

Curs 3

Accesul la date – Entity Framework Core

Entity Framework - ADO.NET API

- Introdus începând cu versiunea .NET 3.5
- EF Core – versiune multi-platforma a lui EF
- EF Core – (O/RM)- Object-Relational Mapper:
 - Permite dezvoltatorilor .NET sa lucreze cu bd utilizand obiecte .NET
 - Elimina necesitatea scrierii de cod specific pentru accesul la date
- Un set de date - o colecție de rânduri și coloane => colecție obiecte tipizate denumite entități
- Aceste entități pot fi interogate utilizând LINQ, motorul EF va traduce LINQ în interogări SQL.



Rolul entitatilor

- Entitatile reprezinta un model conceptual al bazei de date - EDM (Entity data model)
- EDM – un set de clase client-side care sunt mapate la o baza de date pe baza conventiilor definite in EF si pe baza unor configurari
- Creare Model – abordari:
 - Generarea unui model dintr-o baza de date existenta
 - Scrierea de mana a claselor care reprezinta modelul -> Migrarea pentru a crea baza de date din model.

Componente EF

1. Clasa DbContext - utilizata pentru interogarea bazei de date si pentru a grupa modificarile pentru a putea fi scrise in bloc
 - Metoda SaveChanges – salvează in baza de date toate modificările făcute in context. Returneaza numarul de entitati efectuate
 - Proprietatea Database – ofera un mechanism pentru creare/stergerea/verificarea bazei de date, execută proceduri stocate si expune funcționalități legate de tranzacții
 - Evenimente SavingChanges – se apeleaza când modificările se salveaza în baza de date, dar înainte de a deveni persistente

Componente EF

2. Clasa derivată din DbContext – se trimite la constructor numele string-ului de conexiune pentru clasa context

```
public Nume_Pren_Lab2Context (DbContextOptions<Nume_Pren_Lab2Context>  
options)  
    : base(options)  
{  
  
}
```

Componente EF

3. Entity Sets DbSet<T> - adaugarea de tabele in context

- `public DbSet<Nume_Pren_Lab2.Models.Book> Book { get; set; }`
- `public DbSet<Nume_Pren_Lab2.Models.Publisher> Publisher { get; set; }`
- `public DbSet<Nume_Pren_Lab2.Models.Category> Category { get; set; }`

Membrii DbSet<T>:

- Add – adaugarea unui obiect in colectie ; acestea vor fi marcate cu starea Added si vor fi inserate in baza de date cand se apeleaza SaveChanges pentru DbContext
- Find – gasește un rand dupa cheia primară și returnează un obiect reprezentând acel rând
- Remove – marcheaza un obiect pentru ștergere

Componente EF

4. DbChangeTracker - realizează tracking-ul automat al stării oricărui obiect `DbSet<T>` în cadrul unui `DbContext`

Stările entităților:

1. Detached

2. Unchanged

3. Added

4. Deleted

5. Modified

- a. obiectul nu a fost modificat de când a fost atașat la context sau de la ultimul apel a lui `SaveChanges()`
- b. obiectul a fost marcat pentru stergere
- c. obiectul există, dar nu se face încă tracking pe el; se afla în această stare imediat ce a fost creat și înainte să fie adăugat la obiectul context
- d. una din proprietățile obiectului a fost modificată și metoda `SaveChanges()` nu a fost apelată
- e. obiectul este nou și a fost adăugat la obiectul context, iar metoda `SaveChanges()` nu a fost apelată.

Componente EF

5. Adnotări – reprezintă attribute utilizate pentru modelarea entităților în vederea mapării cu bd

- | | | |
|----------------------|---|--|
| 1. Key | → | a. definește o proprietate care este utilizată ca și cheie străină |
| 2. Required | → | b. marchează un camp pentru a fi verificat în cazuri de concurență când se realizează inserări, actualizări sau ștergeri |
| 3. ForeignKey | → | c. o proprietate nu este mapată pe un câmp al bazei de date |
| 4. NotMapped | → | d. definește cheia primară. Nu este necesară dacă proprietatea se numește Id sau combina numele clasei cu Id- ex. |
| 5. ConcurrencyCheck | → | e. permite numirea claselor și campurilor diferit față de numele din baza de date. Atributul table permite specificarea inclusiv a numelui schemei bazei de date |
| 6. Table/ Column | → | f. DatabaseGenerated – specifică faptul că un câmp este generat din baza de date cum ar fi Identity |
| 7. DatabaseGenerated | → | g. proprietățile nu pot lua valori null |

Modelarea relatiilor intre entitati

- O relatie defineste modul in care doua entitati se raporteaza una la cealaltata
- Intr-o baza de date relationala ->constrangere de cheie straina

Definire termeni

- Entitate dependenta : O entitate care contine proprietati de tip cheie straina. Este referita uneori ca si entitatea copil din relatie
- Entitate principala: Entitatea care contine proprietatea de tip cheie primara. Este referita uneori ca si entitatea parinte din relatie
- Navigation property : O proprietate definita in entitatea principala si/sau dependenta care referentiaza entitatea relationata
 - **Collection navigation property:** Contine referinta la mai multe entitati
 - **Reference navigation property:** Contine referinta la o singura entitate

Entitate principala-dependenta

```
public class Blog
{
    public int BlogId { get; set; }
    public string Url { get; set; }
    public List<Post> Posts { get; set; }
}
```

```
public class Post {
    public int PostId { get; set; }
    public string Title { get; set; }
    public string Content { get; set; }
    public int BlogId { get; set; }
    public Blog Blog { get; set; }
}
```

