

















Hibernate rozumie dziedziczenie i polimorfizm.

Przykłady dotyczą Player i Club

Poniżej klasa Player

**RELACJA ONE TO ONE**

@OneToOne  
@JoinColumn(name = **"club\_player"**)  
**private** Club **club**;

1) Jeżeli chcemy zapisać osobno player i club bez żadnej kaskady

Club club = **new** Club(**"Bayern"**);  
Player player = **new** Player(**"Robert"**, **"Lewandowski"**, 31);  
player.setClub(club);  
session.save(club);  
session.save(player);

2) Zapis kaskadowy

@OneToOne(cascade = CascadeType.***PERSIST***)  
@JoinColumn(name = **"club\_player"**)  
**private** Club **club**;

Zadziała jedynie gdy wykonamy metodę persist na session a nie save

session.persist(player);

*Dziwne. Bo działa także save*

3) cascade = CascadeType.***REMOVE***

Usuwanie powiązanych rekordów także będzie kaskadowane, session.remove()

4) cascade = CascadeType.***REFRESH***

Odświeżenie obiektu, session.refresh()

4) cascade = CascadeType.***DETACH***

Jeśli odłączy obiekt od utrwalania, to także obiekty powiązane z nim zostaną odłączone

4) cascade = CascadeType.***MERGE***

Jeśli podłączymy obiekt do utrwalania, to także obiekty powiązane z nim zostaną podłączone

4) cascade = CascadeType.***ALL***

Podłączenie wszystkich typów kaskad

Jeśli chcemy zrobić dwukierunkową relację tzn nie tylko Player->Club, ale Club->Player (chociaż to trochę bez sensu) to oprócz klasy Player

@OneToOne  
@JoinColumn(name = **"club\_player"**)  
**private** Club **club**;

Trzeba dokonać edycji klasy Club

@OneToOne(mappedBy = **"club"**)  
**private** Player **player**;

RELACJA ONE TO MANY

Club posiada wielu piłkarzy, a każdy piłkarz przypisany jest tylko do jednego klubu.

**Player**

@ManyToOne(cascade = {  
 CascadeType.***PERSIST***,  
 CascadeType.***DETACH***,  
 CascadeType.***MERGE***,  
 CascadeType.***REFRESH***})  
@JoinColumn(name = **"id\_club"**)  
**private** Club **club**;

**Club**

@OneToMany(mappedBy = **"club"**,  
 cascade = {  
 CascadeType.***PERSIST***,  
 CascadeType.***DETACH***,  
 CascadeType.***MERGE***,  
 CascadeType.***REFRESH***})  
**private** List<Player> **players**=**new** ArrayList<>();  
  
**public void** addPlayer(Player player){  
 player.setClub(**this**);  
 **players**.add(player);  
}

**FetchTypes – sposoby pobieranie danych**

Eager – niecierpliwy

Lazy – leniwy





