

***HIBERNATE***

**Training Lab**

**Many-to-Many Entity Mapping**

|  |  |
| --- | --- |
| Document Code | 25e-BM/HR/HDCV/FSOFT |
| Version | 1.1 |
| Effective Date | 20/11/2012 |

**Hanoi, 06/2020**

RECORD OF CHANGES

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Effective Date | Change Description | Reason | Reviewer | Approver |
| 1 | 15/06/2020 | Create Lab |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Contents

**1. Problem Description 4**

**2. Technologies and Tools Used 4**

**3. Development Steps 5**

3.1, Project Directory Structure 5

3.2, Add jar Dependencies to pom.xml 6

3.3, ★Creating the JPA Entity Class (Persistent class) 7

3.4, Create a Hibernate configuration file - hibernate.cfg.xml 9

3.5, Create a Hibernate utility class 10

3.6, Implement CRUD operations (ex: save an entity) 11

3.7, Create the class for testing 12

|  |  |
| --- | --- |
|  | **CODE: ORM.L.A102**  **TYPE: LONG**  **LOC: NA**  **DURATION: 480 Minutes (completed in 2 work days)** |

1. Problem Description

* We will show you how to implement step by step **many-to-many** entity mapping using JPA/Hibernate with MySQL database.
* Consider the following tables where employees and projects exhibit a **many-to-many** relationship between each other. The many-to-many relationship is implemented using a third table called employees\_projects which contains the details of the employees and their associated projects. Note that here Employee is a primary entity.

**employee\_id**: BIGINT(20)

**created\_at**: DATETIME(6)

**updated\_at**: DATETIME(6)

**first\_name**: VARCHAR(255)

**last\_name**: VARCHAR(255)

employees

**project\_id**: BIGINT(20)

**created\_at**: DATETIME(6)

**updated\_at**: DATETIME(6)

**title**: VARCHAR(255)

projects

**employee\_id**: BIGINT(20)

**project\_id**: BIGINT(20)

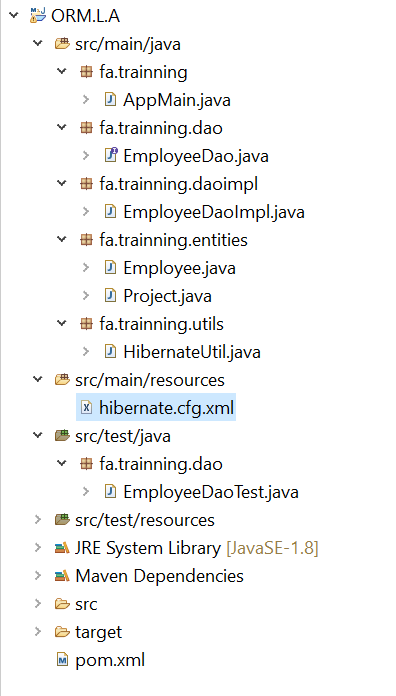
Employees\_projects

2. Technologies and Tools Used

* Hibernate 5.4
* IDE - Eclipse
* Maven
* JavaSE 1.8
* MySQL

3. Development Steps

**3.1, Project Directory Structure**

**3.2, Add jar Dependencies to pom.xml**

JPA Entity

Hibernate configuration

Session factory configuration

<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>

<parent>

<groupId>hibernate</groupId>

<artifactId>hibernate-lab</artifactId>

<version>0.0.1-SNAPSHOT</version>

</parent>

<artifactId>hibernate-lab2</artifactId>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

**<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->**

<dependency>

<groupId>mysql</groupId>

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

<version>8.0.13</version>

</dependency>

**<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->**

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>**hibernate-core**</artifactId>

<version>5.4.18.Final</version>

</dependency>

</dependencies>

<build>

<sourceDirectory>src/main/java</sourceDirectory>

<plugins>

<plugin>

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

<version>3.5.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

**3.3, ★Creating the JPA Entity Class (Persistent class)**

Create a EmployeeandProject entities class under fa.trainning.entities package as follows.

