Hibernate Association Mapping & Inheritance

Sarbananda Behera

manipalglobal
education services

INTRODUCTION

Association Mapping

OBJECTIVES

Managing Associations and Inheritance Mapping in Hibernate

manipalglobal
education services

LEARNING OBJECTIVES

At the end of this lesson, you will be able to:

- Define Association Mapping
- Learn various different types of Associations
- Differentiate between Unidirectional vs Bidirectional Associations
- One to One , One to Many and Many to Many Types of Associations

One to One Mapping

One employee can have one address and one address belongs to one employee only. Here, we are using bidirectional association.

Employee.java

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

Address.java

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

Employe.hbm.xml

Mapping for Employee entity

```
<?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.mage.Employee" table="emp">
          <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 (Mapping file for Address Entity)

```
<?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="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>
          <one-to-one name="employee"></one-to-one>
          </class>
          </hibernate-mapping>
```

One to Many Association Mapping

➢ There can be many answers for a question .

Question.java

```
import java.util.List;

public class Question {
private int id;
private String qname;
private Set<Answer> answers;

//getters and setters

}
```

Answer.java

```
public class Answer {
private int id;
private String answername;
private String postedBy;
//getters and setters

}
}
```

One to Many Association Mapping

Question.hbm.xml

```
<set name="answers" cascade="all">
          <key column="qid"></key>
          <one-to-many class="com.mage.Answer"/>
</set>
```

Many to Many Association Mapping

- A STOCK table has more than one CATEGORY, and CATEGORY can belong to more than one STOCK, the relationship is linked with a third table called STOCK_CATEGORY.

Stock.java

```
  public class Stock {
           private Integer stockId;
           private String stockCode;
           private String stockName;
           private Set<Category>
categories = new
HashSet<Category>(0);

           //getter, setter
}
```

Category.java

```
public class Category
 private Integer categoryId;
 private String name;
 private String desc;
 private Set<Stock> stocks = new
HashSet<Stock>(0);

 //getter, setter and constructor
}
```

Stock.hbm.xml

Mapping file for Stock Entity

```
<set name="categories" table="stock_category"
       inverse="false" lazy="true" fetch="select" cascade="all" >
          <key>
              <column name="STOCK_ID" not-null="true" />
          </key>
          <many-to-many entity-name="com.mage.Category">
              <column name="CATEGORY_ID" not-null="true" />
          </many-to-many>
      </set>
```

Category.hbm.xml

Mapping file for Category Entity

```
 <set name="stocks" table="stock_category" inverse="true" lazy="true"
fetch="select">
            <key>
                <column name="CATEGORY_ID" not-null="true" />
            </key>
            <many-to-many entity-name="com.mage.Stock">
                <column name="STOCK_ID" not-null="true" />
            </many-to-many>
        </set>
```

Component Mapping

- A component is an object that is stored as an value rather than entity reference.
- This is mainly used if the dependent object doesn't have primary key.
- It is used in case of composition (HAS-A relation), that is why it is termed as component.
- Let's see the class that have HAS-A relationship.

Employee.java

```
public class Employee {
private int id;
private String name;
private Address address;//HAS-A

//getters and setters
}
```

Address.java

```
public class Address {
private String city,country;
private int pincode;

//getters and setters
}
```

Composition Relationship mapping

```
 <class name="com.mage.Employee" table="emp177">
<id name="id">
<generator class="increment"></generator>
</id>
<property name="name"></property>

