColumn(name="CUST_ID")

Application6 (Bidirectional one-to-many/Many-to-one using joinColumn and annotation)

- It is same as **one-to-many** but in Bidirectional **one-to-many** parent objects **to <many-to-one name="">** is used from **child object.**
- In the child pojo class mapping file we need to write **<many-to-one name="">** .element.

```
@Entity
@Table(name="CUSTOMER_TAB3")
public class Customer {
        @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    @Column(name="CUST_ID")
    private long custId;

    @Column(name="CUST_NAME")
    private String custName;
    private String email;
     private long mobile;

    @OneToMany(cascade=CascadeType.ALL)
    @JoinColumn(name="CUST_ID")
    private Set<Order> orders = new HashSet<>(0);

}
```

```
@Entity
@Table(name="ORDER_TAB3")
public class Order {
    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    @Column(name="ORDER_ID")
     private long orderId;

    @Column(name="ITEM_NAME")
    private String itemName;
    private String brand;
    private double price;

    @ManyToOne
    @JoinColumns(value=@JoinColumn(name="CUST_ID"))
     private Customer customer;

}
```

Many to One Mapping

- It is same as one to many but in many to one, the relationship is applied from **child object to parent** object, but in **one to many parent objects to child object.**
- In the child pojo class mapping file we need to write **<many-to-one name="">** .element.

Many to Many Mapping

- It can only be Bi-Directional.
- whenever we are applying **many to many** relationship between two pojo class objects, on **both sides** we need a collection property
- **Join table** is mandatory in the database, to store primary key as foreign key both sides.