Hibernate Association Mapping

Till now we have created a simple Object and persist or retrieve that Object from Database. But in real life Objects are not so simple it has the has-A relationship with other Objects which may have other relationship so always there is an Object graph and we have to deal with that graph.

Now, shift our view to the database, In real life, we can’t found a table which does not have a relationship with another table by the foreign key.

As Hibernate objective is transform the RDBMS relation to Object oriented relation so somehow we have to manage this association between tables.

There are many shades of relation in the RDBMS.

Please found the below table for different type of association

|  |  |
| --- | --- |
| **Association Type** | **Description** |
| **Many-To-One** | **Owner table’s many rows mapped to foreign key(one row) of reference table. Like many employees work in same department** |
| **One-to-One** | **Owner table’s one row mapped to primary key of reference table(one row) . Like one Indian citizen has only one Aadhar card.** |
| **One-To-Many** | **Owner table’s one row mapped to primary keys of reference table(many rows) . Like one Department has many Employees.** |
| **Many-To-Many** | **Owner table’s many rows mapped to primary key of reference table(many rows) . Like One person can have multiple hobbies and multiple persons can have same hobby.** |

To express these associations in Object oriented programming/java, Hibernate introduces different annotations by using those annotations on the properties level with proper values, Hibernate understands How One entity should map with Other and on the fly resolves the mapping and returns an Object which will hold the associated Objects. So Developers get relief from manually mapping different entities.

In this chapter, we will see how different mappings can be done?

**Many To One**

Suppose we want to map players with Country .

So if we view the relation from Player perspective we can see many players are compete for Indian Cricket team. So this is a many to one relation where many players are playing for India.

**Create Country Entity**

*Country Entity mapped with Country table and id act as the foreign key of player table*

package com.example.hibernate.entity;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

import javax.persistence.Table;

import javax.persistence.UniqueConstraint;

@Entity

@Table(name = "COUNTRY", uniqueConstraints = {@UniqueConstraint(columnNames = "ID")})

public class Country {

@Id

private Long id;

@Column(name = "COUNTRY\_NAME", unique = true, nullable = false, length = 100)

private String 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;

}

@Override

public String toString() {

return "Country [id=" + id + ", name=" + name + "]";

}

}

**Create Player Table**

Player entity mapped with Player table, Please pay attention to the @ManytoOne mapping we define cascade =Persists that means if insert operation done on Player table is cascaded to Country table so we have to not manually save the Country. The most important tag annotation is ***@JoinColumn(name="COUNTRY\_ID",referencedColumnName="ID")*** here we define Player table’s **COUNTRY\_ID** column (written in **name** attribute) mapped with **Id** column of Country table, which is written in **referencedColumnName** attribute.

package com.example.hibernate.entity;

import javax.persistence.CascadeType;

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;

import javax.persistence.UniqueConstraint;

@Entity

@Table(name = "PLAYER", uniqueConstraints = {@UniqueConstraint(columnNames = "ID")})

public class Player {

@Id

private long id;

@Column(name = "PLAYER\_NAME", unique = true, nullable = false, length = 100)

private String name;

@Column(name = "SPORTS\_NAME", unique = false, nullable = false, length = 100)

private String sports;// Not separate Sports in a different table for sake of simplicity

@ManyToOne(cascade = CascadeType.ALL)

@JoinColumn(name="COUNTRY\_ID",referencedColumnName="ID")

private Country country;

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 String getSports() {

return sports;

}

public void setSports(String sports) {

this.sports = sports;

}

public Country getCountry() {

return country;

}

public void setCountry(Country country) {

this.country = country;

}

@Override

public String toString() {

return "Player [id=" + id + ", name=" + name + ", sports=" + sports + ", country=" + country + "]";

}

}

**Hibernate.cfg.xml**

<?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>

<property name="hibernate.archive.autodetection">class,hbm</property>

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

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

<property name="hibernate.connection.driver\_class">org.hsqldb.jdbcDriver</property>

<property name="hibernate.connection.username">sa</property>

<!-- <property name="hibernate.connection.password">1</property>-->

<property name="hibernate.connection.url">jdbc:hsqldb:mem:hibernateTutorial</property>

