**Cascading in Hibernate:**

Entity relationships often depend on the existence of another entity, for example the *Person*–*Address* relationship. Without the *Person*, the *Address* entity doesn't have any meaning of its own. When we delete the *Person* entity, our *Address* entity should also get deleted.

**Cascading** is the way to achieve it means when we delete person object address object also will get deleted.

If we **persist** person object then address object will also get persist because both are **interconnected** with each other.

Note:

When we perform some action on target entity then same action will be applied to associated entity.

**JPA Cascade Type:**

\*All.

\*Persist.

\*Merge.

\*Remove.

\*Refresh.

\*Detach.

**Hibernate Cascade Type:**

**Hibernate supports three additional cascade type along with JPA cascade type.**

\*Replicate.

\*save update

\*Lock

**CasecadeType.All:**

**CasecadeType.All** propagates(support) all operations –including hibernate-specific ones – from a parent to a child entity.

Note: where ever you provide the mappedBy there itself you have to write cascading code.

**Example:**

@Entity

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String emplyeeName;

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

List<Laptop> laptops;

}

@Entity

public class Laptop {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String name;

@ManyToOne

@JoinColumn

private Employee employee;

}

**Main Class:**

public class EmployeeController {

public static void main(String[] args) {

EntityManagerFactory entityManagerFactory =Persistence.createEntityManagerFactory("ansari");

EntityManager entityManager = entityManagerFactory.createEntityManager();

EntityTransaction entityTransaction =entityManager.getTransaction()

Employee employee = new Employee();

employee.setEmplyeeName("ANSARI");

Laptop laptop = new Laptop();

laptop.setName("HP");

laptop.setEmployee(employee);

Laptop laptop1 = new Laptop();

laptop1.setName("Lenovo");

laptop1.setEmployee(employee);

List<Laptop> laptops = new ArrayList<Laptop>();

laptops.add(laptop);

laptops.add(laptop1);

employee.setLaptops(laptops);

entityTransaction.begin();

entityManager.persist(employee);

entityTransaction.commit();

System.out.println("........................DATA-Stored..................");

}

}

**CascadeType.Persist:**

The persist operation makes a transient instance persistent. Cascade Type *PERSIST* propagates(supports) the persist operation from a parent to a child entity. When we save the *person* entity, the *address* entity will also get saved.

**CascadeType.Merge:**

This will propagate(support) the merge operation from the parent to a child entity.

**Note:**

Here we can merge the one entity from another entity.

**Example:**

@Entity

public class Student {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String StudentName;

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

private List<Subject> subject;

@Entity

public class Subject {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String name;

@ManyToOne

@JoinColumn(name="student\_id")

Student student;

public class StudentController {

public static void main(String[] args) {

EntityManagerFactoryentityManagerFactory=Persistence.createEntityManagerFactory("ansari");

EntityManager entityManager = entityManagerFactory.createEntityManager();

EntityTransaction entityTransaction = entityManager.getTransaction();

Student student = new Student();

Subject subject = entityManager.find(Subject.class,1);

subject.setName("Hindi");

student.setSubject(Arrays.asList(subject));

entityTransaction.begin();

entityManager.merge(student);

entityTransaction.commit();

System.out.println("....................DATA-Modified-Successfully........................");

}

}

### ***CascadeType.REMOVE***

It propagates the remove operation from parent to child entity. Similar to JPA's **CascadeType.Remove.**

We have **CascadeType.Delete,** which is specific to the hibernate There is no difference between the two.

@Entity

public class Person {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int personId;

private String personName;

@OneToOne(mappedBy = "person",cascade = CascadeType.REMOVE)

private Address address;

}

@Entity

public class Address {

@Id

@GeneratedValue(strategy =GenerationType.IDENTITY)

private int id;

private String cityName;

@OneToOne

@JoinColumn(name="person\_id")

private Person person;

}

**Main:**

public class PersonRemoveApp {

public static void main(String[] args) {

EntityManagerFactory entityManagerFactory =Persistence.createEntityManagerFactory("ansari");

EntityManager entityManager = entityManagerFactory.createEntityManager();

EntityTransaction entityTransaction = entityManager.getTransaction();

Person person = entityManager.find(Person.class,1);

entityTransaction.begin();

entityManager.remove(person);

entityTransaction.commit();

System.out.println("...................DATA-Deleted-Successfully..................");

}

}

### ***CascadeType.DETACH:***

The detach operation removes the entity from the persistent context. When we use CascadeType.Detach The child entity will also get removed from the persistent context.

Note:

Here if you will detach your object then we will not able to save child data inside database.

@Entity

public class Person {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int personId;

private String personName;

@OneToOne(mappedBy = "person",cascade = CascadeType.DETACH)

private Address address;

}

@Entity

public class Address {

@Id

@GeneratedValue(strategy =GenerationType.IDENTITY)

private int id;

private String cityName;

@OneToOne

@JoinColumn(name="person\_id")

private Person person;

}

**Main:**

public class PersonApp {

public static void main(String[] args) {

EntityManagerFactory entityManagerFactory = Persistence.createEntityManagerFactory("ansari");

EntityManager entityManager = entityManagerFactory.createEntityManager();

EntityTransaction entityTransaction = entityManager.getTransaction();

Address address = new Address();

Person person = new Person();

person.setPersonName("Abhi");

address.setCityName("Nagpur");

person.setAddress(address);

address.setPerson(person);

entityTransaction.begin();

entityManager.persist(person);

entityManager.detach(person);

entityTransaction.commit();

System.out.println("........................Data Stored Successfully...................."); }

}

### ***CascadeType. REFRESH:***