Relazione Aesthetic

BackEnd

# Entities

Le **entities** sono la rappresentazione dei dati contenuti all’interno di un DB. A fare in modo che i dati siano persistenti, e che quindi i cambiamenti si ripercuotano pure sul DB, vi è l’**entity manager**.

## RELAZIONI

Tra oggetti ci sono spesso e volentieri relazioni. Queste si distinguono per **direzione** e **cardinalità**.

* **Bidirezionali:** Se da un’entità è possibile cercare l’altra e viceversa. Per esempio, data un’entità **attore** possiamo vedere i **film** ai quali è associato. Data un’entità **film** possiamo vedere gli attori che hanno preso parte al cast.
* **Unidirezionale:** Se da un’entità è possibile cercare l’altra ma non è vero il viceversa. Per esempio, data un’entità **Persona** possiamo capire a quale **indirizzo** risiede però non cercheremmo mai una persona a partire da degli indirizzi.
* **Uno ad uno:** ad un’istanza di un’entità e associata una e una sola istanza di un’altra entità. Ad una **persona** è associato un solo **codice fiscale** e viceversa.
* **Uno a molti:** ad un’istanza di un’entità sono associate più istanze di un’altra entità. Ad un **artista** sono associate più **opere.**
* **Molti a uno:** a più istanze di un’entità è associata una e una sola istanza di un’altra entità. Ad una serie di **opere** è associato un unico **artista**
* **Molti a molti**: a più istanze di un’entità sono associate più istanze di un’altra entità. Un **cliente** può effettuare prenotazioni su più crociere e una **crociera** può avere molti clienti

## CASCADE

Le relazioni con entità spesso dipendono dall'esistenza di un'altra entità, ad esempio la relazione Persona-Indirizzo. Senza la Persona, l'entità Address non ha alcun significato proprio. Quando eliminiamo l'entità Persona, anche la nostra entità Address dovrebbe essere eliminata.

**Cascade** è il modo per raggiungere questo obiettivo. Quando eseguiamo qualche azione sull'entità target, la stessa azione verrà applicata all'entità associata.

Ci sono diversi tipi di cascade:

* **ALL**: ogni operazione viene propagata all’entità correlata;
* **PERSIST**: fa in modo che un’istanza “transitoria” (ovvero non ancora persistente) diventi persistente. Quando viene resa persistente l’entità “madre” allora diventano persistenti tutte quelle correlate.
* **MERGE:** copia lo stato dell'oggetto dato sull'oggetto persistente con lo stesso identificatore. Propaga l'operazione di unione da un genitore a un'entità figlio;
* **REMOVE:** rimuove nel DB, e anche nel peristence context, la riga corrispondente all’entità. L’operazione di rimozione viene propagata anche all’entità figlie.
* **REFRESH:** In alcuni casi potremmo voler cambiare una determinata istanza dopo averla resa persistente nel DB. In questo caso il Refresh è un’operazione utile per raggiungere tale scopo. Come per tutti gli altri tipi di Cascade anche questo si ripercuote sulle entità “figlie”.

## NOTAZIONI AGGIUNTIVE

* **@Version:** JPA usa questa notazione per intercettare modifiche concorrenti alla stessa riga del DB che stiamo utilizzando. Quindi serve per poter identificare possibili transazioni che potrebbero entrare in conflitto. Quando vengono intercettate due transazioni in conflitto viene lanciata un’eccezione
* **@JsonIgnore:** serve per fare in modo che in un determinato oggetto JSON quel campo venga ignorato. Per esempio, se ho un’entità **artista** che, come campo, ha una lista di opere, se sul campo List<Opere> metto questa annotazione allora l’oggetto JSON rappresentante l’artista conterrà tutti i campi tranne la lista delle opere associata.

Immagine che contiene testo

Descrizione generata automaticamente

**Immagine che contiene testo

Descrizione generata automaticamente**

* **@ToString.Exclude:** esclude dal metodo toString, se esiste, il campo annotato
* **@Data:** genera automaticamente getter, setter, equals, hashCode, toString

# Repositories

# Services

Nel metodo *addOpera* di *OrdineService* usiamo il metodo *entityManager.merge(opera)*.

Consente di unire le modifiche allo stato apportate a un'entità distaccata nell'archiviazione di persistenza utilizzando il metodo MERGE.

Se non si sta tentando di utilizzare lo stato dell’object caricato in un gestore entity framework in un altro nuovo gestore quadro, **non è necessario utilizzare merge ()**.

Solitamente viene utilizzata nel caso in cui alcune modifiche sono apportate all’object, l’applicazione persiste a queste modifiche chiamando merge () in un secondo gestore.

# Controllers

# Configurations