<property name="hibernate.hbm2ddl.auto">create</property>

<mapping class="com.example.hibernate.entity.Country"/>

<mapping class="com.example.hibernate.entity.Player"/>

</session-factory>

</hibernate-configuration>

**Test class**

Here we first insert a Country then Create two players and associate it with same Country after that we insert these players into Database. After successful insertion we again fetch the players with it’s details and associated country.

package com.example.hibernate.test;

import org.hibernate.Session;

import com.example.hibernate.core.HibernateUtil;

import com.example.hibernate.entity.Country;

import com.example.hibernate.entity.Player;

public class AssociationTest {

public static void main(String[] args) {

AssociationTest test = new AssociationTest();

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

test.insertCountry();

test.insertPlayers();

Player player = test.findEntityById(1L, Player.class, session);

Player player2 = test.findEntityById(2L, Player.class, session);

System.out.println(player);

System.out.println(player2);

session.close();

}

public void insertCountry() {

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

session.beginTransaction();

Country country = new Country();

country.setId(1L);;

country.setName("INDIA");

session.save(country);

session.getTransaction().commit();

session.close();

}

public void insertPlayers() {

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

session.beginTransaction();

Player player = new Player();

Country country = this.findEntityById(1L, Country.class , session);

player.setId(1L);;

player.setName("Virat Kohli");

player.setSports("Cricket");

player.setCountry(country);

Player player2 = new Player();

player2.setId(2L);;

player2.setName("MS Dhoni");

player2.setSports("Cricket");

player2.setCountry(country);

session.save(player);

session.save(player2);

session.getTransaction().commit();

session.close();

}

@SuppressWarnings("unchecked")

public <T> T findEntityById(Long id,Class<T> clazz,Session session) {

return (T) session.get(clazz, id);

}

}

**Output**

Hibernate: alter table PLAYER drop constraint FK\_lk8v8u2e4ohcoxtm7qw2ojd5v

Hibernate: drop table COUNTRY if exists

Hibernate: drop table PLAYER if exists

Hibernate: create table COUNTRY (id bigint not null, COUNTRY\_NAME varchar(100) not null, primary key (id))

Hibernate: create table PLAYER (id bigint not null, PLAYER\_NAME varchar(100) not null, SPORTS\_NAME varchar(100) not null, COUNTRY\_ID bigint, primary key (id))

Hibernate: alter table COUNTRY add constraint UK\_jmix1b5991708djqkulvtu8ob unique (COUNTRY\_NAME)

Hibernate: alter table Employee add constraint UK\_ardf0f11mfa6tujs3hflthwdv unique (EMAIL)

Hibernate: alter table PLAYER add constraint UK\_oku1rvhcq6n0rqdej1ubn150a unique (PLAYER\_NAME)

Hibernate: alter table PLAYER add constraint FK\_lk8v8u2e4ohcoxtm7qw2ojd5v foreign key (COUNTRY\_ID) references COUNTRY

Hibernate: insert into COUNTRY (COUNTRY\_NAME, id) values (?, ?)

Hibernate: select country0\_.id as id1\_0\_0\_, country0\_.COUNTRY\_NAME as COUNTRY\_2\_0\_0\_ from COUNTRY country0\_ where country0\_.id=?

Hibernate: insert into PLAYER (COUNTRY\_ID, PLAYER\_NAME, SPORTS\_NAME, id) values (?, ?, ?, ?)

Hibernate: insert into PLAYER (COUNTRY\_ID, PLAYER\_NAME, SPORTS\_NAME, id) values (?, ?, ?, ?)

Hibernate: select player0\_.id as id1\_2\_0\_, player0\_.COUNTRY\_ID as COUNTRY\_4\_2\_0\_, player0\_.PLAYER\_NAME as PLAYER\_N2\_2\_0\_, player0\_.SPORTS\_NAME as SPORTS\_N3\_2\_0\_, country1\_.id as id1\_0\_1\_, country1\_.COUNTRY\_NAME as COUNTRY\_2\_0\_1\_ from PLAYER player0\_ left outer join COUNTRY country1\_ on player0\_.COUNTRY\_ID=country1\_.id where player0\_.id=?

