Hibernate-2017

<http://javadata.blogspot.in/2011/07/uni-directionalbi-directionalcascade.html>

# Uni directional,Bi directional,Cascade and MappedBy Hibernate

**Unidirectional and Bidirectional relationship**

In order to understand this relationship, take the case of Book Library. Book Library contains collection of books. So book contains foreign key of book library to maintain the relationship.

**Unidirectional relationship refers to the case when relationship between two objects can be accessed by one way only. i.e**From Book Library we can access books not vice versa  
class Book{  
     Long bookId;  
     String bookName;  
      …........  
}  
class BookLibrary{  
      Long libraryId;  
      String libraryName;  
       Set<Book> bookSet;  
}  
**Bidirectional relationship refers to the case when relationship between two objects can be accessed both ways. i.e** From Book Library we can access collection of Books and Book Object will have reference to BookLibrary

class Book{  
     Long bookId;  
     String bookName;  
     BookLibrary library;  
      …........  
}

Though in both the cases for relationship to occur there would be foreign key mapping in Book Table corresponding to Book Library.

**Cascade By Attribute**

Taking the Book Library example further, lets assume we have a form/web page where we input the information regarding the library and books. So we have a library object containing library information and collection of books. In traditional JDBC approach,we will persist the Book Library first ,get the corresponding primary key and then iterate through book list and persist it one by one by adding the foreign key attribute. All this work has to be done by developer.

Similarly for updates and deletes we have to follow the same procedure.

**Cascade By attribute provided by Hibernate keeps the above mechanism abstract to the developer. Developer persists/updates/deletes the top level object and changes are propagated to the associated entities depending upon the value of cascade by attribute**. So taking the above example if we provide the cascade by attribute to the book set as ALL. Whatever operations are done on BookLibrary object ,it will be passed on to the book collection object also. Developer need not have to code this,it will happen automatically.

So by providing the right cascade attributes we can govern the flow of changes from parent to child entities.

class BookLibrary{  
      Long libraryId;  
      String libraryName;  
       …...  
    **@Cascade(value = { CascadeType.ALL })**       Set<Book> bookSet;  
}

**MappedBy Attribute**

MappedBy Attribute is the most misunderstood concept. It comes into action when we have bi directional relationship between the entities. In relational database we have foreign key attribute which identifies the parent and child tables.

However when we have bi-directional relationship,in order to provide the identification regarding the same hibernate has come up with mappedBy attribute.

**Normally mappedBy attribute is provided on the object which does not contains foreign key.  
In hibernate terms,it means “I am not the owner of the relationship, ownership is governed by other entity'.**

Its the job of the associated entity on non mappedBy side to maintain the association. In case other entity does not do the association, there will be NULL in the foreign key column.

Inorder to understand it better lets take case of  parent and child.Child belongs to a parent and parent can have more than one child.Its a classical case of one to many relationship.

Here mappedBy attribute is at parent side.So its the responsibilty of child to maintain the realtionship.

create table ORM\_MappedBy\_Parent(

Id bigint Identity(1,1) primary key,

ParentName varchar(50) not null,

ParentAddr varchar(50)not null

)

create table ORM\_MappedBy\_Child(

ChildId bigint identity(1,1) primary key,

ChildName varchar(50) not null,

ParentId bigint null

)

alter table ORM\_MappedBy\_Child

ADD FOREIGN KEY (ParentId) REFERENCES ORM\_MappedBy\_Parent(Id);

Here ORM\_MappedBy\_Child contains the foriegn key.So  Parent will have collection of child elements.

<http://chathurangat.blogspot.in/2013/07/difference-between-unidirectional-and.html>

### Difference between Unidirectional and Bidirectional mapping in hibernate

**Unidirectional Mapping**  
When only one of the pair of entities contains a reference to the other, the association is unidirectional. for example, assume that there are two entities called **Teacher** and **Course**.  
In unidirectional mapping, the teacher will hold a reference for the course OR course will hold a reference for the teacher.(Not Both) It is mandatory that the references should not be mapped to the both directions and there should be only one direction(unidirectional) mapping.  
In unidirectional mapping, it will provide the navigational access only to one direction.  
**assumption: teacher can teach only once course and a given course can be taught only by one teacher**  
  
*unidirectional mapping  from lecturer to course*  
class Teacher {  
   private Long id;  
   private  String lecturerName;  
**private Course;**  
   //getter and setters   
}

class Course {  
   private Long id;  
   private String courseName;  
    //getter and setters   
}   
  
*unidirectional mapping  from course to lecturer*  
class Teacher {  
   private Long id;  
   private  String lecturerName;  
   //getter and setters   
}

class Course {  
   private Long id;  
   private String courseName;  
**private Teacher teacher;**  
    //getter and setters   
}

**Bidirectional Mapping**  
if the association between both entities are mutual, then it is known as bidirectional mapping. in bidirectional mapping, the lecturer should hold to a reference for the course and the same time the course should a reference to the lecturer.  
therefore in bi-directional mapping, it will provide the navigational access to the both directions.   
**assumption: teacher can teach only once course and a given course can be taught only by one teacher**  
  
class Teacher {  
   private Long id;  
   private  String lecturerName;  
**private Course;**  
   //getter and setters   
}  
  
class Course{  
   private Long id;  
   private String courseName;  
**private Teacher teacher;**  
    //getter and setters   
}

One To One Unidirectional Mapping (Student has a Subject)

# Maven Configuration (pom.xml)

<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>one-to-one-unidirectional3</groupId>

<artifactId>one-to-one-unidirectional3</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>one-to-one-unidirectional3</name>

<url>http://maven.apache.org</url>

<properties>

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

<spring.version>4.1.2.RELEASE</spring.version>

<spring.security.version>3.2.3.RELEASE</spring.security.version>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

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

<version>5.1.30</version>

</dependency>

<!-- C3P0 library -->

<dependency>

<groupId>com.mchange</groupId>

<artifactId>c3p0</artifactId>

<version>0.9.5</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>4.3.7.Final</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

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

<version>3.3</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

# SQL(ddl.sql)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*One To One Unidirectional\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Drop the table\*/

**SET** FOREIGN\_KEY\_CHECKS = 0;

**drop** **table** IF **EXISTS** Student;

**drop** **table** IF **EXISTS** Subject;

/\*Create the required tables\*/

**CREATE** **TABLE** IF **NOT** **EXISTS** Subject (

subjectId **int**(11) **NOT** **NULL** AUTO\_INCREMENT,

name **varchar**(255) **DEFAULT** **NULL**,

**PRIMARY** **KEY** (subjectId)

) ;

**CREATE** **TABLE** IF **NOT** **EXISTS** Student (

studentId **int**(11) **NOT** **NULL** AUTO\_INCREMENT,

firstName **varchar**(255) **DEFAULT** **NULL**,

subjectId **int**(11) **DEFAULT** **NULL**,

**PRIMARY** **KEY** (studentId),

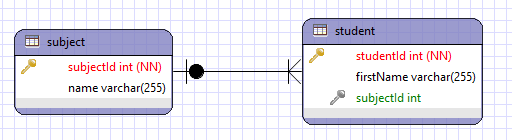
**FOREIGN** **KEY** (subjectId) **REFERENCES** Subject (subjectId)

) ;

/\*Select the created table\*/

**select** \* **from** Student;

**select** \* **from** Subject;



**Student Table** **Subject Table**

# Spring configuration(src/main/resources/app-context.xml)

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:context=*"http://www.springframework.org/schema/context"*

xmlns:aop=*"http://www.springframework.org/schema/aop"* xmlns:jee=*"http://www.springframework.org/schema/jee"*

xmlns:tx=*"http://www.springframework.org/schema/tx"* xmlns:jdbc=*"http://www.springframework.org/schema/jdbc"*

xmlns:osgi=*"http://www.springframework.org/schema/osgi"* xmlns:security=*"http://www.springframework.org/schema/security"*

xsi:schemaLocation=*"http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.0.xsd*

*http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd*

*http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-jee-3.0.xsd*

*http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.0.xsd*

*http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc-3.0.xsd*

*http://www.springframework.org/schema/osgi http://www.springframework.org/schema/osgi/spring-osgi.xsd*

*http://www.springframework.org/schema/security http://www.springframework.org/schema/security/spring-security-3.0.3.xsd"*>

