**Spring JPA Hibernate Mapping-Better-Way-2022**

**One To One Unidirectional**

@Entity(name = "Item\_O2OU")

@Table(name = "item\_o2ou")

@Getter @Setter

**public** **class** Item {

@Id @GeneratedValue

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

@Column(name = "name")

**private** String name;

}

@Entity(name = "ShoppingCart\_O2OU")

@Table(name = "shopping\_cart\_o2ou")

@Getter @Setter

**public** **class** ShoppingCart {

@Id @GeneratedValue

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

@Column(name = "name")

**private** String name;

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

**private** Item item;

}

**Database Table Design**

Graphical user interface, text

Description automatically generated Graphical user interface, application

Description automatically generated

**One To One Bidirectional**

@Entity(name = "Item\_O2OB")

@Table(name = "item\_o2ob")

@Getter @Setter

**public** **class** Item {

@Id @GeneratedValue

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

@Column(name = "name")

**private** String name;

**@OneToOne(mappedBy = "item")**

**private** ShoppingCart cart;

}

@Entity(name = "ShoppingCart\_O2OB")

@Table(name = "shopping\_cart\_o2ob")

@Getter @Setter

**public** **class** ShoppingCart {

@Id @GeneratedValue

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

@Column(name = "name")

**private** String name;

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

**private** Item item;

}

**Database Table Design**

Graphical user interface, text

Description automatically generated Graphical user interface

Description automatically generated

You can also mentioned mappedBy in ShoppingCart class also.

**One To Many Unidirectional**

@Entity(name = "Item\_O2MU")

@Table(name = "item\_o2mu")

@Getter @Setter

**public** **class** Item {

@Id @GeneratedValue

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

@Column(name = "name")

**private** String name;

}

@Entity(name = "ShoppingCart\_O2MU")

@Table(name = "shopping\_cart\_o2mu")

@Getter @Setter

**public** **class** ShoppingCart {

@Id @GeneratedValue

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

@Column(name = "name")

**private** String name;

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

**private** Set<Item> items =

**new** HashSet<Item>();

}

**Database Table Design**

Graphical user interface, application

Description automatically generated Text

Description automatically generated with low confidence Table

Description automatically generated

**One To Many Bidirectional**

@Entity(name = "ShoppingCart\_O2MB")

@Table(name = "shopping\_cart\_o2mb")

@Getter

@Setter

**public** **class** ShoppingCart {

@Id

@GeneratedValue

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

@Column(name = "name")

**private** String name;

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

**mappedBy = "shoppingCart")**

**private** Set<Item> items =

**new** HashSet<Item>();

}

@Entity(name = "Item\_O2MB")

@Table(name = "item\_o2mb")

@Getter @Setter

**public** **class** Item {

@Id @GeneratedValue

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

@Column(name = "name")

**private** String name;

**@ManyToOne**

**private** ShoppingCart shoppingCart;

}

**Database Table Design**

Graphical user interface, text

Description automatically generated Table

Description automatically generated

**Many 2 Many Unidirectional**

@Entity(name = "Item\_m2mu")

@Table(name = "item\_m2mu")

@Getter @Setter

**public** **class** Item {

@Id @GeneratedValue

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

@Column(name = "name")

**private** String name;

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

@JoinTable(name = "shopping\_cart\_item\_m2mu",

joinColumns =

@JoinColumn(name = "cart\_id"),

inverseJoinColumns =

@JoinColumn(name = "item\_id"))

**private** Set<ShoppingCart> shoppingCarts =

**new** HashSet<ShoppingCart>();

}

@Entity(name = "ShoppingCart\_m2mu")

@Table(name = "shoppingcart\_m2mu")

@Getter @Setter

**public** **class** ShoppingCart {

@Id @GeneratedValue

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

@Column(name="name")

**private** String name;

}

**Database Table Design**

Graphical user interface, text

Description automatically generated Text

Description automatically generated Table

Description automatically generated with medium confidence

**Many 2 Many Bidirectional**

@Entity(name = "Item\_m2mb")

@Table(name = "item\_m2mb")

@Getter @Setter

**public** **class** Item {

@Id @GeneratedValue

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

@Column(name = "name")

**private** String name;

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

**@JoinTable(name = "shopping\_cart\_item\_m2mb",**

**joinColumns = @JoinColumn(name = "cart\_id"),**

**inverseJoinColumns = @JoinColumn(name = "item\_id"))**

**private** Set<ShoppingCart> shoppingCarts =

**new** HashSet<ShoppingCart>();

}

**Database Table Design**

@Entity(name = "ShoppingCart\_m2mb")