Hibernate: select player0\_.id as id1\_2\_0\_, player0\_.COUNTRY\_ID as COUNTRY\_4\_2\_0\_, player0\_.PLAYER\_NAME as PLAYER\_N2\_2\_0\_, player0\_.SPORTS\_NAME as SPORTS\_N3\_2\_0\_, country1\_.id as id1\_0\_1\_, country1\_.COUNTRY\_NAME as COUNTRY\_2\_0\_1\_ from PLAYER player0\_ left outer join COUNTRY country1\_ on player0\_.COUNTRY\_ID=country1\_.id where player0\_.id=?

**Actual Output :**

**Player [id=1, name=Virat Kohli, sports=Cricket, country=Country [id=1, name=INDIA]]**

**Player [id=2, name=MS Dhoni, sports=Cricket, country=Country [id=1, name=INDIA]]**

**One To One**

Now, Suppose One country represents One badminton player in the Olympic. Now this is a One to One relation One Player represents One country.

This is same as many to One relation but instread of ManytoOne we have to use One to One annotation.

**Create BadmintonPlayer.java**

The Badminton player is same as Player but in the Country attribute we put a @OneToOne mapping and a @PrimaryKeyJoinColumn to tell hibernate it is a one to one mapping and we use Country’s primary key used as a foreign key to join Player and Country tables.

package com.example.hibernate.entity;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import javax.persistence.OneToOne;

import javax.persistence.PrimaryKeyJoinColumn;

import javax.persistence.Table;

import javax.persistence.UniqueConstraint;

@Entity

@Table(name = "PLAYER", uniqueConstraints = {@UniqueConstraint(columnNames = "ID")})

public class BadmintonPlayer {

@Id

private long id;

@Column(name = "PLAYER\_NAME", unique = true, nullable = false, length = 100)

private String name;

@Column(name = "SPORTS\_NAME", unique = false, nullable = false, length = 100)

private String sports;// Not separate Sports in a different table for sake of simplicity

@OneToOne

@PrimaryKeyJoinColumn

private Country country;

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 String getSports() {

return sports;

}

public void setSports(String sports) {

this.sports = sports;

}

public Country getCountry() {

return country;

}

public void setCountry(Country country) {

this.country = country;

}

@Override

public String toString() {

return "BadmintonPlayer [id=" + id + ", name=" + name + ", sports=" + sports + ", country=" + country + "]";

}

}

**No Changes in Country class.**

**Craete a Test class to test the result**

package com.example.hibernate.test;

import org.hibernate.Session;

import com.example.hibernate.core.HibernateUtil;

import com.example.hibernate.entity.BadmintonPlayer;

import com.example.hibernate.entity.Country;

import com.example.hibernate.entity.Player;

public class AssociationTest {

public static void main(String[] args) {

AssociationTest test = new AssociationTest();

test.oneToOne();

}

public void oneToOne(){

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

insertCountry();

insertBadmintonPlayer();

Player player = findEntityById(1L, Player.class, session);

System.out.println(player);

session.close();

}

public void insertCountry() {

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

session.beginTransaction();

Country country = new Country();

country.setId(1L);;

country.setName("INDIA");

session.save(country);

session.getTransaction().commit();

session.close();

}

public void insertBadmintonPlayer() {

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

session.beginTransaction();

BadmintonPlayer player = new BadmintonPlayer();

Country country = this.findEntityById(1L, Country.class , session);

player.setId(1L);;

player.setName("PV SINDHU");

player.setSports("Badminton");

player.setCountry(country);

session.save(player);

session.getTransaction().commit();

session.close();

}

@SuppressWarnings("unchecked")

public <T> T findEntityById(Long id,Class<T> clazz,Session session) {

return (T) session.get(clazz, id);

}

}

**Output :**

Hibernate: alter table PLAYER drop constraint FK\_lk8v8u2e4ohcoxtm7qw2ojd5v

Hibernate: drop table COUNTRY if exists

Hibernate: drop table PLAYER if exists

Hibernate: create table COUNTRY (id bigint not null, COUNTRY\_NAME varchar(100) not null, primary key (id))

Hibernate: create table PLAYER (id bigint not null, PLAYER\_NAME varchar(100) not null, SPORTS\_NAME varchar(100) not null, COUNTRY\_ID bigint, primary key (id))

Hibernate: alter table COUNTRY add constraint UK\_jmix1b5991708djqkulvtu8ob unique (COUNTRY\_NAME)

Hibernate: alter table Employee add constraint UK\_ardf0f11mfa6tujs3hflthwdv unique (EMAIL)

Hibernate: alter table PLAYER add constraint UK\_oku1rvhcq6n0rqdej1ubn150a unique (PLAYER\_NAME)

Hibernate: alter table PLAYER add constraint FK\_lk8v8u2e4ohcoxtm7qw2ojd5v foreign key (COUNTRY\_ID) references COUNTRY

Hibernate: insert into COUNTRY (COUNTRY\_NAME, id) values (?, ?)