<tx:annotation-driven transaction-manager=*"discussionTransactionManager"* />

<bean id=*"dataSourceInternal"* class=*"com.mchange.v2.c3p0.ComboPooledDataSource"*

destroy-method=*"close"*>

<property name=*"driverClass"* value=*"com.mysql.jdbc.Driver"* />

<property name=*"jdbcUrl"* value=*"jdbc:mysql://localhost/test"* />

<property name=*"user"* value=*"test"* />

<property name=*"password"* value=*"test"* />

<!-- these are C3P0 properties -->

<property name=*"acquireIncrement"* value=*"5"* />

<property name=*"initialPoolSize"* value=*"5"* />

<property name=*"minPoolSize"* value=*"5"* />

<property name=*"maxPoolSize"* value=*"20"* />

</bean>

<!-- This is the lazy DataSource proxy that interacts with the target DataSource once a real statement is sent to the database. Users use this DataSource to set up their Hibernate session factory, which in turn forces the Hibernate second-level cache and also everything that interacts with that Hibernate session factory to use it. -->

<bean id=*"dataSource"* class=*"org.springframework.jdbc.datasource.LazyConnectionDataSourceProxy"*>

<property name=*"targetDataSource"*><ref bean=*"dataSourceInternal"* /></property>

</bean>

<!-- <bean id="hibSessionFactory" class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean"> -->

<bean id=*"hibSessionFactory"* class=*"org.springframework.orm.hibernate4.LocalSessionFactoryBean"*>

<property name=*"dataSource"* ref=*"dataSource"* />

<!--<property name="hibernateProperties"> <value> hibernate.show\_sql=true

</value> </property> -->

<property name=*"hibernateProperties"*>

<value>

hibernate.id.new\_generator\_mappings=true,

hibernate.show\_sql=true

<!-- hibernate.hbm2ddl.auto=update -->

<!-- hibernate.current\_session\_context\_class=thread -->

</value>

</property>

<property name=*"annotatedClasses"*>

<list>

<value>com.ddlab.rnd.hibernate.Student</value>

<value>com.ddlab.rnd.hibernate.Subject</value>

</list>

</property>

</bean>

</beans>

# Java Files

## **Student.java**

**package** com.ddlab.rnd.hibernate;

**import** java.io.Serializable;

**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.OneToOne;

**import** javax.persistence.Table;

@Entity

@Table(name = "student")

**public** **class** Student **implements** Serializable {

**private** **static** **final** **long** ***serialVersionUID*** = -282016336815027846L;

**@Id**

**@GeneratedValue(strategy = GenerationType.*IDENTITY*)**

**@Column(name = "studentId")**

**private int studentId;**

**@Column(name = "firstName")**

**private String firstName;**

**/\***

**\* How will you get subject from Student table ?**

**\* By joining subjectId column from Student table**

**\*/**

**@OneToOne(cascade = CascadeType.*ALL*)**

**@JoinColumn(name = "subjectId")**

**private Subject subject;**

**public Student() {} //Required while getting data from database**

**public** Student(String firstName) {

**this**.firstName = firstName;

}

**public** **int** getStudentId() {

**return** studentId;

}

**public** **void** setStudentId(**int** studentId) {

**this**.studentId = studentId;

}

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** Subject getSubject() {

**return** subject;

}

**public** **void** setSubject(Subject subject) {

**this**.subject = subject;

}

}

## **Subject.java**

**package** com.ddlab.rnd.hibernate;

**import** java.io.Serializable;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

@Entity

@Table(name = "subject")

**public** **class** Subject **implements** Serializable {

**private** **static** **final** **long** ***serialVersionUID*** = -6494554948746566564L;

**@Id**

**@Column(name = "subjectId")**

**@GeneratedValue(strategy = GenerationType.*IDENTITY*)**

**private int subjectId;**

@Column(name = "name")

**private** String name;

**public Subject() {} //Required while getting data from database**

**public** Subject(String name) {

**this**.name = name;

}

**public** **int** getSubjectId() {

**return** subjectId;

}

**public** **void** setSubjectId(**int** subjectId) {

**this**.subjectId = subjectId;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

}

## **Test.java**

**package** com.ddlab.rnd.hibernate;

**import** org.hibernate.HibernateException;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

ApplicationContext context = **new** ClassPathXmlApplicationContext("app-context.xml");

SessionFactory sessionFactory = (SessionFactory) context.getBean("hibSessionFactory");

Session session = sessionFactory.openSession();

Transaction transaction = **null**;

**try** {

transaction = session.beginTransaction();

Subject subject1 = **new** Subject("Physics");

//First save Subject

session.save(subject1);

Student student1 = **new** Student("Deb");

student1.setSubject(subject1);

//Second save Student

session.save(student1);

transaction.commit();

} **catch** (HibernateException e) {

**if** (transaction != **null**)

transaction.rollback();

e.printStackTrace();

} **finally** {

session.close();

}

System.***out***.println("successfully saved into database");

}

}

## **TestQuery.java**

**package** com.ddlab.rnd.hibernate;

**import** java.util.List;

**import** org.hibernate.Query;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** TestQuery {

**public** **static** **void** showCustomer(Session session) {

**try** {

// String hql = "FROM Customer";

String hql = "FROM Student s where s.firstName =:firstName";

Query query = session.createQuery(hql);

query.setParameter("firstName", "Deb");

List results = query.list();

**for** (**int** i = 0; i < results.size(); i++) {

Student student = (Student) results.get(i);

System.***out***.println(student.getStudentId()+"---"+student.getFirstName()+"---"+student.getSubject().getSubjectId()+"---"+student.getSubject().getName());

}

} **catch** (Exception e) {

e.printStackTrace();

}

}

**public** **static** **void** showAddress(Session session) {

**try** {

String hql = "FROM Subject";

// String hql = "FROM Address C where C.lastName =:lastName";

Query query = session.createQuery(hql);

List results = query.list();

**for** (**int** i = 0; i < results.size(); i++) {

Subject subject = (Subject) results.get(i);

System.***out***.format("%10s%10s\n", subject.getSubjectId(),

subject.getName());

}

} **catch** (Exception e) {

e.printStackTrace();

}

}

**public** **static** **void** main(String[] args) {

ApplicationContext context = **new** ClassPathXmlApplicationContext("app-context.xml");

SessionFactory sessionFactory = (SessionFactory) context.getBean("hibSessionFactory");

Session session = sessionFactory.openSession();

*showCustomer*(session);

*showAddress*(session);

}

}

# Key to remember in case one-to-one unidirectional

@Entity

@Table(name = "student")

**public** **class** Student **implements** Serializable {

**/\***

**\* How will you get subject from Student table ?**

**\* By joining subjectId column from Student table**

**\*/**

**@OneToOne(cascade = CascadeType.*ALL*)**

**@JoinColumn(name = "subjectId")**

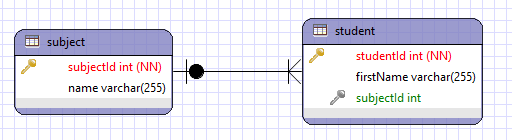
**private Subject subject;**

**public Student() {} //Required while getting data from database**

}

One To One Bidirectional Mapping (Student has a Subject)

# SQL



**Student Table** **Subject Table**

Same as pom.xml, app-context.xml

# Java Files

## **Student.java**

**package** com.ddlab.rnd.hibernate;

**import** java.io.Serializable;

**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.OneToOne;

**import** javax.persistence.Table;

@Entity

@Table(name = "student")

**public** **class** Student **implements** Serializable {

**private** **static** **final** **long** ***serialVersionUID*** = -282016336815027846L;

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

@Column(name = "studentId")

**private** **int** studentId;

@Column(name = "firstName")

**private** String firstName;

**/\***

**\* How will you get subject from Student table ?**

**\* By joining subjectId column from Student table**

**\*/**

**@OneToOne(cascade = CascadeType.*ALL*)**

**@JoinColumn(name = "subjectId")**

**private Subject subject;**

**public Student() {} //Required while getting data from database**

**public** Student(String firstName) {

**this**.firstName = firstName;

}

**public** **int** getStudentId() {

**return** studentId;

}

**public** **void** setStudentId(**int** studentId) {

**this**.studentId = studentId;

}

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** Subject getSubject() {

**return** subject;

}

**public** **void** setSubject(Subject subject) {

**this**.subject = subject;

}

}

## **Subject.java**

**package** com.ddlab.rnd.hibernate;

**import** java.io.Serializable;

**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.OneToOne;

**import** javax.persistence.Table;

@Entity

@Table(name = "subject")

**public** **class** Subject **implements** Serializable {

**private** **static** **final** **long** ***serialVersionUID*** = -6494554948746566564L;

@Id

@Column(name = "subjectId")

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

**private** **int** subjectId;

@Column(name = "name")

