Laboratorio di informatica – classe 5ª – data 17/11/2017

Biblioteca - Proprietà di navigazione e *lazy loading*

Attualmente l'entità **Libro** si limita a definire i campi **IdGenere** e **IdEditore**, (chiavi esterne della corrispondente tabella), ma non le proprietà **Genere** ed **Editore**, come sarebbe naturale in un modello a oggetti (vedi: *proprietà di navigazione tra gli oggetti*).

Vogliamo aggiungere le due proprietà in modo trasparente al codice applicativo. In sostanza si tratta di applicare la tecnica di *lazy loading* ad entrambe.

Proprietà di navigazione e *lazy loading*

Consideriamo l'entità `Book`, così definita:

```
public class Book
{
    public int BookId { get; set; }
    public string Title { get; set; }
    public int GenreId { get; set; }  // non obbligatorio (anche se utile)
    public Genre Genre { get; set; }
    ...
}
```

Vogliamo che l'accesso alla proprietà `Genre` produca l'oggetto contenente il genere del libro. Due sono le tecniche per ottenere questo risultato:

1. *lazy loading*: finché non si accede alla proprietà, non viene eseguito alcun accesso al database. (quindi `Genre` è `null`)
2. *caching*: una volta caricato il genere in memoria, verrà riutilizzato negli eventuali accessi successivi, senza che sia necessario accedere al database ogni volta.

Di seguito è mostrato lo pseudo-codice necessario per applicare entrambe le tecniche:

```
public class Book
{
    ...
    public int GenreId { get; set; }  // non obbligatorio
    //DEFINISCE CAMPO PRIVATO GENRE
    public Genre Genre
    {
        get
        {
            // SE CAMPO PRIVATO E' NULL (il genere non è stato caricato)
                // ESEGUI QUERY PER CARICARE IL GENERE E SALVALO NEL CAMPO PRIVATO
            // RESTITUISCI IL CAMPO PRIVATO
        }
        set {// ASSEGNA VALORE AL CAMPO PRIVATO}
    }
}
```

Uso di proprietà virtuali e classi derivate

L'approcciò precedente solleva un problema: occorre modificare la classe `Book`, rendendola dipendente dal codice di accesso al database. Al contrario, è opportuno che questa classe, come tutte le classi *entity*, resti un semplice record. Si può risolvere questa contraddizione utilizzando la derivazione virtuale:

1. In `Book` si definisce virtuale la proprietà `Genre` (e qualsiasi altra proprietà di navigazione).
2. Si definisce una classe che deriva da `Book`: esempio: `BookProxy`. Questa ridefinirà la proprietà `Genre`, includendo il codice di *lazy loading+caching*.
3. Nel *repository*, il caricamento dei libri sarà implementato mediante la creazione di oggetti di tipo `BookProxy`.

```
public class BookRepository
{
    ...
    public IEnumerable<Book> GetAll()
    {
        // ACCEDE AL DATABASE, CARICA I LIBRI E RESTITUISCE UNA SEQUENZA DI BOOKPROXY
    }
    ...
}
```

Nota bene: il tipo statico restituito è una sequenza di `Book` e non di `BookProxy`, ma il tipo *run-time* degli oggetti è `BookProxy`, i quali saranno utilizzati in modo completamente trasparente dal codice applicativo.