Hibernate: select country0\_.id as id1\_0\_0\_, country0\_.COUNTRY\_NAME as COUNTRY\_2\_0\_0\_ from COUNTRY country0\_ where country0\_.id=?

Hibernate: insert into PLAYER (PLAYER\_NAME, SPORTS\_NAME, id) values (?, ?, ?)

Hibernate: select badmintonp0\_.id as id1\_2\_0\_, badmintonp0\_.PLAYER\_NAME as PLAYER\_N2\_2\_0\_, badmintonp0\_.SPORTS\_NAME as SPORTS\_N3\_2\_0\_, country1\_.id as id1\_0\_1\_, country1\_.COUNTRY\_NAME as COUNTRY\_2\_0\_1\_ from PLAYER badmintonp0\_ left outer join COUNTRY country1\_ on badmintonp0\_.id=country1\_.id where badmintonp0\_.id=?

**Actual Output : BadmintonPlayer [id=1, name=PV SINDHU, sports=Badminton, country=Country [id=1, name=INDIA]]**

**One To Many**

Suppose we want fetch the Player list for a Country.

So if we view the relation from Country perspective we can realize one Country can have many players. One Country entity is associated with many Players this is called One to many relation we will store the associated entitylist in one of the java Collections.

Let see the Example

**Create Player class**

package com.example.hibernate.entity;

import javax.persistence.CascadeType;

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;

import javax.persistence.UniqueConstraint;

@Entity

@Table(name = "PLAYER", uniqueConstraints = {@UniqueConstraint(columnNames = "ID")})

public class Player {

@Id

private long id;

@Column(name = "PLAYER\_NAME", unique = true, nullable = false, length = 100)

private String name;

@Column(name = "SPORTS\_NAME", unique = false, nullable = false, length = 100)

private String sports;// Not separate Sports in a different table for sake of simplicity

@ManyToOne(cascade = CascadeType.PERSIST)

@JoinColumn(name="COUNTRY\_ID",referencedColumnName="ID")

private Country country;

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 String getSports() {

return sports;

}

public void setSports(String sports) {

this.sports = sports;

}

public Country getCountry() {

return country;

}

public void setCountry(Country country) {

this.country = country;

}

@Override

public String toString() {

return "Player [id=" + id + ", name=" + name + ", sports=" + sports + "]";

}

}

**Create Country entity**

Pay attention to @OneToMany(mappedBy="country") mapping, here we use mappedBy property which is the name of the java property of Player class, or we can say the Joining column

package com.example.hibernate.entity;

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;

import javax.persistence.UniqueConstraint;

@Entity

@Table(name = "COUNTRY", uniqueConstraints = {@UniqueConstraint(columnNames = "ID")})

public class Country {

@Id

private Long id;

@Column(name = "COUNTRY\_NAME", unique = true, nullable = false, length = 100)

private String name;

@OneToMany(mappedBy="country")

private Set<Player> playerList;

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<Player> getPlayerList() {

return playerList;

}

public void setPlayerList(Set<Player> playerList) {

this.playerList = playerList;

}

@Override

public String toString() {

return "Country [id=" + id + ", name=" + name + ", playerList=" + playerList + "]";

}

}

**Now Test the Example**

package com.example.hibernate.test;

import org.hibernate.Session;

import com.example.hibernate.core.HibernateUtil;

import com.example.hibernate.entity.BadmintonPlayer;

import com.example.hibernate.entity.Country;

import com.example.hibernate.entity.Player;

public class AssociationTest {

public static void main(String[] args) {

AssociationTest test = new AssociationTest();

test.oneToMany();

}

public void oneToMany(){

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

insertCountry();

insertPlayers();

Country country = findEntityById(1L, Country.class, session);

System.out.println(country);

session.close();

}

public void insertCountry() {

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

session.beginTransaction();

Country country = new Country();

country.setId(1L);;

country.setName("INDIA");

session.save(country);

session.getTransaction().commit();

session.close();