**private** String name;

**/\***

**\* How will you get Student from Subject**

**\* By using mappedBy as Subject does not direct relationship**

**\*/**

**@OneToOne(cascade = CascadeType.*ALL*, mappedBy = "subject")**

**private Student student;**

**public Subject() {} //Required while getting data from database**

**public** Subject(String name) {

**this**.name = name;

}

**public** **int** getSubjectId() {

**return** subjectId;

}

**public** **void** setSubjectId(**int** subjectId) {

**this**.subjectId = subjectId;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** Student getStudent() {

**return** student;

}

**public** **void** setStudent(Student student) {

**this**.student = student;

}

}

## **Test.java**

**package** com.ddlab.rnd.hibernate;

**import** org.hibernate.HibernateException;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** Test {

**public** **static** **void** saveStudent(SessionFactory sessionFactory) {

Session session = sessionFactory.openSession();

Transaction transaction = **null**;

**try** {

transaction = session.beginTransaction();

Student student1 = **new** Student("John Abraham");

Subject subject1 = **new** Subject("Chemistry");

subject1.setStudent(student1);

student1.setSubject(subject1);

//Save Student

session.save(student1);

transaction.commit();

} **catch** (HibernateException e) {

**if** (transaction != **null**)

transaction.rollback();

e.printStackTrace();

} **finally** {

session.close();

}

}

**public** **static** **void** saveSubject(SessionFactory sessionFactory) {

Session session = sessionFactory.openSession();

Transaction transaction = **null**;

**try** {

transaction = session.beginTransaction();

Student student1 = **new** Student("Vidya Balan");

Subject subject1 = **new** Subject("Biology");

subject1.setStudent(student1);

student1.setSubject(subject1);

//Save Subject

session.save(subject1);

transaction.commit();

} **catch** (HibernateException e) {

**if** (transaction != **null**) transaction.rollback();

e.printStackTrace();

} **finally** {

session.close();

}

}

**public** **static** **void** main(String[] args) {

ApplicationContext context = **new** ClassPathXmlApplicationContext("app-context.xml");

SessionFactory sessionFactory = (SessionFactory) context.getBean("hibSessionFactory");

***saveStudent*(sessionFactory);**

***saveSubject*(sessionFactory);**

System.***out***.println("successfully saved into database");

}

}

# Key to remember in case one-to-one Bidirectional

**@Entity**

**@Table(name = "student")**

**public class Student implements Serializable {**

**/\***

**\* How will you get subject from Student table ?**

**\* By joining subjectId column from Student table**

**\*/**

**@OneToOne(cascade = CascadeType.*ALL*)**

**@JoinColumn(name = "subjectId")**

**private Subject subject;**

**public Student() {} //Required while getting data from database**

**}**

**@Entity**

**@Table(name = "subject")**

**public class Subject implements Serializable {**

**/\***

**\* How will you get Student from Subject**

**\* By using mappedBy as Subject does not direct relationship**

**\*/**

**@OneToOne(cascade = CascadeType.*ALL*, mappedBy = "subject")**

**private Student student;**

**public Subject() {} //Required while getting data from database**

**}**

One To Many Unidirectional Mapping (Student has many Subjects)

Same as pom.xm, app-context.xml

# SQL

**drop** **table** IF **EXISTS** STUDENT\_SUBJECT;

**drop** **table** IF **EXISTS** STUDENT;

**drop** **table** IF **EXISTS** SUBJECT;

**create** **table** STUDENT (

student\_id BIGINT **NOT** **NULL** AUTO\_INCREMENT,

first\_name **VARCHAR**(30) **NOT** **NULL**,

**PRIMARY** **KEY** (student\_id)

);

**create** **table** SUBJECT (

subject\_id BIGINT **NOT** **NULL** AUTO\_INCREMENT,

name **VARCHAR**(30) **NOT** **NULL**,

**PRIMARY** **KEY** (subject\_id)

);

**CREATE** **TABLE** STUDENT\_SUBJECT (

student\_id BIGINT **NOT** **NULL**,

subject\_id BIGINT **NOT** **NULL**,

**PRIMARY** **KEY** (student\_id, subject\_id),

**CONSTRAINT** FK\_STUDENT **FOREIGN** **KEY** (student\_id) **REFERENCES** STUDENT (student\_id),

**CONSTRAINT** FK\_SUBJECT **FOREIGN** **KEY** (subject\_id) **REFERENCES** SUBJECT (subject\_id)

);

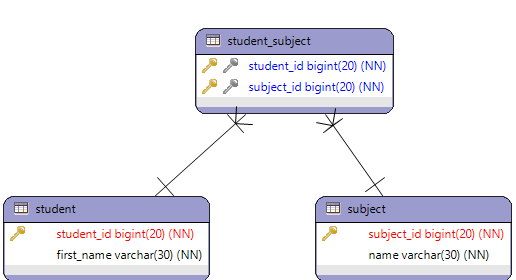


Table : **Student\_Subject** Table : **Student** Table : **Subject**

# Java Files

## **Student.java**

**package** com.ddlab.rnd.hibernate;

**import** java.util.ArrayList;

**import** java.util.List;

**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.JoinTable;

**import** javax.persistence.OneToMany;

**import** javax.persistence.Table;

**@Entity**

**@Table(name = "STUDENT")**

**public class Student** {

@Id

@GeneratedValue

@Column(name = "STUDENT\_ID")

**private** **long** id;

@Column(name = "FIRST\_NAME")

**private** String firstName;

**@OneToMany(cascade = CascadeType.*ALL*)**

**@JoinTable(name = "STUDENT\_SUBJECT",**

**joinColumns = { @JoinColumn(name = "STUDENT\_ID") },**

**inverseJoinColumns = { @JoinColumn(name = "SUBJECT\_ID") })**

**private List<Subject> subjects = new ArrayList<Subject>();**

**public** Student() {

}

**public** Student(String firstName) {

**this**.firstName = firstName;

}

**public** **long** getId() {

**return** id;

}

**public** **void** setId(**long** id) {

**this**.id = id;

}

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** List<Subject> getSubjects() {

**return** subjects;

}

**public** **void** setSubjects(List<Subject> subjects) {

**this**.subjects = subjects;

}

}

## **Subject.Java**

**package** com.ddlab.rnd.hibernate;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

**@Entity**

**@Table(name = "SUBJECT")**

**public class Subject** {

@Id

@GeneratedValue

@Column(name = "SUBJECT\_ID")

**private** **long** id;

@Column(name = "NAME")

**private** String name;

**public** Subject(){

}

**public** Subject(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;

}

}

## **Test1.java**

**package** com.ddlab.rnd.hibernate;

**import** org.hibernate.HibernateException;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** Test1 {

**public** **static** **void** main(String[] args) {

ApplicationContext context = **new** ClassPathXmlApplicationContext("app-context.xml");

SessionFactory sessionFactory = (SessionFactory) context.getBean("hibSessionFactory");

Session session = sessionFactory.openSession();

Transaction transaction = **null**;

**try** {

transaction = session.beginTransaction();

Student student1 = **new** Student("Sam");

Student student2 = **new** Student("Joshua");

Subject subject1 = **new** Subject("Economics");

Subject subject2 = **new** Subject("Politics");

Subject subject3 = **new** Subject("Foreign Affairs");

// Student1 have 3 subjects

student1.getSubjects().add(subject1);

student1.getSubjects().add(subject2);

student1.getSubjects().add(subject3);

// Student2 have 2 subjects

student2.getSubjects().add(subject1);

student2.getSubjects().add(subject2);

session.save(student1);

session.save(student2);

transaction.commit();

} **catch** (HibernateException e) {

**if** (transaction != **null**) transaction.rollback();

e.printStackTrace();

} **finally** {

session.close();

}

System.***out***.println("successfully saved into database");

}

}

# Key to remember in case of one-to-many Unidirectional

**@Entity**

**@Table(name = "STUDENT")**

**public class Student {**

**@OneToMany(cascade = CascadeType.*ALL*)**

**@JoinTable(name = "STUDENT\_SUBJECT",**

**joinColumns = { @JoinColumn(name = "STUDENT\_ID") },**

**inverseJoinColumns = { @JoinColumn(name = "SUBJECT\_ID") })**

**private List<Subject> subjects = new ArrayList<Subject>();**

**}**

One To Many Bidirectional Mapping (Student has many Subjects)

Same as pom.xml, app-context.xml

# SQL

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*One To Many Bidirectional\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Drop the table\*/

**SET** FOREIGN\_KEY\_CHECKS = 0;

**drop** **table** if **exists** subject;

**drop** **table** if **exists** student;

**CREATE** **TABLE** student (

studentId BIGINT(20) **NOT** **NULL** AUTO\_INCREMENT,

firstName **VARCHAR**(50) **NOT** **NULL** ,

**PRIMARY** **KEY** (studentId)

);

**CREATE** **TABLE** subject (

subjectId BIGINT(10) **NOT** **NULL** AUTO\_INCREMENT,

name **VARCHAR**(50) **NULL** **DEFAULT** **NULL**,

studentId BIGINT(20) **NULL** **DEFAULT** **NULL**,

**PRIMARY** **KEY** (subjectId),

**FOREIGN** **KEY** (studentId) **REFERENCES** student(studentId)

);

