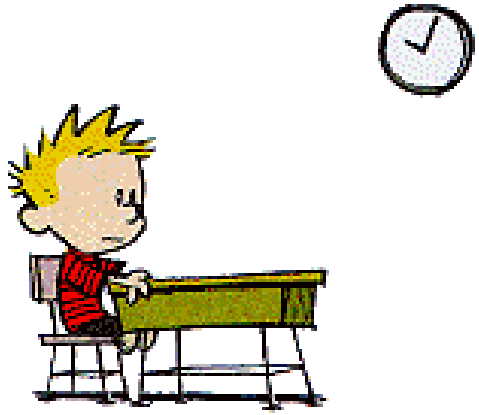
LIVE PROGRAMMING



ovvero: non importa se la scuola è chiusa, vi tocca studiare lo stesso!!!

RIEPILOGO FUNZIONI “EF” 1

- LibraryDemo:
  - Visualizzazione dei libri (applicazione di filtri)
  - Visualizzazione di generi e autori
  - Visualizzazione di un libro (compresa copertina)
  - Visualizzazione di un autore (comprese foto e biografia)
  - Inserimento di un nuovo genere
  - Prestito di un libro

RIEPILOGO FUNZIONI "EF"$_2$

- **Applicazione *lazy loading* (proprietà navigazione):**
  - **Book → Genre (reference property)**
  - **Book → Authors (Collection property)**
  - **Author → Books (Collection property)**
  - **AuthorNote → Author (Reference property)**
  - **(Nota bene: AuthorNote→Author non è "lazy loaded". La proprietà serve solo alla corretta configurazione dell'associazione 1↔1 tra le due entità**

PANORAMICA SUL PROGRAMMA
(senza considerare EF)

- “Cache” delle immagini:
  - Evita il caricamento multiplo della stessa immagine
  - Fornisce un’interfaccia comune per le copertine dei libri e le foto degli autori
- “Cache” delle finestre:
  - BookForm e AuthorForm forniscono dei metodi statici di visualizzazione che evitano il caricamento multiplo della stessa finestra

QUERY IN EF

- Usare il tipo IQueryable: consente a EF di produrre la migliore istruzione SQL
- Assegnare a IEnumerable<> (o a List<>, etc) solo quando la query è completa

ESEMPIO: FILTRO DEI LIBRI

- Se uso subito IEnumerable<>, viene generato l' SQL che carica tutti i libri, soltanto dopo, in memoria, sono filtrati
- Usando IQueryable<>, viene generato l' SQL che applica i filtri stabiliti in C#

(Il tool ExpressProfiler è molto utile in questi scenari)

MODIFICA DEI DATI
(inserimento/eliminazione/aggiornamento)

- Esistono due scenari, convenzionalmente definiti:
  - Connesso
  - Disconnesso
- In entrambi i casi occorre invocare il metodo SaveChanges(), il quale può generare eccezioni (vincoli non soddisfatti, problemi di connettività, etc)

Tutorial Entity Framework, pagina 35: “Uso in scenari N-Tier”

SCENARIO CONNESSO

- **Viene usato lo stesso oggetto *context* per caricare i dati e per inviare al database le modifiche effettuate**
- **Non esistono problematiche particolari, perché il *context* tiene "traccia" di tutte le modifiche fatte alle entità e dunque è in grado di generare le istruzioni SQL corrette**

SCENARIO DISCONNESSO

- **Viene usato un oggetto *context* per caricare i dati e un altro oggetto *context* per inviare al database le modifiche**
- **Possono esserci dei problemi, perché il *context* che invia le istruzioni SQL al database "non ha traccia" delle modifiche effettuate alle entità (questa traccia ce l'ha il primo *context*)**

INSERIMENTO

I casi più semplici non richiedono spiegazioni

(non importa come e dove viene creato il "context")

```
Library db = new Library();
//...
Genre g = new Genre { Name = "Filosofia"};
db.Genres.Add(g);
db.SaveChanges();  // viene recuperato l'Id del genere
```

Nei casi "meno semplici" esistono considerazioni da fare:

(Tutorial EF, pagina 35: "Uso in scenari N-Tier, paragrafo 6.12: Inserimento...)

AGGIORNAMENTO 1

Scenario connesso (nessun problema):

```
Library db = new Library();
var g = db.Genres.First();
g.Name = "Modificato";
db.SaveChanges();
```

AGGIORNAMENTO 2

- **Scenario disconnesso: il *context* che esegue l'aggiornamento non genera l'istruzione corretta**
- **Occorre informarlo che l'entità è stata modificata:**

```
// genere caricato da un altro context
0 references
static void UpdateGenre(Genre g)
{
    Library db = new Library();
    var en = db.Entry<Genre>(g);
    en.State = EntityState.Modified;
    db.SaveChanges();
}
```

ELIMINAZIONE 1

- **Scenario connesso (nessun problema):**

```
Library db = new Library();
var g = db.Genres.Find(10);  //-> genere "Cancellami"
db.Genres.Remove(g);
db.SaveChanges();
```

ELIMINAZIONE$_2$

- **Scenario disconnesso: il *context* che esegue l'eliminazione non genera l'istruzione corretta**
- **Occorre informarlo che l'entità è stata eliminata:**

```
// genere caricato da un altro context
1 reference
static void DeleteGenre(Genre g)
{
    Library db = new Library();
    var en = db.Entry<Genre>(g);
    en.State = EntityState.Deleted;
    db.SaveChanges();
}
```

SCENARI REALI

- Negli esempi precedenti ho eseguito modifiche di singole entità
- Negli scenari reali accade di dover modificare entità correlate (grafi di entità). Alcuni esempi:
  - Inserire un libro di un determinato genere, esistente
  - Inserire un libro di un nuovo genere
  - Cambiare il genere di un libro
  - Eliminare un autore (e i libri che appartengono)

SCENARI REALI

- **In questi casi, dover gestire uno scenario disconnesso risulta sicuramente più complicato**
- **Le applicazioni web rappresentano un classico esempio di scenario disconnesso, poiché:**

il processo (il context) che carica i dati non è lo stesso processo (lo stesso context) che li salva sul database

(Tutorial EF, pagina 35: “Uso in scenari N-Tier”)