CURSO DE HIBERNATE 5

OpenWebinars

HIBERNATE

(1) INTRODUCCION

Persistencia, desfase objecto-relacional, ORM. Productos y estándares

(2) HIBERNATE

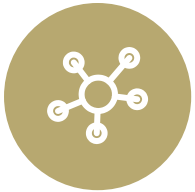
Más que un ORM. Comparativa con otros productos. JPA. Maven. Módulos

(3) PRIMER PROYECTO

Hibernate.cfg.xml, EntityManager y persistence.xml

(4) ENTIDADES

Definición del modelo del dominio. Entidades y ciclo de vida. XML y anotaciones. Tipos de datos.



(5) ASOCIACIONES

ManyToOne, OneToMany, OneToOne, ManyToMany

HIBERNATE

(6) ELEMENTOS AVANZADOS

Campos calculados, herencia.

(7) COLECCIONES

Mapeo de colecciones. Tipos (list, set, map). Colecciones ordenadas (sorted vs. ordered).

(8) GENERACION DEL ESQUEMA

Customización del proceso de generación del esquema.

(9) CONTEXTO DE PERSISTENCIA

Almacenamiento, recuperación y borrado de entidades.

(10) TRANSACCIONES

Control de concurrencia. Patrones y antipatrones.

HIBERNATE

(12)
ENVERS

Introducción a la
auditoria de entidades.

(11)
CONSULTAS
HPQL VS JPQL

Consultas con
parámetros,
Anotaciones. SQL nativo

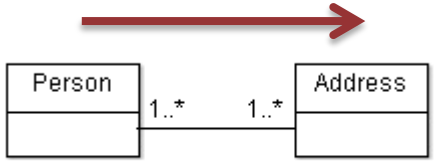
1.

ASOCIACIONES MUCHOS A MUCHOS UNIDIRECCIONALES

ASOCIACIONES MUCHOS A MUCHOS UNIDIRECCIONALES

```java
@ManyToMany(cascade = {CascadeType.PERSIST, CascadeType.MERGE})
private List<Address> addresses = new ArrayList<>();
```

```java
Person person1 = new Person();
Person person2 = new Person();

Address address1 = new Address( "Rue del Percebe", "13" );
Address address2 = new Address( "Av. de la Constitución", "1" );

person1.getAddresses().add(address1);
person1.getAddresses().add(address2);

person2.getAddresses().add(address1);

em.persist(person1);
em.persist(person2);

em.flush();
```

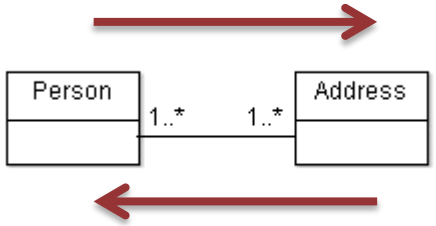


La operación de eliminación de las entidades incluidas en la lista es *muy pesada*, ya que hibernate elimina la lista entera en la base de datos, y vuelve a insertar las entidades que aun quedan incluidas.

2.

ASOCIACIONES MUCHOS A MUCHOS BIDIRECCIONALES

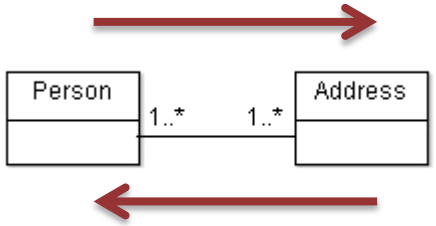
ASOCIACIONES MUCHOS A MUCHOS BIDIRECCIONALES




```java
@ManyToMany(cascade = {CascadeType.PERSIST, CascadeType.MERGE})
private List<Address> addresses = new ArrayList<>();
```

```java
@ManyToMany(mappedBy = "addresses")
private List<Person> owners = new ArrayList<>();
```

ASOCIACIONES MUCHOS A MUCHOS BIDIRECCIONALES



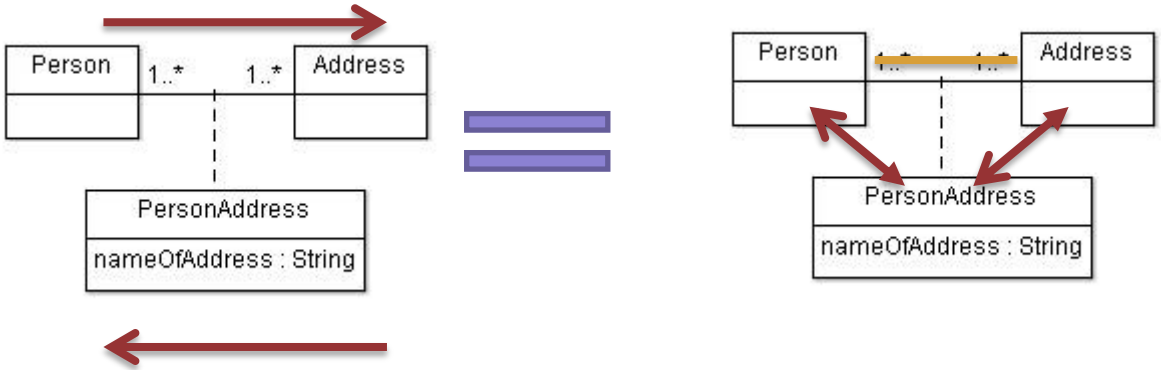

```
public void addAddress(Address address) {
        addresses.add( address );
        address.getOwners().add( this );
}

public void removeAddress(Address address) {
        addresses.remove( address );
        address.getOwners().remove( this );
}
```