**select** \* **from** student;

**select** \* **from** subject;

/\* Get all the subjects of where student id is 2\*/

**select** st.firstName , sb.name **from** student st, subject sb **where** st.studentId = 2 **and** st.studentId = sb.studentId;

/\*Get all student name and their subjects \*/

**select** st.firstName , sb.name **from** student st, subject sb **where** st.studentId = sb.studentId;

/\* Get all the subjects of sam\*/

**select** st.firstName , sb.name **from** student st, subject sb **where** st.firstName = 'Sam' **and** st.studentId = sb.studentId;

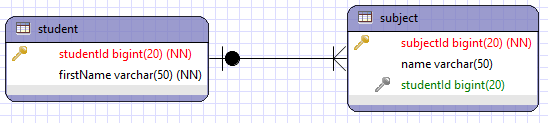


Table : **Student** Table : **Subject**

# Java Files

## **Student.java**

**package** com.ddlab.rnd.hibernate;

**import** java.io.Serializable;

**import** java.util.ArrayList;

**import** java.util.List;

**import** javax.persistence.CascadeType;

**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 = "STUDENT")

**public** **class** Student **implements** Serializable {

**private** **static** **final** **long** ***serialVersionUID*** = 8345128974931566292L;

@Id

@GeneratedValue

@Column(name = "studentId")

**private** **long** id;

@Column(name = "firstName")

**private** String firstName;

**/\***

**\* How will you get subjects from Student table**

**\* By using mappedBy as there is no direct relationship.**

**\* Student does not maintain relations, it is owned by Subject.**

**\* Subject class has a reference to student variable**

**\*/**

**@OneToMany(cascade = CascadeType.*ALL*, mappedBy = "student")**

**private List<Subject> subjects = new ArrayList<Subject>();**

**public** Student() {}

**public** Student(String firstName) {

**this**.firstName = firstName;

}

**public** **long** getId() {

**return** id;

}

**public** **void** setId(**long** id) {

**this**.id = id;

}

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** List<Subject> getSubjects() {

**return** subjects;

}

**public** **void** setSubjects(List<Subject> subjects) {

**this**.subjects = subjects;

}

}

## **Subject.java**

**package** com.ddlab.rnd.hibernate;

**import** java.io.Serializable;

**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;

@Entity

@Table(name = "SUBJECT")

**public** **class** Subject **implements** Serializable {

**private** **static** **final** **long** ***serialVersionUID*** = 3331340035611931999L;

@Id

@GeneratedValue

@Column(name = "subjectId")

**private** **long** id;

@Column(name = "name")

**private** String name;

**/\***

**\* How will you get student information from Subject table ?**

**\* By joining studentId**

**\*/**

**@ManyToOne( cascade = CascadeType.*ALL* )**

**@JoinColumn(name = "studentId")**

**private Student student;**

**public** Student getStudent() {

**return** student;

}

**public** **void** setStudent(Student student) {

**this**.student = student;

}

**public** Subject() {}

**public** Subject(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;

}

}

## **Test2.java**

**package** com.ddlab.rnd.hibernate;

**import** org.hibernate.HibernateException;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** Test2 {

**public** **static** **void** saveStudent(SessionFactory sessionFactory, Student student) {

Session session = sessionFactory.openSession();

Transaction transaction = **null**;

**try** {

transaction = session.beginTransaction();

session.save(student);

transaction.commit();

} **catch** (HibernateException e) {

**if** (transaction != **null**)

transaction.rollback();

e.printStackTrace();

} **finally** {

session.close();

}

}

**public** **static** **void** saveSubject(SessionFactory sessionFactory, Subject subject) {

Session session = sessionFactory.openSession();

Transaction transaction = **null**;

**try** {

transaction = session.beginTransaction();

session.save(subject);

transaction.commit();

} **catch** (HibernateException e) {

**if** (transaction != **null**)

transaction.rollback();

e.printStackTrace();

} **finally** {

session.close();

}

}

**public** **static** **void** main(String[] args) {

ApplicationContext context = **new** ClassPathXmlApplicationContext("app-context.xml");

SessionFactory sessionFactory = (SessionFactory) context.getBean("hibSessionFactory");

Student student1 = **new** Student("Sam");

Subject subject1 = **new** Subject("Economics");

subject1.setStudent(student1);

Subject subject2 = **new** Subject("Politics");

subject2.setStudent(student1);

Subject subject3 = **new** Subject("Foreign Affairs");

subject3.setStudent(student1);

// Student1 have 3 subjects

student1.getSubjects().add(subject1);

student1.getSubjects().add(subject2);

student1.getSubjects().add(subject3);

// Save Student along with Subject

*saveStudent*(sessionFactory, student1);

Student student2 = **new** Student("Deb");

subject1.setStudent(student2);

student2.getSubjects().add(subject2);

// Save subject along with Student

*saveSubject*(sessionFactory, subject1);

System.***out***.println("successfully saved into database");

}

}

# Key to remember in case one-to-many Bidirectional

**@Entity**

**@Table(name = "STUDENT")**

**public class Student implements Serializable {**

**/\***

**\* How will you get subjects from Student table**

**\* By using mappedBy as there is no direct relationship.**

**\* Student does not maintain relations, it is owned by Subject.**

**\* Subject class has a reference to student variable**

**\*/**

**@OneToMany(cascade = CascadeType.*ALL*, mappedBy = "student")**

**private List<Subject> subjects = new ArrayList<Subject>();**

**public Student() {}**

**}**

**@Entity**

**@Table(name = "SUBJECT")**

**public class Subject implements Serializable {**

**/\***

**\* How will you get student information from Subject table ?**

**\* By joining studentId**

**\*/**

**@ManyToOne( cascade = CascadeType.*ALL* )**

**@JoinColumn(name = "studentId")**

**private Student student;**

**}**

Many To One Unidirectional Mapping (Many students have same University)

Same as pom.xml, app-context.xml

# SQL

**drop** **table** if **exists** STUDENT1;

**drop** **table** if **exists** UNIVERSITY1;

**create** **table** UNIVERSITY1 (

university\_id BIGINT **NOT** **NULL** AUTO\_INCREMENT,

name **VARCHAR**(30) **NOT** **NULL**,

**PRIMARY** **KEY** (university\_id)

);

**create** **table** STUDENT1 (

student\_id BIGINT **NOT** **NULL** AUTO\_INCREMENT,

university\_id BIGINT **NOT** **NULL**,

first\_name **VARCHAR**(30) **NOT** **NULL**,

**PRIMARY** **KEY** (student\_id),

**CONSTRAINT** student\_university **FOREIGN** **KEY** (university\_id) **REFERENCES** UNIVERSITY1 (university\_id) **ON** **UPDATE** **CASCADE** **ON** **DELETE** **CASCADE**

);

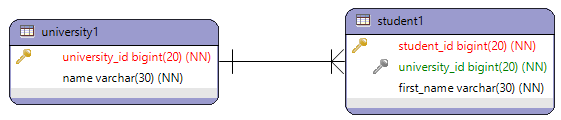


Table : University1 Table : Student1

# Java Files

## **Student.java**

**package** com.ddlab.rnd.hibernate;

**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.ManyToOne;

**import** javax.persistence.Table;

**@Entity**

**@Table(name = "STUDENT1")**

**public class Student** {

**@Id**

**@GeneratedValue(strategy=GenerationType.*IDENTITY*)**

**@Column(name = "STUDENT\_ID")**

**private long id;**

**@Column(name = "FIRST\_NAME")**

**private String firstName;**

**@ManyToOne(optional = false) // you may or may not provide (optional = false)**

**@JoinColumn(name="UNIVERSITY\_ID")**

**private University university;**

**public** Student() {

}

**public** Student(String firstName) {

**this**.firstName = firstName;

}

**public** **long** getId() {

**return** id;

}

**public** **void** setId(**long** id) {

**this**.id = id;

}

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** University getUniversity() {

**return** university;

}

**public** **void** setUniversity(University university) {

**this**.university = university;

}

}

## **University.java**

**package** com.ddlab.rnd.hibernate;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

**@Entity**

**@Table(name = "UNIVERSITY1")**

**public class University** {

**@Id**

**@GeneratedValue(strategy=GenerationType.*IDENTITY*)**

**@Column(name = "UNIVERSITY\_ID")**

**private long id;**

**@Column(name = "NAME")**

**private String name;**

**public** University() {

}

**public** University(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;

}

}

## **App.java**

**package** com.ddlab.rnd.hibernate;

**import** java.util.List;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** App {

**public** **static** **void** main(String[] args) {

ApplicationContext context = **new** ClassPathXmlApplicationContext("app-context.xml");