}

public void insertPlayers() {

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

session.beginTransaction();

Player player = new Player();

Country country = this.findEntityById(1L, Country.class , session);

player.setId(1L);;

player.setName("Virat Kohli");

player.setSports("Cricket");

player.setCountry(country);

Player player2 = new Player();

player2.setId(2L);;

player2.setName("MS Dhoni");

player2.setSports("Cricket");

player2.setCountry(country);

session.save(player);

session.save(player2);

session.getTransaction().commit();

session.close();

}

@SuppressWarnings("unchecked")

public <T> T findEntityById(Long id,Class<T> clazz,Session session) {

return (T) session.get(clazz, id);

}

}

**Output:**

Hibernate: alter table PLAYER drop constraint FK\_lk8v8u2e4ohcoxtm7qw2ojd5v

Hibernate: drop table COUNTRY if exists

Hibernate: drop table PLAYER if exists

Hibernate: create table COUNTRY (id bigint not null, COUNTRY\_NAME varchar(100) not null, primary key (id))

Hibernate: create table PLAYER (id bigint not null, PLAYER\_NAME varchar(100) not null, SPORTS\_NAME varchar(100) not null, COUNTRY\_ID bigint, primary key (id))

Hibernate: alter table COUNTRY add constraint UK\_jmix1b5991708djqkulvtu8ob unique (COUNTRY\_NAME)

Hibernate: alter table Employee add constraint UK\_ardf0f11mfa6tujs3hflthwdv unique (EMAIL)

Hibernate: alter table PLAYER add constraint UK\_oku1rvhcq6n0rqdej1ubn150a unique (PLAYER\_NAME)

Hibernate: alter table PLAYER add constraint FK\_lk8v8u2e4ohcoxtm7qw2ojd5v foreign key (COUNTRY\_ID) references COUNTRY

======================================

Hibernate: insert into COUNTRY (COUNTRY\_NAME, id) values (?, ?)

Hibernate: select country0\_.id as id1\_0\_0\_, country0\_.COUNTRY\_NAME as COUNTRY\_2\_0\_0\_ from COUNTRY country0\_ where country0\_.id=?

Hibernate: insert into PLAYER (COUNTRY\_ID, PLAYER\_NAME, SPORTS\_NAME, id) values (?, ?, ?, ?)

Hibernate: insert into PLAYER (COUNTRY\_ID, PLAYER\_NAME, SPORTS\_NAME, id) values (?, ?, ?, ?)

Hibernate: select country0\_.id as id1\_0\_0\_, country0\_.COUNTRY\_NAME as COUNTRY\_2\_0\_0\_ from COUNTRY country0\_ where country0\_.id=?

Hibernate: select playerlist0\_.COUNTRY\_ID as COUNTRY\_4\_0\_0\_, playerlist0\_.id as id1\_2\_0\_, playerlist0\_.id as id1\_2\_1\_, playerlist0\_.COUNTRY\_ID as COUNTRY\_4\_2\_1\_, playerlist0\_.PLAYER\_NAME as PLAYER\_N2\_2\_1\_, playerlist0\_.SPORTS\_NAME as SPORTS\_N3\_2\_1\_ from PLAYER playerlist0\_ where playerlist0\_.COUNTRY\_ID=?

===============================================================

**Actual Output : Country [id=1, name=INDIA, playerList=[Player [id=2, name=MS Dhoni, sports=Cricket], Player [id=1, name=Virat Kohli, sports=Cricket]]]**

**Many to Many**

Think about a Situation When a Plyer may play in multiple teams in various T20 League, and Each tam contains multiple player so it is a many to many relationship.

In this section we will see how create a many to many relationship.