* Employee:

package fa.trainning.entities;

import java.util.HashSet;

import java.util.Set;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.JoinTable;

import javax.persistence.ManyToMany;

import javax.persistence.Table;

@Entity

@Table(name = "employees")

public class Employee {

private static final long serialVersionUID = 1L;

@Id

@Column(name = "employee\_id")

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long employeeId;

@Column(name = "first\_name")

private String firstName;

@Column(name = "last\_name")

private String lastName;

@ManyToMany(cascade = { CascadeType.ALL })

@JoinTable(

name = "employees\_projects",

joinColumns = { @JoinColumn(name = "employee\_id") },

inverseJoinColumns = { @JoinColumn(name = "project\_id") }

)

Set <Project> projects = new HashSet <Project>();

public Employee() {

super();

}

public Employee(String firstName, String lastName) {

this.firstName = firstName;

this.lastName = lastName;

}

public Employee(String firstName, String lastName, Set < Project > projects) {

this.firstName = firstName;

this.lastName = lastName;

this.projects = projects;

}

// getter & setter

// …

}

* Project:

package fa.trainning.entities;

import java.util.HashSet;

import java.util.Set;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

import javax.persistence.ManyToMany;

import javax.persistence.Table;

@Entity

@Table(name = "projects")

public class Project {

　 private static final long serialVersionUID = 1L;

@Id

@Column(name = "project\_id")

@GeneratedValue

private Long projectId;

@Column(name = "title")

private String title;

@ManyToMany(mappedBy = "projects", cascade = { CascadeType.ALL })

private Set<Employee> employees = new HashSet<Employee>();

public Project() {

super();

}

public Project(String title) {

this.title = title;

}

// getter & setter

// …

}

**3.4, Create a Hibernate configuration file - hibernate.cfg.xml**

The configuration file contains information about the database and mapping file. Conventionally, its name should be hibernate.cfg.xml.

Let's create an XML file named as hibernate.cfg.xml*under*resources*folder* and write the following code in it.

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<!-- JDBC Database connection settings -->

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

**<property name="connection.url">jdbc:mysql://localhost:3306/hibernatelab\_db?useSSL=false</property>**

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

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

<!-- JDBC connection pool settings ... using built-in test pool -->

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

<!-- Select our SQL dialect -->

<property name="dialect">org.hibernate.dialect.MySQL5Dialect</property>

<!-- Echo the SQL to stdout -->

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

<!-- Set the current session context -->

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

<!-- Drop and re-create the database schema on startup -->

<property name="hbm2ddl.auto">create-drop</property>

<!-- dbcp connection pool configuration -->

<property name="hibernate.dbcp.initialSize">5</property>

<property name="hibernate.dbcp.maxTotal">20</property>

<property name="hibernate.dbcp.maxIdle">10</property>

<property name="hibernate.dbcp.minIdle">5</property>

<property name="hibernate.dbcp.maxWaitMillis">-1</property>

<mapping class="**fa.trainning.entities.Employee**"/>

<mapping class="**fa.trainning.entities.Project**"/>

</session-factory>

</hibernate-configuration>

**3.5, Create a Hibernate utility class**

The bootstrapping API is quite flexible, but in most cases, it makes the most sense to think of it as a 3 steps process:

* Build the StandardServiceRegistry
* Build the Metadata
* Use those 2 to build the SessionFactory

package fa.trainning.utils;

import org.hibernate.SessionFactory;

import org.hibernate.boot.Metadata;

import org.hibernate.boot.MetadataSources;

import org.hibernate.boot.registry.StandardServiceRegistry;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

public class HibernateUtil {

private static StandardServiceRegistry registry;

private static SessionFactory sessionFactory;

public static SessionFactory getSessionFactory() {

if (sessionFactory == null) {

try {

// Create registry

registry = new StandardServiceRegistryBuilder().configure().build();

// Create MetadataSources

MetadataSources sources = new MetadataSources(registry);

// Create Metadata

Metadata metadata = sources.getMetadataBuilder().build();