SessionFactory sessionFactory = (SessionFactory) context.getBean("hibSessionFactory");

Session session = sessionFactory.openSession();

Student student1 = **new** Student("Sam");

Student student2 = **new** Student("Joshua");

Student student3 = **new** Student("Peter");

University university = **new** University("CAMBRIDGE");

student1.setUniversity(university);

student2.setUniversity(university);

student3.setUniversity(university);

Transaction tx = **null**;

**try** {

tx = session.beginTransaction();

session.save(university);

session.save(student1);

session.save(student2);

session.save(student3);

tx.commit();

List<Student> students = (List<Student>)session.createQuery("from Student ").list();

**for**(Student s: students){

System.***out***.println("Details : "+s);

System.***out***.println("Student University Details: "+s.getUniversity());

}

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

**if** (!sessionFactory.isClosed()) {

System.***out***.println("Closing SessionFactory");

sessionFactory.close();

}

}

}

}

# Key to remember in case many-to-one Unidirectional

**@Entity**

**@Table(name = "STUDENT1")**

**public class Student {**

**@ManyToOne(optional = false) // you may or may not provide (optional = false)**

**@JoinColumn(name="UNIVERSITY\_ID")**

**private University university;**

**}**

Many To One Bidirectional Mapping (Many students have same University)

# SQL

**drop** **table** if **exists** STUDENT1;

**drop** **table** if **exists** UNIVERSITY1;

**create** **table** UNIVERSITY1 (

university\_id BIGINT **NOT** **NULL** AUTO\_INCREMENT,

name **VARCHAR**(30) **NOT** **NULL**,

**PRIMARY** **KEY** (university\_id)

);

**create** **table** STUDENT1 (

student\_id BIGINT **NOT** **NULL** AUTO\_INCREMENT,

university\_id BIGINT **NOT** **NULL**,

first\_name **VARCHAR**(30) **NOT** **NULL**,

**PRIMARY** **KEY** (student\_id),

**CONSTRAINT** student\_university **FOREIGN** **KEY** (university\_id) **REFERENCES** UNIVERSITY1 (university\_id) **ON** **UPDATE** **CASCADE** **ON** **DELETE** **CASCADE**

);

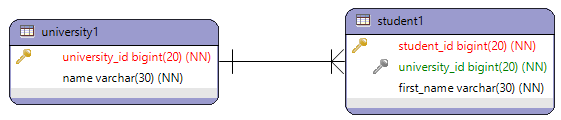


Table : University1 Table : Student1

# Java Files

## **Student.java**

**package** com.ddlab.rnd.hibernate;

**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.ManyToOne;

**import** javax.persistence.Table;

@Entity

@Table(name = "STUDENT1")

**public** **class** Student {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

@Column(name = "STUDENT\_ID")

**private** **long** id;

@Column(name = "FIRST\_NAME")

**private** String firstName;

**@ManyToOne(cascade = CascadeType.*ALL*) //Not required to mention optional = false**

**@JoinColumn(name = "UNIVERSITY\_ID")**

**private University university;**

**public** Student() {}

**public** Student(String firstName) {

**this**.firstName = firstName;

}

**public** **long** getId() {

**return** id;

}

**public** **void** setId(**long** id) {

**this**.id = id;

}

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** University getUniversity() {

**return** university;

}

**public** **void** setUniversity(University university) {

**this**.university = university;

}

}

## **University.java**

**package** com.ddlab.rnd.hibernate;

**import** java.util.List;

**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.OneToMany;

**import** javax.persistence.Table;

@Entity

@Table(name = "UNIVERSITY1")

**public** **class** University {

@Id

@GeneratedValue(strategy=GenerationType.***IDENTITY***)

@Column(name = "UNIVERSITY\_ID")

**private** **long** id;

@Column(name = "NAME")

**private** String name;

**@OneToMany(cascade = CascadeType.*ALL*, mappedBy = "university")**

**private List<Student> students;**

**public** University() { }

**public** University(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** List<Student> getStudents() {

**return** students;

}

**public** **void** setStudents(List<Student> students) {

**this**.students = students;

}

}

## **App.java**

**package** com.ddlab.rnd.hibernate;

**import** java.util.ArrayList;

**import** java.util.List;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** App {

**public** **static** **void** saveUniversity(SessionFactory sessionFactory) {

Session session = sessionFactory.openSession();

Student student1 = **new** Student("Sam");

Student student2 = **new** Student("Joshua");

Student student3 = **new** Student("Peter");

University university = **new** University("CAMBRIDGE");

List<Student> allStudents = **new** ArrayList<Student>();

student1.setUniversity(university);

student2.setUniversity(university);

student3.setUniversity(university);

allStudents.add(student1);

allStudents.add(student2);

allStudents.add(student3);

university.setStudents(allStudents);

Transaction tx = **null**;

**try** {

tx = session.beginTransaction();

session.save(university);

tx.commit();

List<Student> students = (List<Student>) session.createQuery("from Student ").list();

**for** (Student s : students) {

System.***out***.println("Student Details : " + s);

System.***out***.println("Student University Details: " + s.getUniversity());

}

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

session.close();

}

}

**public** **static** **void** saveStudent(SessionFactory sessionFactory) {

Session session = sessionFactory.openSession();

University university = **new** University("Standford");

Student student1 = **new** Student("Deb");

student1.setUniversity(university);

List<Student> allStudents = **new** ArrayList<Student>();

allStudents.add(student1);

university.setStudents(allStudents);

Transaction tx = **null**;

**try** {

tx = session.beginTransaction();

session.save(student1);

tx.commit();

List<University> universities = (List<University>) session.createQuery("from University ").list();

**for** (University u : universities) {

System.***out***.println(u.getName()+"---"+u.getStudents());

}

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

session.close();

}

}

**public** **static** **void** main(String[] args) {

ApplicationContext context = **new** ClassPathXmlApplicationContext("app-context.xml");

SessionFactory sessionFactory = (SessionFactory) context.getBean("hibSessionFactory");

*saveUniversity*(sessionFactory);

*saveStudent*(sessionFactory);

System.***out***.println("All information saved successfully ...");

}

}

# Key to remember in case many-to-one Bidirectional

**@Entity**

**@Table(name = "STUDENT1")**

**public class Student {**

**@ManyToOne(cascade = CascadeType.*ALL*) //You may or may not provide (optional = false)**

**@JoinColumn(name = "UNIVERSITY\_ID")**

**private University university;**

**}**

**@Entity**

**@Table(name = "UNIVERSITY1")**

**public class University {**

**@OneToMany(cascade = CascadeType.*ALL*, mappedBy = "university")**

**private List<Student> students;**

**}**

Many To Many Unidirectional Mapping (Many students have many Subjects)

# SQL

**drop** **table** IF **EXISTS** STUDENT\_SUBJECT3;

**drop** **table** IF **EXISTS** STUDENT3;

**drop** **table** IF **EXISTS** SUBJECT3;

**create** **table** STUDENT3 (

student\_id BIGINT **NOT** **NULL** AUTO\_INCREMENT,

first\_name **VARCHAR**(30) **NOT** **NULL**,

**PRIMARY** **KEY** (student\_id)

);

**create** **table** SUBJECT3 (

subject\_id BIGINT **NOT** **NULL** AUTO\_INCREMENT,

name **VARCHAR**(30) **NOT** **NULL**,

**PRIMARY** **KEY** (subject\_id)

);

**CREATE** **TABLE** STUDENT\_SUBJECT3 (

student\_id BIGINT **NOT** **NULL**,

subject\_id BIGINT **NOT** **NULL**,

**PRIMARY** **KEY** (student\_id, subject\_id),

**CONSTRAINT** FK\_STUDENT3 **FOREIGN** **KEY** (student\_id) **REFERENCES** STUDENT3 (student\_id),

**CONSTRAINT** FK\_SUBJECT3 **FOREIGN** **KEY** (subject\_id) **REFERENCES** SUBJECT3 (subject\_id)

);