Say Sunil Narine Play for Trinidad and KKR

So creating a Cricket Player entity

package com.example.hibernate.entity;

import java.util.HashSet;

import java.util.Set;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.JoinTable;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToMany;

import javax.persistence.Table;

import javax.persistence.UniqueConstraint;

@Entity

@Table(name = "PLAYER", uniqueConstraints = {@UniqueConstraint(columnNames = "ID")})

public class CricketPlayer {

@Id

private long id;

@Column(name = "PLAYER\_NAME", unique = true, nullable = false, length = 100)

private String name;

@Column(name = "SPORTS\_NAME", unique = false, nullable = false, length = 100)

private String sports;// Not separate Sports in a different table for sake of simplicity

@ManyToMany

@JoinTable(name = "PLAYER\_TEAM", joinColumns = {

@JoinColumn(name = "PLAYER\_ID", nullable = false, updatable = false) },

inverseJoinColumns = { @JoinColumn(name = "TEAM\_ID",

nullable = false, updatable = false) })

private Set<Team> teams = new HashSet<Team>();

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 String getSports() {

return sports;

}

public void setSports(String sports) {

this.sports = sports;

}

public void addTeam(Team team) {

teams.add(team);

}

@Override

public String toString() {

return "CricketPlayer [id=" + id + ", name=" + name + ", sports=" + sports + ", team=" + teams + "]";

}

}

Pay attention to many to many mapping over teams property here we use three different annotations

**@JoinTable** : By this annotation we tell hibernate there is intermediate table called “Player\_Team” exists which map the Team Id with Player Id. This table only maintains relation between CricketPlayer and Team

**@joinColumns** : It says intermediate table’s player\_Id column map with id of player table.

**@inverseJoinColumns** : It says intermediate table’s team\_Id column map with id of Team table.

By this we achieve many to many relation.

**Create Team entity**

package com.example.hibernate.entity;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.Table;

import javax.persistence.UniqueConstraint;

@Entity

@Table(name = "TEAM", uniqueConstraints = {@UniqueConstraint(columnNames = "ID")})

public class Team {

@Id

private Long id;

@Column(name = "TEAM\_NAME", unique = true, nullable = false, length = 100)

private String 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;

}

@Override

public String toString() {

return "Country [id=" + id + ", name=" + name +"]";

}

}

**Add new entities to mapping file**

<?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>

<property name="hibernate.archive.autodetection">class,hbm</property>

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

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

<property name="hibernate.connection.driver\_class">org.hsqldb.jdbcDriver</property>

<property name="hibernate.connection.username">sa</property>

<!-- <property name="hibernate.connection.password">1</property>-->

<property name="hibernate.connection.url">jdbc:hsqldb:mem:hibernateTutorial</property>

<property name="hibernate.hbm2ddl.auto">create</property>

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

<mapping class="com.example.hibernate.entity.Country"/>

<mapping class="com.example.hibernate.entity.Player"/>

<mapping class="com.example.hibernate.entity.BadmintonPlayer"/>

<mapping class="com.example.hibernate.entity.CricketPlayer"/>

<mapping class="com.example.hibernate.entity.Team"/>

</session-factory>

</hibernate-configuration>

**Test the solution**

package com.example.hibernate.test;

import org.hibernate.Session;

import com.example.hibernate.core.HibernateUtil;

import com.example.hibernate.entity.BadmintonPlayer;

import com.example.hibernate.entity.Country;

import com.example.hibernate.entity.CricketPlayer;

import com.example.hibernate.entity.Player;

import com.example.hibernate.entity.Team;

public class AssociationTest {

public static void main(String[] args) {

AssociationTest test = new AssociationTest();

test.manyTomany();

}

public void manyTomany(){

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

insertTeams();

insertCricketPlayer();

CricketPlayer player = findEntityById(1L, CricketPlayer.class, session);

System.out.println(player);

session.close();

}

public void insertTeams() {

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

session.beginTransaction();

Team team = new Team();

team.setId(1L);;

team.setName("KKR");

Team team1 = new Team();

team1.setId(2L);

team1.setName("TRINIDAD");

session.save(team);

session.save(team1);

session.getTransaction().commit();

session.close();

}

public void insertCricketPlayer() {

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

session.beginTransaction();

CricketPlayer player = new CricketPlayer();

Team team1 = this.findEntityById(1L, Team.class , session);

Team team2 = this.findEntityById(2L, Team.class , session);

player.setId(1L);;

player.setName("Sunil Narine");

player.setSports("Cricket");

player.addTeam(team1);

player.addTeam(team2);

session.save(player);

session.getTransaction().commit();

session.close();

}

@SuppressWarnings("unchecked")

public <T> T findEntityById(Long id,Class<T> clazz,Session session) {

return (T) session.get(clazz, id);

}

}

**Output**

Hibernate: alter table PLAYER drop constraint FK\_lk8v8u2e4ohcoxtm7qw2ojd5v

Hibernate: alter table PLAYER\_TEAM drop constraint FK\_p9eab0ahqtdy7evoiljubxvlj