// Create SessionFactory

sessionFactory = metadata.getSessionFactoryBuilder().build();

} catch (Exception e) {

e.printStackTrace();

if (registry != null) {

StandardServiceRegistryBuilder.destroy(registry);

}

}

}

return sessionFactory;

}

public static void shutdown() {

if (registry != null) {

StandardServiceRegistryBuilder.destroy(registry);

}

}

}

**3.6, Implement** **CRUD operations**

Create a EmployeeDao interface and EmployeeDao Impl class as follows

* Interface

package fa.trainning.dao;

import fa.trainning.entities.Employee;

public interface EmployeeDao {

public void saveEmployee(Employee employee);

}

* Implementation class

package fa.trainning.daoimpl;

import org.hibernate.Session;

import fa.trainning.entities.Employee;

public class EmployeeDaoImpl implements EmployeeDao {

private static Session session = HibernateUtil.getSessionFactory().openSession();

**public void saveEmployee(Employee employee) {**

// Save the employee object

session.save(employee);

}

}

**3.7, Create the class for testing**

Here is main App class which is used to connect MySQL database and persist EmployeeandProject object in database table.

* Main class

package hibernate;

import java.util.List;

import org.hibernate.Session;

import org.hibernate.Transaction;

import fa.trainning.entities.Employee;

import fa.trainning.entities.Project;

import fa.trainning.utils.HibernateUtil;

import fa.trainning.dao.EmployeeDao;

public class App {

public static void main(String[] args) {

// Create DAO object

EmployeeDao employeeDao = new EmployeeDao();

// Create Transaction

Transaction transaction = null;

try (**Session session = HibernateUtil.getSessionFactory().openSession()**) {

// Start a transaction

transaction = session.beginTransaction();

**// TODO: add you code here… BEGIN**

// Create an employee

Employee employee = new Employee();

employee.setFirstName("Anna");

employee.setLastName("Hook");

// Create a project

Project project1 = new Project();

project.setTitle("Employee Management System");

// Create other project

Project project2 = new Project();

project1.setTitle("Content Management System");

// ★Employee can work on 2 projects, add project references into it

employee.getProjects().add(project1);

employee.getProjects().add(project2);

// Add employee reference in the projects

project1.getEmployees().add(employee);

project2.getEmployees().add(employee);

// Call DAO to save the Employee and Project

employeeDao.saveEmployee(employee);

**// TODO: add you code here… END**

// Commit transaction

transaction.commit();

} catch (Exception e) {

if (transaction != null) {

transaction.rollback();

}

e.printStackTrace();

}

}

}

* JUnitTest class

package fa.trainning.dao;

import org.junit.Assert;

import org.junit.Test;

import org.junit.runner.RunWith;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.test.context.ContextConfiguration;

import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

import fa.trainning.entities.Employee;

import fa.trainning.entities.Project;

import fa.trainning.dao.EmployeeDao;

@RunWith(SpringJUnit4ClassRunner.class)

public class EmployeeDaoTest {

@Autowired

EmployeeDao employeeDao;

**@Test**

public void **saveEmployeeTest**() {

// INPUT

// Create an employee

Employee employee = new Employee();

employee.setFirstName("Anna");

employee.setLastName("Hook");

// Create a project

project project1 = new Project();

project.setTitle("Employee Management System");

// Create other project

project project2 = new Project();

project1.setTitle("Content Management System");

// ★Employee can work on 2 projects, add project references into it

employee.getProjects().add(project1);

employee.getProjects().add(project2);

// Add employee reference in the projects

project1.getEmployees().add(employee);

project2.getEmployees().add(employee);

// PROCCESS

// Call DAO to save the Employee and Project

employeeDao.saveEmployee(employee);

// VERIFY

Employee employeeOut = employeeDao.**getEmployeeWithName**("Anna");

// ↑getEmployeeWithName method must implemented in DAO

Assert.assertNotNull(employee);

Assert.assertEquals("Hook", employeeOut.getLastName());

Assert.assertEquals(2, employeeOut.getProjects().size());

}

}

**-- THE END --**