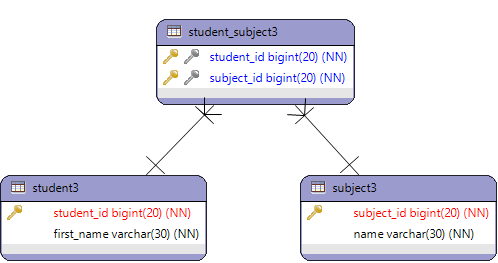


Table : Student\_Subject3 Table : Student3 Table : Subject3

# Java Files

## **Student.java**

**package** com.ddlab.rnd.hibernate;

**import** java.util.ArrayList;

**import** java.util.List;

**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 = "STUDENT3")**

**public class Student** {

**@Id**

**@GeneratedValue(strategy=GenerationType.*IDENTITY*)**

**@Column(name = "STUDENT\_ID")**

**private long id;**

**@Column(name = "FIRST\_NAME")**

**private String firstName;**

**@ManyToMany(cascade = CascadeType.*ALL*)**

**@JoinTable(name = "STUDENT\_SUBJECT3",**

**joinColumns = { @JoinColumn(name = "STUDENT\_ID") },**

**inverseJoinColumns = { @JoinColumn(name = "SUBJECT\_ID") })**

**private List<Subject> subjects = new ArrayList<Subject>();**

**public** Student() {

}

**public** Student(String firstName) {

**this**.firstName = firstName;

}

**public** **long** getId() {

**return** id;

}

**public** **void** setId(**long** id) {

**this**.id = id;

}

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** List<Subject> getSubjects() {

**return** subjects;

}

**public** **void** setSubjects(List<Subject> subjects) {

**this**.subjects = subjects;

}

}

## **Subject.java**

**package** com.ddlab.rnd.hibernate;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

**@Entity**

**@Table(name = "SUBJECT3")**

**public class Subject** {

**@Id**

**@GeneratedValue(strategy=GenerationType.*IDENTITY*)**

**@Column(name = "SUBJECT\_ID")**

**private long id;**

**@Column(name = "NAME")**

**private String name;**

**public** Subject(){

}

**public** Subject(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;

}

}

## **App.java**

**package** com.ddlab.rnd.hibernate;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** App {

**public** **static** **void** main(String[] args) {

ApplicationContext context = **new** ClassPathXmlApplicationContext("app-context.xml");

SessionFactory sessionFactory = (SessionFactory) context.getBean("hibSessionFactory");

Session session = sessionFactory.openSession();

Student student1 = **new** Student("Sam");

Student student2 = **new** Student("Joshua");

Subject subject1 = **new** Subject("Economics");

Subject subject2 = **new** Subject("Politics");

Subject subject3 = **new** Subject("Foreign Affairs");

//Student1 have 3 subjects

student1.getSubjects().add(subject1);

student1.getSubjects().add(subject2);

student1.getSubjects().add(subject3);

//Student2 have 2 subjects

student2.getSubjects().add(subject1);

student2.getSubjects().add(subject2);

Transaction tx = **null**;

**try** {

tx = session.beginTransaction();

session.persist(student1);

session.persist(student2);

tx.commit();

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

**if** (!sessionFactory.isClosed()) {

System.***out***.println("Closing SessionFactory");

sessionFactory.close();

}

}

}

}

# Key to remember in case many-to-many Unidirectional

**@Entity**

**@Table(name = "STUDENT3")**

**public class Student {**

**@ManyToMany(cascade = CascadeType.*ALL*)**

**@JoinTable(name = "STUDENT\_SUBJECT3",**

**joinColumns = { @JoinColumn(name = "STUDENT\_ID") },**

**inverseJoinColumns = { @JoinColumn(name = "SUBJECT\_ID") })**

**private List<Subject> subjects = new ArrayList<Subject>();**

**}**

Many To Many Bidirectional Mapping (Many students have many Subjects)

# SQL

**drop** **table** IF **EXISTS** STUDENT\_SUBJECT3;

**drop** **table** IF **EXISTS** STUDENT3;

**drop** **table** IF **EXISTS** SUBJECT3;

**create** **table** STUDENT3 (

student\_id BIGINT **NOT** **NULL** AUTO\_INCREMENT,

first\_name **VARCHAR**(30) **NOT** **NULL**,

**PRIMARY** **KEY** (student\_id)

);

**create** **table** SUBJECT3 (

subject\_id BIGINT **NOT** **NULL** AUTO\_INCREMENT,

name **VARCHAR**(30) **NOT** **NULL**,

**PRIMARY** **KEY** (subject\_id)

);

**CREATE** **TABLE** STUDENT\_SUBJECT3 (

student\_id BIGINT **NOT** **NULL**,

subject\_id BIGINT **NOT** **NULL**,

**PRIMARY** **KEY** (student\_id, subject\_id),

**CONSTRAINT** FK\_STUDENT3 **FOREIGN** **KEY** (student\_id) **REFERENCES** STUDENT3 (student\_id),

**CONSTRAINT** FK\_SUBJECT3 **FOREIGN** **KEY** (subject\_id) **REFERENCES** SUBJECT3 (subject\_id)

);

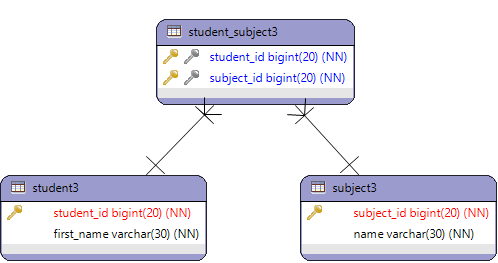


Table : Student\_Subject3 Table : Student3 Table : Subject3

# Java Files

## **Student.java**

**package** com.ddlab.rnd.hibernate;

**import** java.util.ArrayList;

**import** java.util.List;

**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 = "STUDENT3")

**public** **class** Student {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

@Column(name = "studentId")

**private** **long** id;

@Column(name = "firstName")

**private** String firstName;

**/\***

**\* How will you get subject information from Student table ? This is a**

**\* special case for many to many. We get by joining third table**

**\* student-subject3**

**\*/**

**@ManyToMany(cascade = CascadeType.*ALL*)**

**@JoinTable(name = "STUDENT\_SUBJECT3", joinColumns = { @JoinColumn(name = "studentId") }, inverseJoinColumns = { @JoinColumn(name = "subjectId") })**

**private List<Subject> subjects = new ArrayList<Subject>();**

**public** Student() {}

**public** Student(String firstName) {

**this**.firstName = firstName;

}

**public** **long** getId() {

**return** id;

}

**public** **void** setId(**long** id) {

**this**.id = id;

}

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** List<Subject> getSubjects() {

**return** subjects;

}

**public** **void** setSubjects(List<Subject> subjects) {

**this**.subjects = subjects;

}

}

## **Subject.java**

**package** com.ddlab.rnd.hibernate;

**import** java.util.ArrayList;

**import** java.util.List;

**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.ManyToMany;

**import** javax.persistence.Table;

@Entity

@Table(name = "SUBJECT3")

**public** **class** Subject {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

@Column(name = "subjectId")

**private** **long** id;

@Column(name = "name")

**private** String name;

@ManyToMany(cascade = CascadeType.***ALL***, mappedBy = "subjects")

**private** List<Student> students = **new** ArrayList<Student>();

**public** Subject() {}

**public** Subject(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** List<Student> getStudents() {

**return** students;

}

**public** **void** setStudents(List<Student> students) {

**this**.students = students;

}

}

## **App.java**

**package** com.ddlab.rnd.hibernate;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** App {

**public** **static** **void** saveStudent(SessionFactory sessionFactory) {

Session session = sessionFactory.openSession();

Student student1 = **new** Student("Sam");

Student student2 = **new** Student("Joshua");

Subject subject1 = **new** Subject("Economics");

Subject subject2 = **new** Subject("Politics");

Subject subject3 = **new** Subject("Foreign Affairs");

// Student1 have 3 subjects

student1.getSubjects().add(subject1);

student1.getSubjects().add(subject2);

student1.getSubjects().add(subject3);

// Student2 have 2 subjects

student2.getSubjects().add(subject1);

student2.getSubjects().add(subject2);

Transaction tx = **null**;

**try** {

tx = session.beginTransaction();

session.save(student1);

session.save(student2);

tx.commit();

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

session.close();

}

}

**public** **static** **void** saveSubject(SessionFactory sessionFactory) {

Session session = sessionFactory.openSession();

Student student1 = **new** Student("Vidya Balan");

Student student2 = **new** Student("John Abraham");

Subject subject1 = **new** Subject("Physics");

subject1.getStudents().add(student1);

subject1.getStudents().add(student2);

Subject subject2 = **new** Subject("Chemistry");

subject2.getStudents().add(student1);

subject2.getStudents().add(student2);

Subject subject3 = **new** Subject("Mathematics");

subject3.getStudents().add(student1);

// subject3.getStudents().add(student2);

// Student1 have 3 subjects

student1.getSubjects().add(subject1);

student1.getSubjects().add(subject2);

student1.getSubjects().add(subject3);

// Student2 have 2 subjects

student2.getSubjects().add(subject1);

student2.getSubjects().add(subject2);

Transaction tx = **null**;

**try** {

tx = session.beginTransaction();

session.save(subject1);

session.save(subject2);

session.save(subject3);

tx.commit();

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

session.close();

}

}

**public** **static** **void** main(String[] args) {

ApplicationContext context = **new** ClassPathXmlApplicationContext("app-context.xml");