Métodos **HELPER**

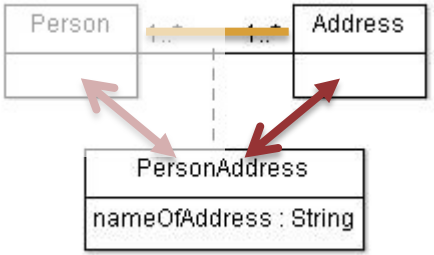
3.

ASOCIACIONES MUCHOS A MUCHOS BIDIRECCIONALES CON ATRIBUTOS EXTRA

ASOCIACIONES MUCHOS A MUCHOS BIDIRECCIONALES CON ATRIBUTOS EXTRA

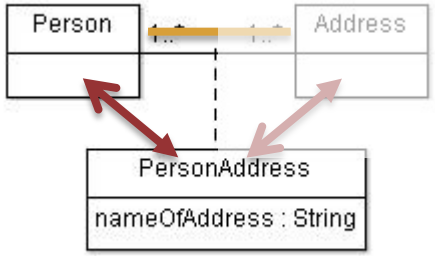


ASOCIACIONES MUCHOS A MUCHOS BIDIRECCIONALES CON ATRIBUTOS EXTRA




```java
@Entity
public class Address {

    @Id
    @GeneratedValue
    private Long id;

    private String street;

    private String number;

    private String postalCode;

    @OneToMany(mappedBy = "address", cascade = CascadeType.ALL, orphanRemoval = true)
    private List<PersonAddress> owners = new ArrayList<>();
```

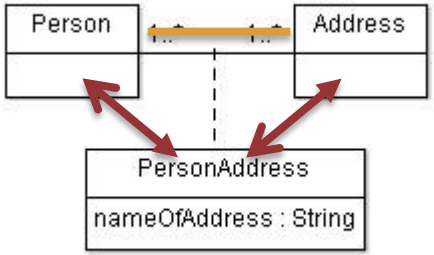
ASOCIACIONES MUCHOS A MUCHOS BIDIRECCIONALES CON ATRIBUTOS EXTRA




```java
@Entity
public class Person {


    @Id
    @GeneratedValue
    private Long id;

    @NaturalId
    private String registrationNumber;

    @OneToMany(mappedBy = "person", cascade = CascadeType.ALL, orphanRemoval = true)
    private List<PersonAddress> addresses = new ArrayList<>();
```

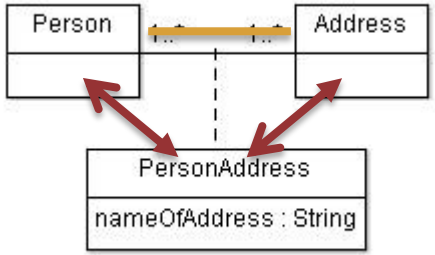
ASOCIACIONES MUCHOS A MUCHOS BIDIRECCIONALES CON ATRIBUTOS EXTRA

```java
@Entity
@IdClass(PersonAddressId.class)
public class PersonAddress {


    @Id
    @ManyToOne
    @JoinColumn(
            name="person_id",
            insertable = false, updatable = false
    )
    private Person person;

    @Id
    @ManyToOne
    @JoinColumn(
            name="address_id",
            insertable = false, updatable = false
    )
    private Address address;
```




```java
public class PersonAddressId implements Serializable {

    private Long person;
    private Long address;

    public PersonAddressId() {

    }

    //Getters, setters, equals y hashCode

}
```

ASOCIACIONES MUCHOS A MUCHOS BIDIRECCIONALES CON ATRIBUTOS EXTRA




```
public void addAddress(Address address, String name) {
    PersonAddress personAddress = new PersonAddress( this, address, name );
    addresses.add( personAddress );
    address.getOwners().add(personAddress);
}

public void removeAddress(Address address) {
    PersonAddress personAddress = new PersonAddress( this, address);
    address.getOwners().remove( personAddress );
    addresses.remove( personAddress );
}
```