* Hibernate manytomany mapping is creating using join table also called linked table which has foreignkey from both the table .
* Owner Side has @manytomany annotation with joincolumn and inversejoincolumn for owner and inverse end resp.
* Inverse end has @manytomany annotation with mappedby to reference attribute on owner side.

|  |
| --- |
| Implementation Details |

package org.learnhibernatein5days.on.java.model;

import java.io.Serializable;

import java.util.HashSet;

import java.util.Set;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.JoinTable;

import javax.persistence.ManyToMany;

import javax.persistence.Table;

@Entity

@Table(name = "users")

public class User implements Serializable {

private static final long serialVersionUID = 5313493413859894403L;

@Id

@GeneratedValue

private long id;

@Column(nullable = false)

private String userId;

@Column(nullable = false, length = 50)

private String firstName;

@Column(nullable = false, length = 50)

private String lastName;

@Column(nullable = false, length = 120)

private String email;

@ManyToMany(cascade = { CascadeType.PERSIST }, fetch = FetchType.LAZY)

@JoinTable(name = "users\_roles", joinColumns = @JoinColumn(name = "users\_id"), inverseJoinColumns = @JoinColumn(name = "roles\_id"))

private Set<Role> roles = new HashSet<Role>();

public long getId() {

return id;

}

public void setId(long id) {

this.id = id;

}

public String getUserId() {

return userId;

}

public void setUserId(String userId) {

this.userId = userId;

}

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 getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public Set<Role> getRoles() {

return roles;

}

public void setRoles(Set<Role> roles) {

this.roles = roles;

}

/\* Utility Methods \*/

public void addRole(Role role) {

this.roles.add(role);

role.getUsers().add(this);

}

public void removeRole(Role role) {

this.roles.remove(role);

role.getUsers().remove(this);

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append("User [userId=");

builder.append(userId);

builder.append(", firstName=");

builder.append(firstName);

builder.append(", lastName=");

builder.append(lastName);

builder.append(", email=");

builder.append(email);

builder.append("]");

return builder.toString();

}

}

package org.learnhibernatein5days.on.java.model;

import java.io.Serializable;

import java.util.HashSet;

import java.util.Set;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.ManyToMany;

import javax.persistence.Table;

@Entity

@Table(name = "roles")

public class Role implements Serializable {

private static final long serialVersionUID = 5605260522147928803L;

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private long id;

@Column(nullable = false, length = 20)

private String name;

@ManyToMany(mappedBy = "roles")

private Set<User> users = new HashSet<User>();

public Role() {

}

public Role(String name) {

this.name = name;

}

public long getId() {

return id;

}

public void setId(long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Set<User> getUsers() {

return users;

}

public void setUsers(Set<User> users) {

this.users = users;

}

/\* Utility Methods \*/

public void addUser(User user) {

this.users.add(user);

user.getRoles().add(this);

}

public void removeRole(User user) {

this.users.remove(user);

user.getRoles().remove(this);

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

builder.append("Role [name=");

builder.append(name);

builder.append("]");

return builder.toString();

}

}

log.info("... bidirectionalManyToMany ...");

// add a new user and 2 roles to the user

EntityManager em = emf.createEntityManager();

em.getTransaction().begin();

Role roleAdmin = **new** Role("ROLE\_ADMIN");

Role roleUser = **new** Role("ROLE\_USER");

User user = **new** User();

user.setFirstName("Bhavesh");

user.setUserId("1223DFSFDF");

user.setLastName("Patel");

user.setEmail("emailtobhavesh@gmail.com");

user.addRole(roleAdmin);

user.addRole(roleUser);

em.persist(user);

em.getTransaction().commit();

em.close();

|  |
| --- |
| Query |

// Querying the users

EntityManager em = emf.createEntityManager();

em.getTransaction().begin();

/\* Just Getting User \*/

User user = em.find(User.**class**, 1L);

System.***out***.println("Role with user 1" + user.getRoles());

/\*\*\*\*\*\*\*\*\*\*\*\*\*\* N + 1 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

System.***out***.println("/\* N + 1 Problem \*/");

String hql = "SELECT u FROM User u where u.id = :userId ";

Query query = em.createQuery(hql).setParameter("userId", 1L);

List<User> users = query.getResultList();

System.***out***.println(users);

**for** (User u : users) {

Set<Role> roles = u.getRoles();

**for** (Iterator<Role> iterator = roles.iterator(); iterator.hasNext();) {

Role role = iterator.next();

System.***out***.println("Role \*\*\*\*\*\*\*\*\*\*\*\*\*" + role.getName());

}

}

/\*\*\*\*\*\*\*Resolve N + 1 Problem \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

System.***out***.println("/\*\*\*\*\*\*\*Resolve N + 1 Problem \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/");

hql ="SELECT u FROM User u LEFT JOIN FETCH u.roles where u.id = :userId ";

Query N1Query = em.createQuery(hql).setParameter("userId", 1L);

List<User> n1users = query.getResultList();

**for** (User u : n1users)

{

System.***out***.println( "Roles for this User" + u.getRoles());

}

em.getTransaction().commit();

em.close();

|  |
| --- |
| Pointer for Better Performance |