SessionFactory sessionFactory = (SessionFactory) context.getBean("hibSessionFactory");

// Session session = sessionFactory.openSession();

*saveStudent*(sessionFactory);

*saveSubject*(sessionFactory);

System.***out***.println("All information saved successfully ...");

}

}

# Key to remember in case many-to-many Bidirectional

**@Entity**

**@Table(name = "STUDENT3")**

**public class Student {**

**/\***

**\* How will you get subject information from Student table ? This is a**

**\* special case for many to many. We get by joining third table**

**\* student-subject3**

**\*/**

**@ManyToMany(cascade = CascadeType.*ALL*)**

**@JoinTable(name = "STUDENT\_SUBJECT3",**

**joinColumns = { @JoinColumn(name = "STUDENT\_ID") },**

**inverseJoinColumns = { @JoinColumn(name = "SUBJECT\_ID") })**

**private List<Subject> subjects = new ArrayList<Subject>();**

**}**

**@Entity**

**@Table(name = "SUBJECT3")**

**public class Subject {**

**@ManyToMany(cascade = CascadeType.*ALL*, mappedBy = "subjects")**

**private List<Student> students = new ArrayList<Student>();**

**}**

Inheritance Mapping in Hibernate

There are three inheritance mapping strategies defined in the hibernate:

1. Table Per Hierarchy
2. Table Per Concrete class
3. Table Per Subclass

# Table Per Hierarchy

In table per hierarchy mapping, single table is required to map the whole hierarchy, an extra column (known as discriminator column) is added to identify the class. But nullable values are stored in the table.

# Table Per Concrete class

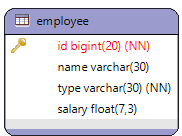
In case of table per concrete class, tables are created as per class. But duplicate column is added in subclass tables. Disadvantage of this approach is that duplicate columns are created in the subclass tables.

# Table Per Subclass

In this strategy, tables are created as per class but related by foreign key. So there are no duplicate columns.

# Table Per Class Hierarchy

# SQL



**drop** **table** IF **EXISTS** Employee;

**create** **table** Employee (

id BIGINT **NOT** **NULL** AUTO\_INCREMENT,

name **VARCHAR**(30) ,

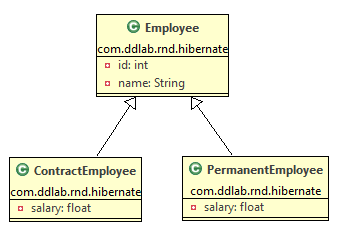
**type** **VARCHAR**(30) **NOT** **NULL**,

salary **FLOAT**(7,3),

**PRIMARY** **KEY** (id)

);

# Class Diagrams



# Maven Configuration (pom.xml)

<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>table-per-class-hierarchy</groupId>

<artifactId>table-per-class-hierarchy</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>table-per-class-hierarchy</name>

<url>http://maven.apache.org</url>

<properties>

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

<spring.version>4.1.2.RELEASE</spring.version>

<spring.security.version>3.2.3.RELEASE</spring.security.version>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

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

<version>5.1.30</version>

</dependency>

<!-- C3P0 library -->

<dependency>

<groupId>com.mchange</groupId>

<artifactId>c3p0</artifactId>

<version>0.9.5</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>4.3.7.Final</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

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

<version>3.3</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

# Java Files

## **Employee.java**

**package** com.ddlab.rnd.hibernate;

**import** javax.persistence.Column;

**import** javax.persistence.DiscriminatorColumn;

**import** javax.persistence.DiscriminatorType;

**import** javax.persistence.DiscriminatorValue;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.Inheritance;

**import** javax.persistence.InheritanceType;

**import** javax.persistence.Table;

**@Entity**

**@Table(name = "employee")**

**@Inheritance(strategy=InheritanceType.*SINGLE\_TABLE*)**

**@DiscriminatorColumn(name="type",discriminatorType=DiscriminatorType.*STRING*)**

**@DiscriminatorValue(value="Employee")**

**public** **class** Employee {

@Id

@GeneratedValue(strategy=GenerationType.***IDENTITY***)

@Column(name = "id")

**private** **int** id;

@Column(name = "name")

**private** String name;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

}

## **PermanentEmployee.java**

**package** com.ddlab.rnd.hibernate;

**import** javax.persistence.Column;

**import** javax.persistence.DiscriminatorValue;

**import** javax.persistence.Entity;

**@Entity**

**@DiscriminatorValue("PermanentEmployee")**

**public** **class** PermanentEmployee **extends** Employee {

@Column(name="salary")

**private** **float** salary;

**public** **float** getSalary() {

**return** salary;

}

**public** **void** setSalary(**float** salary) {

**this**.salary = salary;

}

}

## **ContractEmployee.java**

**package** com.ddlab.rnd.hibernate;

**import** javax.persistence.Column;

**import** javax.persistence.DiscriminatorValue;

**import** javax.persistence.Entity;

**@Entity**

**@DiscriminatorValue("ContractEmployee")**

**public** **class** ContractEmployee **extends** Employee {

@Column(name="salary")

**private** **float** salary;

**public** **float** getSalary() {

**return** salary;

}

**public** **void** setSalary(**float** salary) {

**this**.salary = salary;

}

}

## **App.java**

**package** com.ddlab.rnd.hibernate;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** App {

**public** **static** **void** main(String[] args) {

ApplicationContext context = **new** ClassPathXmlApplicationContext("app-context.xml");

SessionFactory sessionFactory = (SessionFactory) context.getBean("hibSessionFactory");

Session session = sessionFactory.openSession();

Employee e1 = **new** Employee();

e1.setName("Deb");

PermanentEmployee pe = **new** PermanentEmployee();

pe.setName("John Abraham");

pe.setSalary(3000.50F);

ContractEmployee ce = **new** ContractEmployee();

ce.setName("Vidya Balan");

ce.setSalary(1000.50F);

Transaction tx = **null**;

**try** {

tx = session.beginTransaction();

session.save(e1);

session.save(pe);

session.save(ce);

tx.commit();

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

**if** (!sessionFactory.isClosed()) {

System.***out***.println("Closing SessionFactory");

sessionFactory.close();

}

}

}

}

# Key to Remember for Table Per Class Hierarchy

**@Inheritance(strategy=InheritanceType.*SINGLE\_TABLE*)**

**@DiscriminatorColumn(name="type",discriminatorType=DiscriminatorType.*STRING*)**

**@DiscriminatorValue(value="Employee")**

**public** **class** Employee {

}

**@Entity**

**@DiscriminatorValue("PermanentEmployee")**

**public** **class** PermanentEmployee **extends** Employee {

}

**@Entity**

**@DiscriminatorValue("ContractEmployee")**

**public** **class** ContractEmployee **extends** Employee {

}

# Table Per Concrete Class

# SQL

**drop** **table** IF **EXISTS** Employee;

**drop** **table** IF **EXISTS** ContractEmployee;

**drop** **table** IF **EXISTS** PermanentEmployee;

**create** **table** Employee (

id BIGINT **NOT** **NULL** AUTO\_INCREMENT,

name **VARCHAR**(30) ,

**PRIMARY** **KEY** (id)

);

**create** **table** ContractEmployee (

id BIGINT **NOT** **NULL** ,

name **VARCHAR**(30) ,

salary **FLOAT**(7,3),

**PRIMARY** **KEY** (id)

);

**create** **table** PermanentEmployee (

id BIGINT **NOT** **NULL** ,

name **VARCHAR**(30) ,

projectname **VARCHAR**(30) ,

**PRIMARY** **KEY** (id)

);

Pom.xml, class diagram same

# Java Files

## **Employee.java**

**package** com.ddlab.rnd.hibernate;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.Id;

**import** javax.persistence.Inheritance;

**import** javax.persistence.InheritanceType;

**import** javax.persistence.Table;

@**Entity**

**@Table(name = "employee")**

**@Inheritance(strategy=InheritanceType.*TABLE\_PER\_CLASS*)**

**public** **class** Employee {

@Id

@Column(name = "id")

**private** **int** id;

@Column(name = "name")

**private** String name;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

}

## **PermanetEmployee.java**

**package** com.ddlab.rnd.hibernate;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.Table;

**@Entity**

**@Table(name="PermanentEmployee")**

**public** **class** PermanentEmployee **extends** Employee {

@Column(name="projectName")

**private** String projectName;

**public** String getProjectName() {

**return** projectName;

}

**public** **void** setProjectName(String projectName) {

**this**.projectName = projectName;

}

}

## **ContractEmployee.java**

**package** com.ddlab.rnd.hibernate;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.Table;

**@Entity**

**@Table(name="ContractEmployee")**

**public** **class** ContractEmployee **extends** Employee {

@Column(name="salary")

