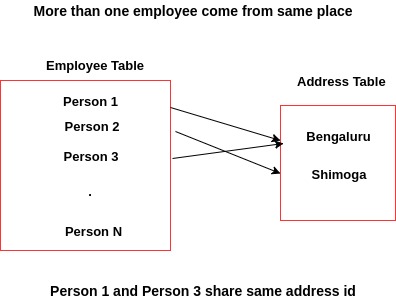
Many rows from one table map to single row in another table

Ex. more than one employee may come from same address(city)



The *address\_id* is the foreign key to the table *employee*. With the help of *address\_id* , we are here doing one to many mapping. The logic behind using this mapping in below example is - many employees can have the same street, city, state ,country as address. (In the sample we are ignoring door number)

**Prepare database**

Create database manytoone;

CREATE TABLE `employee` (

`employee\_id` bigint(10) NOT NULL auto\_increment,

`firstname` varchar(50) default NULL,

`lastname` varchar(50) default NULL,

`cell\_phone` varchar(15) default NULL,

`address\_id` bigint(20) default NULL,

PRIMARY KEY (`employee\_id`),

KEY `FK\_employee` (`address\_id`),

CONSTRAINT `FK\_employee` FOREIGN KEY (`address\_id`) REFERENCES `address` (`address\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1

CREATE TABLE `address` (

`address\_id` bigint(20) NOT NULL auto\_increment,

`street` varchar(50) default NULL,

`city` varchar(50) default NULL,

`state` varchar(50) default NULL,

`country` varchar(50) default NULL,

PRIMARY KEY (`address\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1

**Steps to achieve one to many mapping**

**1.create project web development project and configure to maven**

(may be not the ideal way, you could create directly maven project)

**2.open pom.xml file and add dependencies into it.**

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>OneToMany</groupId>

<artifactId>OneToMany</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>war</packaging>

<dependencies>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>4.0.1.Final</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-validator</artifactId>

<version>4.2.0.Final</version>

</dependency>

<dependency>

<groupId>org.hibernate.common</groupId>

<artifactId>hibernate-commons-annotations</artifactId>

<version>4.0.1.Final</version>

<classifier>tests</classifier>

</dependency>

<dependency>

<groupId>org.hibernate.javax.persistence</groupId>

<artifactId>hibernate-jpa-2.0-api</artifactId>

<version>1.0.1.Final</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-entitymanager</artifactId>

<version>4.1.8.Final</version>

</dependency>

<dependency>

<groupId>javax.validation</groupId>

<artifactId>validation-api</artifactId>

<version>1.0.0.GA</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.6.4</version>

</dependency>

<dependency>

<groupId>org.jboss.logging</groupId>

<artifactId>jboss-logging</artifactId>

<version>3.1.0.CR2</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>1.6.4</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.10</version>

</dependency>

</dependencies>

<build>

<sourceDirectory>src</sourceDirectory>

<plugins>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.5.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

<plugin>

<artifactId>maven-war-plugin</artifactId>

<version>3.0.0</version>

<configuration>

<warSourceDirectory>WebContent</warSourceDirectory>

</configuration>

</plugin>

</plugins>

</build>

</project>

3.create hibernate configuration file **hibernate.cfg.xml**

This file should be paste under source folder

<?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">

<hibernate-configuration>

<session-factory>

<!-- Database connection settings -->

<property name="connection.driver\_class">com.mysql.jdbc.Driver</property>

<property name="connection.url">jdbc:mysql://localhost:3306/manytoone</property>

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

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

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

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

<property name="current\_session\_context\_class">thread</property>

<property name="cache.provider\_class">org.hibernate.cache.NoCacheProvider</property>

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

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

<mapping class="com.worldclock.Address" />

<mapping class="com.worldclock.Employee" />

</session-factory>

</hibernate-configuration>

**4.create Address.java**

package com.worldclock;

import java.util.Set;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

import javax.persistence.OneToMany;

import javax.persistence.Table;

@Entity

@Table(name = "address")

public class Address {

@Id

@GeneratedValue

@Column(name = "address\_id")

private Long addressId;

@Column(name = "street")

private String street;

@Column(name = "city")

private String city;

@Column(name = "state")

private String state;

@Column(name = "country")

private String country;

@OneToMany(mappedBy = "**address**")

private Set<Employee> employees;

public Long getAddressId() {

return addressId;

}

public void setAddressId(Long addressId) {

this.addressId = addressId;

}

public String getStreet() {

return street;

}

public void setStreet(String street) {

this.street = street;

}

public String getCity() {

return city;

}

public void setCity(String city) {

this.city = city;

}

public String getState() {

return state;

}

public void setState(String state) {

this.state = state;

}

public String getCountry() {

return country;

}

public void setCountry(String country) {

this.country = country;

}

public Set<Employee> getEmployees() {

return employees;

}

public void setEmployees(Set<Employee> employees) {

this.employees = employees;

}

}

**5.create Employee.java**

package com.worldclock;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import javax.persistence.Table;

@Entity

@Table(name="employee")

public class Employee {

@Id

@GeneratedValue

@Column(name="employee\_id")

private Long employeeId;

@Column(name="firstname")

private String firstname;

@Column(name="lastname")

private String lastname;

@Column(name="cell\_phone")

private String cellphone;

@ManyToOne

@JoinColumn(name="address\_id")

private Address **address**;

public Address getAddress() {

return address;

}

public void setAddress(Address address) {

this.address = address;

}

public Employee() {}

public Employee(String firstname, String lastname, String phone) {

this.firstname = firstname;

this.lastname = lastname;

this.cellphone = phone;

}

public Long getEmployeeId() {

return employeeId;

}

public void setEmployeeId(Long employeeId) {

this.employeeId = employeeId;

}

public String getFirstname() {

return firstname;

}

public void setFirstname(String firstname) {

this.firstname = firstname;

}

public String getLastname() {

return lastname;

}

public void setLastname(String lastname) {

this.lastname = lastname;

}

public String getCellphone() {

return cellphone;

}

public void setCellphone(String cellphone) {

this.cellphone = cellphone;

}

}

**6.Test class ManageEmployee.java**

package com.worldclock;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

import org.hibernate.service.ServiceRegistry;

import org.hibernate.service.ServiceRegistryBuilder;

public class ManageEmployee {

private static SessionFactory sf;

private static ServiceRegistry serviceRegistry;

@SuppressWarnings("unchecked")

public static void main(String[] args) {

try {

Configuration configuration = new Configuration();

configuration.configure();

serviceRegistry = new ServiceRegistryBuilder().applySettings(

configuration.getProperties()).buildServiceRegistry();

sf = configuration.buildSessionFactory(serviceRegistry);

} catch (Throwable ex) {

System.err.println("Failed to create sessionFactory object." + ex);

throw new ExceptionInInitializerError(ex);

}

System.out.println("Hibernate One to Many Mapping Example Using Annotation ");

Session session = sf.openSession();

session.beginTransaction();

Address address = new Address();

address.setStreet("sindhiya street");

address.setCity("Gwalior");

address.setState("Madhya Pradesh");

address.setCountry("India");

session.save(address);

Employee e1 = new Employee("Ankit", "Sharma", "9999999999");

Employee e2 = new Employee("Ankit", "Kaushal", "3333333333");

e1.setAddress(address);

e2.setAddress(address);

session.save(e1);

session.save(e2);

session.getTransaction().commit();//saved to db

session.close();

}

}