@Table(name = "shoppingcart\_m2mb")

@Getter @Setter

**public** **class** ShoppingCart {

@Id @GeneratedValue

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

@Column(name="name")

**private** String name;

**@ManyToMany(mappedBy="shoppingCarts")**

**private** Set<Item> items =

**new** HashSet<Item>();

}

Graphical user interface, application

Description automatically generated Graphical user interface, text, application, table

Description automatically generated Table

Description automatically generated

**JPA - Bidirectional OneToMany/ManyToOne**

**A quick overview of bidirectional one-to-many/many-to-one relationship in JPA**

* In bidirectional one-to-many/many-to-one relationship, the target side has a reference back to the source entity as well.
* The annotation @OneToMany is used on the side which has the collection reference.
* The annotation @ManyToOne is used on the side which has the single-valued back reference.
* We must use 'mappedBy' element of the @OneToMany annotations to specify that the corresponding table will be the parent table. In other words the other side (which has @ManyToOne) will be the foreign-key table (child table).
* The value of 'mappedBy' element should be the name of the reference variable used in the other class's back reference.
* The side which has 'mappedBy' specified, will be the target entity of the relationship and corresponding table will be the parent of the relationship .
* The side which doesn't have 'mappedBy' element will be the source (owner) and the corresponding table will be the child of the relationship, i.e. it will have the foreign key column.

Example

One ClassRoom has many Students – One To Many

Many Students correspond to a ClassRoom – Many To One

**import** jakarta.persistence.CascadeType;

**import** jakarta.persistence.Column;

**import** jakarta.persistence.Entity;

**import** jakarta.persistence.GeneratedValue;

**import** jakarta.persistence.Id;

**import** jakarta.persistence.OneToMany;

**import** jakarta.persistence.Table;

**import** lombok.Getter;

**import** lombok.Setter;

@Entity

@Table(name = "class\_room")

@Getter

@Setter

**public** **class** ClassRoom {

@Id

@GeneratedValue

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

@Column(name = "name")

**private** String name;

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

**private** Set<Student> students = **new** HashSet<>();

}

**import** jakarta.persistence.CascadeType;

**import** jakarta.persistence.Column;

**import** jakarta.persistence.Entity;

**import** jakarta.persistence.GeneratedValue;

**import** jakarta.persistence.Id;

**import** jakarta.persistence.ManyToOne;

**import** jakarta.persistence.Table;

**import** lombok.Getter;

**import** lombok.Setter;

@Entity

@Table(name = "student")

@Getter

@Setter

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

@Id

@GeneratedValue

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

@Column(name = "name")

**private** String name;

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

**private** ClassRoom clsRoom;

@Override

**public** **int** hashCode() {

**return** Objects.*hash*(clsRoom, name);

}

@Override

**public** **boolean** equals(Object obj) {

**if** (**this** == obj)

**return** **true**;

**if** (obj == **null**)

**return** **false**;

**if** (getClass() != obj.getClass())

**return** **false**;

Student other = (Student) obj;

**return** Objects.*equals*(clsRoom, other.clsRoom) && Objects.*equals*(name, other.name);

}

}

AutoRun classs

@Component

**public** **class** AutoRun {

@Autowired

**private** SampleService service;

@EventListener(value = ApplicationReadyEvent.**class**)

**public** **void** run() {

System.***out***.println("Running ....");

saveAllClassRooms();

// saveClassRoom();

}

**private** **void** saveClassRoom() {

Set<Student> students = **new** HashSet<>();

ClassRoom clsRoom = **new** ClassRoom();

Student s = **new** Student();

s.setName("Johny Depp");

s.setClsRoom(clsRoom);

students.add(s);

clsRoom.setName("Room-2");

clsRoom.getStudents().add(s);

// service.saveClassRoom(clsRoom);

service.saveStudent(s);

System.***out***.println("Information saved successfully ...");

}

**private** **void** saveAllClassRooms() {

List<ClassRoom> clsRooms = **new** ArrayList<ClassRoom>();

**for** (**int** i = 0; i < 10; i++) {

ClassRoom room = **new** ClassRoom();

room.setName("Room-" + i);

clsRooms.add(room);

**for** (**int** j = 0; j < 10; j++) {

Student student = **new** Student();

student.setName("Stud-" + i + j);

room.getStudents().add(student);

student.setClsRoom(room);

}

}

service.saveAllClassRooms(clsRooms);

System.***out***.println("Information saved successfully ...");

}

}