<component name="address" class="com.mage.Address">
<property name="city"></property>
<property name="country"></property>
<property name="pincode"></property>
</component>  </class>
```

Inheritance Mapping in Hibernate

- Object oriented systems can model both **“is a”** and **“has a”** relationship.
- Relational model supports only “has a” relationship between two entities.
- Hibernate can help you map such Objects with relational tables by providing certain mapping strategy based on your needs.
- Hibernate Supports the following inheritance mapping strategies
  - Table per class hierarchy
  - Table per Sub Class
  - Table per Concrete Class

Inheritance Mapping Example

The picture shows the Inheritance relationship existing between the following classes in the Object domain

- Vehicle (Base class)
- Two-Wheeler (Child Class)
- Four-Wheeler(Child Class)

➢ Hibernate Inheritance mapping strategies can be applied to map this inheritance hierarchy into relational Database Tables

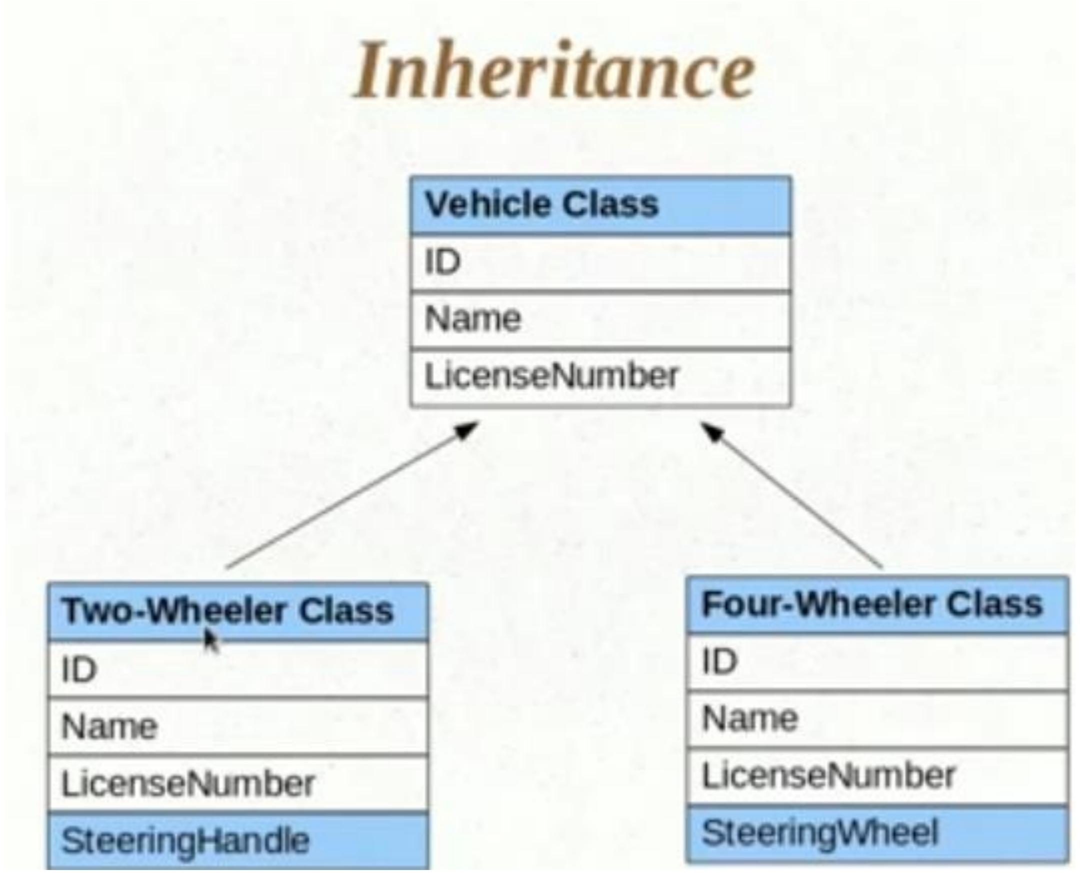

Table per Class Hierarchy

- ➢ The union of all the properties from the inheritance hierarchy is mapped to one table.
- ➢ A discriminator is used to differentiate between different type of data.

Advantages :

- Hierarchy Simplest to implement.
- Only one table to deal with.
- Performance wise better than all strategies because no joins or sub-selects need to be performed.

Disadvantages:

- Most of the column of table are nullable so the NOT NULL constraint cannot be applied.
- Tables are not normalized.

Table Per Concrete Class Strategy

- In this case every entity class has its own table i.e. table per class. The data for Vehicle is duplicated in both the tables.
- This strategy is not popular and also have been made optional in Java Persistence API.

Advantages :

- Possible to define NOT NULL constraints on the table.

Disadvantages:

- Tables are not normalized.
- To support polymorphism either container has to do multiple trips to database or use SQL UNION kind of feature.

Join Strategy (Table per SubClass)

- It's highly normalized.

Advantages :

- Tables are normalized.
- Able to define NOT NULL constraint.

Disadvantages:

- Does not perform as well as SINGLE_TABLE strategy

QUIZ QUESTION

Which type of Association mapping is needed to implement relationship between class “Man” and class “Heart”

- ❑ One to Many Association
- ❑ Inheritance Mapping
- ❑ Component Mapping
- ❑ All of the above

QUIZ QUESTION

In which type of inheritance mapping strategy , a discriminator value is used ?

- ❑ Single Table strategy
- ❑ Table Per Class
- ❑ Joined Subclass
- ❑ All of the above

SUMMARY

Association and Inheritance Mapping

SUMMARY

In this lesson, you've learned to:

- Hibernate Association Mapping
- One to One Mapping
- One to Many Mapping
- Many to Many Mapping
- Inheritance Mapping with Hibernate
- Single Table Inheritance
- Table per class inheritance
- Join Strategy