**private** **float** salary;

**public** **float** getSalary() {

**return** salary;

}

**public** **void** setSalary(**float** salary) {

**this**.salary = salary;

}

}

## **App.java**

**package** com.ddlab.rnd.hibernate;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** App {

**public** **static** **void** main(String[] args) {

ApplicationContext context = **new** ClassPathXmlApplicationContext("app-context.xml");

SessionFactory sessionFactory = (SessionFactory) context.getBean("hibSessionFactory");

Session session = sessionFactory.openSession();

Employee e1 = **new** Employee();

e1.setName("Deb");

e1.setId(1);

PermanentEmployee pe = **new** PermanentEmployee();

pe.setName("John Abraham");

pe.setProjectName("Apollo");

pe.setId(2);

ContractEmployee ce = **new** ContractEmployee();

ce.setName("Vidya Balan");

ce.setSalary(1000.50F);

ce.setId(3);

Transaction tx = **null**;

**try** {

tx = session.beginTransaction();

session.save(e1);

session.save(pe);

session.save(ce);

tx.commit();

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

**if** (!sessionFactory.isClosed()) {

System.***out***.println("Closing SessionFactory");

sessionFactory.close();

}

}

}

}

# Key to Remember for Table Per Concrete Class

@**Entity**

**@Table(name = "employee")**

**@Inheritance(strategy=InheritanceType.*TABLE\_PER\_CLASS*)**

**public** **class** Employee {

}

# Table Per Sub Class

# SQL

**drop** **table** IF **EXISTS** PermanentEmployee;

**drop** **table** IF **EXISTS** ContractEmployee;

**drop** **table** IF **EXISTS** Employee;

**create** **table** Employee (

id BIGINT **NOT** **NULL** AUTO\_INCREMENT,

name **VARCHAR**(30) ,

**PRIMARY** **KEY** (id)

);

**create** **table** ContractEmployee (

id BIGINT **NOT** **NULL** ,

salary **FLOAT**(7,3),

**PRIMARY** **KEY** (id),

**CONSTRAINT** FK\_ContractEmployee **FOREIGN** **KEY** (id) **REFERENCES** Employee (id)

);

**create** **table** PermanentEmployee (

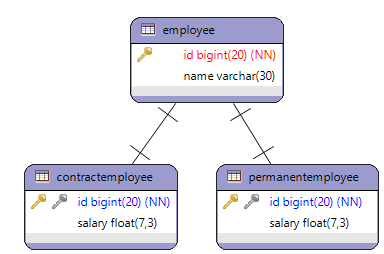
id BIGINT **NOT** **NULL** ,

salary **FLOAT**(7,3),

**PRIMARY** **KEY** (id),

**CONSTRAINT** FK\_PermanentEmployee **FOREIGN** **KEY** (id) **REFERENCES** Employee (id)

);



Pom.xml, class diagrams same

# Java Files

## **Employee.java**

**package** com.ddlab.rnd.hibernate;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.Inheritance;

**import** javax.persistence.InheritanceType;

**import** javax.persistence.Table;

**@Entity**

**@Table(name = "employee")**

**@Inheritance(strategy=InheritanceType.*JOINED*)**

**public** **class** Employee {

@Id

@GeneratedValue(strategy=GenerationType.***IDENTITY***)

@Column(name = "id")

**private** **int** id;

@Column(name = "name")

**private** String name;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

}

## **PermanentEmployee.java**

**package** com.ddlab.rnd.hibernate;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.PrimaryKeyJoinColumn;

**import** javax.persistence.Table;

@Entity

@Table(name="PermanentEmployee")

@PrimaryKeyJoinColumn(name="ID")

**public** **class** PermanentEmployee **extends** Employee {

@Column(name="salary")

**private** **float** salary;

**public** **float** getSalary() {

**return** salary;

}

**public** **void** setSalary(**float** salary) {

**this**.salary = salary;

}

}

## **ContractEmployee.java**

**package** com.ddlab.rnd.hibernate;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.PrimaryKeyJoinColumn;

**import** javax.persistence.Table;

@Entity

@Table(name="ContractEmployee")

@PrimaryKeyJoinColumn(name="ID")

**public** **class** ContractEmployee **extends** Employee {

@Column(name="salary")

**private** **float** salary;

**public** **float** getSalary() {

**return** salary;

}

**public** **void** setSalary(**float** salary) {

**this**.salary = salary;

}

}

## **App.Java**

**package** com.ddlab.rnd.hibernate;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** App {

**public** **static** **void** main(String[] args) {

ApplicationContext context = **new** ClassPathXmlApplicationContext("app-context.xml");

SessionFactory sessionFactory = (SessionFactory) context.getBean("hibSessionFactory");

Session session = sessionFactory.openSession();

Employee e1 = **new** Employee();

e1.setName("Deb");

PermanentEmployee pe = **new** PermanentEmployee();

pe.setName("John Abraham");

pe.setSalary(3000.50F);

ContractEmployee ce = **new** ContractEmployee();

ce.setName("Vidya Balan");

ce.setSalary(1000.50F);

Transaction tx = **null**;

**try** {

tx = session.beginTransaction();

session.save(e1);

session.save(pe);

session.save(ce);

tx.commit();

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

**if** (!sessionFactory.isClosed()) sessionFactory.close();

}

}

}

# Key to Remember for Table Per Sub Class

**@Entity**

**@Table(name = "employee")**

**@Inheritance(strategy=InheritanceType.*JOINED*)**

**public class Employee {**

**}**

**@Entity**

**@Table(name="PermanentEmployee")**

**@PrimaryKeyJoinColumn(name="ID")**

**public class PermanentEmployee extends Employee {**

**}**

**@Entity**

**@Table(name="ContractEmployee")**

**@PrimaryKeyJoinColumn(name="ID")**

**public class ContractEmployee extends Employee {**

**}**

save, persist, saveOrUpdate, update, merge

# save

* Used to save entity in database
* Issue: If we use this without transaction & we have cascading between entities, then only primary key gets save unless we flush the session.
* We should not use save outside transaction boundaries.
* save() method returns generated id immediately, this is possible because primary object is saved as soon as save method is invoked
* If there are other objects mapped to the primary object, they gets saved at the time of committing transaction.
* For the objects in persistent state, save updates the data through update query. This happens when tx.commit() is done.
* save() load entity object to persistent context.

# persist

* Similar to save() & adds entity object to persistent context.
* Persist itself takes care of cascaded objects
* Persist doesn't return anything, persist object is used to get generated identifier.

# saveOrUpdate

* This results into insert or update queries based on data. If the data is present in the database, update will be executed.
* We can use saveOrUpdate without transaction, but issue will mapped objects.
* adds entity objects to persistent cotext & track any further changes. Any further changes can be persisted using persist.

# update

* Should be used only when you know that you are updating the entity information.
* Update doesn't return anything.

# Merge

* Used to update existing values.
* This method creates a copy from the passed entity object & return it.
* This means the returned object is different from passed object.
* Return object can be tracked for further changes but passed object cannot.

session.load() Vs session.get()

# session.load()

* It will always return a “**proxy**” (Hibernate term) without hitting the database. In Hibernate, proxy is an object with the given identifier value, its properties are not initialized yet, it just look like a temporary fake object.
* If no row found , it will throws an **ObjectNotFoundException**.

# session.get()

* It always hit the database and return the real object, an object that represent the database row, not proxy.
* If no row found , it return null.

Hibernate States

A new instance of a a persistent class which is not associated with a Session, has no representation in the database and no identifier value is considered ***transient*** by Hibernate:

Person person = new Person();

person.setName("Foobar");

// person is in a transient state

A ***persistent*** instance has a representation in the database, an identifier value and is associated with a Session. You can make a transient instance ***persistent*** by associating it with a Session:

Long id = (Long) session.save(person); // person is now in a persistent state

Now, if we close the Hibernate Session, the persistent instance will become a ***detached*** instance: it isn't attached to a Session anymore (but can still be modified and reattached to a new Sessionlater though).

<http://www.java4s.com/hibernate/life-cycle-of-pojo-class-objects-in-the-hibernate/>

**Transient & Persistent states:**

* When ever an object of a pojo class is created then it will be in the Transient state
* When the object is in a Transient state it doesn’t represent any row of the database, i mean not associated with any Session object, if we speak more we can say no relation with the database its just an normal object
* If we modify the data of a pojo class object, when it is in transient state then it doesn’t effect on the database table
* When the object is in persistent state, then it represent one row of the database, if the object is in persistent state then it is associated with the unique Session
* if we want to move an object from persistent to detached state, we need to do either closing that session or need to clear the cache of the session
* if we want to move an object from persistent state into transient state then we need to delete that object permanently from the database