Hibernate: alter table PLAYER\_TEAM drop constraint FK\_ssl1bvxj4wp88tx7gl206a48x

Hibernate: drop table COUNTRY if exists

Hibernate: drop table Employee if exists

Hibernate: drop table PLAYER if exists

Hibernate: drop table PLAYER\_TEAM if exists

Hibernate: drop table TEAM if exists

Hibernate: create table COUNTRY (id bigint not null, COUNTRY\_NAME varchar(100) not null, primary key (id))

Hibernate: create table Employee (ID integer not null, EMAIL varchar(100) not null, NAME varchar(100) not null, primary key (ID))

Hibernate: create table PLAYER (id bigint not null, PLAYER\_NAME varchar(100) not null, SPORTS\_NAME varchar(100) not null, COUNTRY\_ID bigint, primary key (id))

Hibernate: create table PLAYER\_TEAM (PLAYER\_ID bigint not null, TEAM\_ID bigint not null, primary key (PLAYER\_ID, TEAM\_ID))

Hibernate: create table TEAM (id bigint not null, TEAM\_NAME varchar(100) not null, primary key (id))

Hibernate: alter table COUNTRY add constraint UK\_jmix1b5991708djqkulvtu8ob unique (COUNTRY\_NAME)

Hibernate: alter table Employee add constraint UK\_ardf0f11mfa6tujs3hflthwdv unique (EMAIL)

Hibernate: alter table PLAYER add constraint UK\_oku1rvhcq6n0rqdej1ubn150a unique (PLAYER\_NAME)

Hibernate: alter table TEAM add constraint UK\_29ddxpmundb2eq8o9mdbcbo6k unique (TEAM\_NAME)

Hibernate: alter table PLAYER add constraint FK\_lk8v8u2e4ohcoxtm7qw2ojd5v foreign key (COUNTRY\_ID) references COUNTRY

Hibernate: alter table PLAYER\_TEAM add constraint FK\_p9eab0ahqtdy7evoiljubxvlj foreign key (TEAM\_ID) references TEAM

Hibernate: alter table PLAYER\_TEAM add constraint FK\_ssl1bvxj4wp88tx7gl206a48x foreign key (PLAYER\_ID) references PLAYER

Hibernate: insert into TEAM (TEAM\_NAME, id) values (?, ?)

Hibernate: insert into TEAM (TEAM\_NAME, id) values (?, ?)

Hibernate: select team0\_.id as id1\_4\_0\_, team0\_.TEAM\_NAME as TEAM\_NAM2\_4\_0\_ from TEAM team0\_ where team0\_.id=?

Hibernate: select team0\_.id as id1\_4\_0\_, team0\_.TEAM\_NAME as TEAM\_NAM2\_4\_0\_ from TEAM team0\_ where team0\_.id=?

Hibernate: insert into PLAYER (PLAYER\_NAME, SPORTS\_NAME, id) values (?, ?, ?)

Hibernate: insert into PLAYER\_TEAM (PLAYER\_ID, TEAM\_ID) values (?, ?)

Hibernate: insert into PLAYER\_TEAM (PLAYER\_ID, TEAM\_ID) values (?, ?)

Hibernate: select cricketpla0\_.id as id1\_2\_0\_, cricketpla0\_.PLAYER\_NAME as PLAYER\_N2\_2\_0\_, cricketpla0\_.SPORTS\_NAME as SPORTS\_N3\_2\_0\_ from PLAYER cricketpla0\_ where cricketpla0\_.id=?

Hibernate: select teams0\_.PLAYER\_ID as PLAYER\_I1\_2\_0\_, teams0\_.TEAM\_ID as TEAM\_ID2\_3\_0\_, team1\_.id as id1\_4\_1\_, team1\_.TEAM\_NAME as TEAM\_NAM2\_4\_1\_ from PLAYER\_TEAM teams0\_ inner join TEAM team1\_ on teams0\_.TEAM\_ID=team1\_.id where teams0\_.PLAYER\_ID=?

================================================

**Actual Output : CricketPlayer [id=1, name=Sunil Narine, sports=Cricket, team=[Country [id=2, name=TRINIDAD], Country [id=1, name=KKR]]]**