Relatii definite complet

- Avem navigation properties definite la ambele capete ale relatiei si o proprietate de tip cheie straina definite in entitatea dependenta

```
public class Blog {  
    public int BlogId { get; set; }  
    public string Url { get; set; }  
    public List<Post> Posts { get; set; }  
}  
  
public class Post {  
    public int PostId { get; set; }  
    public string Title { get; set; }  
    public string Content { get; set; }  
    public int BlogId { get; set; }  
    public Blog Blog { get; set; }  
}
```

No foreign key property

- Desi este recomandat sa definim o proprietate de tip cheie straina in entitatea dependenta, nu este obligatoriu
- Daca nu este gasita o cheie straina se creaza automat o proprietate de tip cheie straina shadow
- Proprietatile shadow – proprietati care nu sunt definite in clasa entitate .NET dar sunt definite pentru acea entitate in modelul EF.
 - valoarea si starea acelor entitati sunt gestionate de ChangeTracker

No foreign key property

```
public class Blog
{ public int BlogId { get; set; }
  public string Url { get; set; }
  public List<Post> Posts { get; set; }
}

public class Post {
  public int PostId { get; set; }
  public string Title { get; set; }
  public string Content { get; set; }
  public Blog Blog { get; set; }
}
```

Configurare manuala prin Fluent API

```
class MyContext : DbContext {  
    public DbSet<Blog> Blogs { get; set; }  
    public DbSet<Post> Posts { get; set; }  
    protected override void OnModelCreating(ModelBuilder modelBuilder)  
    {  
        modelBuilder.Entity<Post>()  
            .HasOne(p => p.Blog)  
            .WithMany(b => b.Posts);  
    }  
}
```

Cascade Delete

```
protected override void OnModelCreating(ModelBuilder
modelBuilder)
{
    modelBuilder.Entity<Post>()
        .HasOne(p => p.Blog)
        .WithMany(b => b.Posts)
        .onDelete(DeleteBehavior.Cascade);
}
```


Relatie one-to-one

-au un reference navigation property la ambele capete ale relatiei

```
public class Blog
{
    public int BlogId { get; set; }
    public string Url { get; set; }
    public BlogImage BlogImage {
        get; set; }
}
```

```
public class BlogImage
{
    public int BlogImageId { get;
        set; }
    public byte[] Image { get; set;
    }
    public string Caption { get;
        set; }
}
```

```
public int BlogId { get; set; }
public Blog Blog { get; set; } }
```

Configurare manuala cu Fluent API

```
class MyContext : DbContext
{
    public DbSet<Blog> Blogs { get; set; }
    public DbSet<BlogImage> BlogImages { get; set; }
    protected override void OnModelCreating(ModelBuilder modelBuilder) {
        modelBuilder.Entity<Blog>()
            .HasOne(b => b.BlogImage)
            .WithOne(i => i.Blog)
    }
}
```

Relatie many-to-many

-necesita un collection navigation property la ambele capete

```
public class Post
{
    public int PostId { get; set; }
    public string Title { get; set; }
    public string Content { get; set; }
    public ICollection<Tag> Tags { get; set; }
}
```

```
public class Tag
{
    public string TagId { get; set; }
    public ICollection<Post> Posts {
        get; set; }
}
```

Relatie many-to-many

entitate de tip join

```
public class Post
{ public int PostId { get; set; }
  public string Title { get; set; }
  public string Content { get; set; }
  public List<PostTag> PostTags { get;
  set; }
}
```

```
public class Tag
{ public string TagId { get; set; }
  public List<PostTag> PostTags { get;
  set; }
}
```

```
public class PostTag {
  public DateTime PublicationDate {
  get; set; }
  public int PostId { get; set; }
  public Post Post { get; set; }
  public string TagId { get; set; }
  public Tag Tag { get; set; }
}
```

Configurare manuala cu Fluent API

```
public class MyContext : DbContext
{
    public MyContext(DbContextOptions<MyContext> options) : base(options) {
    }

    protected override void OnModelCreating(ModelBuilder modelBuilder) {
        modelBuilder.Entity<PostTag>()
            .HasKey(t => new { t.PostId, t.TagId });

        modelBuilder.Entity<PostTag>()
            .HasOne(pt => pt.Post)
            .WithMany(p => p.PostTags)
            .HasForeignKey(pt => pt.PostId);

        modelBuilder.Entity<PostTag>()
            .HasOne(pt => pt.Tag)
            .WithMany(t => t.PostTags)
            .HasForeignKey(pt => pt.TagId); } }
```

Migrarea

- In procesul de dezvoltarea a unei aplicatii modelul se schimba frecvent si nu mai este sincronizat cu baza de date
- La modificarea modelului-adaugare, stergere, modificare de entitati stergem baza de date si EF creaza o noua baza de date corespunzatoare modelului si apeleaza Seed Data
- In productie avem date in baza de date -> nu putem sterge baza de date
- EF Migration – actualizeaza baza de date

Add-Migration ExtendedModel

- EF genereaza cod care va creaza baza de date de la 0
- Directorul Migrations - fisier *<timestamp>_ExtendedModel.cs*

```
public partial class ExtendedModel : Migration
{
    protected override void Up(MigrationBuilder migrationBuilder)
    {
        migrationBuilder.CreateTable(
            name: "Book",
            columns: table => new
            {....
        }
    }
}
```

Remove-Migration ExtendedModel

```
protected override void Down(MigrationBuilder migrationBuilder)
{
    migrationBuilder.DropTable(
        name: "Books");
}
```

Metoda Down se apeleaza cand fac rollback la migrare

SnapShot pentru model

- Migrarea creaza un *snapshot* a schemei bazei de date curente
- Directorul *Migrations/LibraryContextModelSnapshot.cs*
- Cand creem o noua migrare, EF determina ce s-a modificat comparand modelul curent cu modelul din snapshot