

Dr. Hajnal Éva: Haladó
programozás

HALADÓ PROGRAMOZÁS

Entity Framework Code first stratégia

EF ORM Structure

EF in Application

DB1Context

DbSet<student> Students

DbSet<course> Courses

DB Server

DB1

Student

Course

Lépések

- Consol application project
- Add/New item/Student.cs
- Add/New item/DBContext
- Main/kód
- Adatbázis automatikusan elkészül

Osztály létrehozása

- Nuget/Entity Framework
- Using System.Data.ComponentModel

```
public class Student
{
    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    public int StudentId { get; set; }
    public string Name { get; set; }
    public string Adress { get; set; }
}
```

Data Access Layer létrehozása

```
internal class SchoolDBContext:DbContext
{
    public DbSet<Student> Students { get; set; }
}
```

Programozás

```
static void Main(string[] args)
{
    using (var db = new SchoolDBContext())
    {
        Console.WriteLine( "Insert data");
        string name=Console.ReadLine();
        string adress=Console.ReadLine();
        var student = new Student()
        { Name=name, Adress=adress};
        db.Students.Add(student);
        db.SaveChanges();
    }
}
```

Tranzakció-kezelés

- static void TransactedSaveChanges()
- {
- var context = new ApplicationDbContextFactory().CreateDbContext(null);
- **using var trans = context.Database.BeginTransaction();**
- try
- {
- //Create, change, delete stuff
- context.SaveChanges();
- **trans.Commit();**
- }
- catch (Exception ex)
- {
- **trans.Rollback();**
- }

Adatbázis létrehozása

A program futtatásakor
automatikusan létrejön

Connection string bekerül a
konfigurációs állományba
(app.config)

Ez alapján a későbbi futtatáskor az
adatbázis kapcsolat automatikusan
felépül.

Hol van az adatbázis?
SQLServerObject Explorer/Properties

Modell megváltoztatása

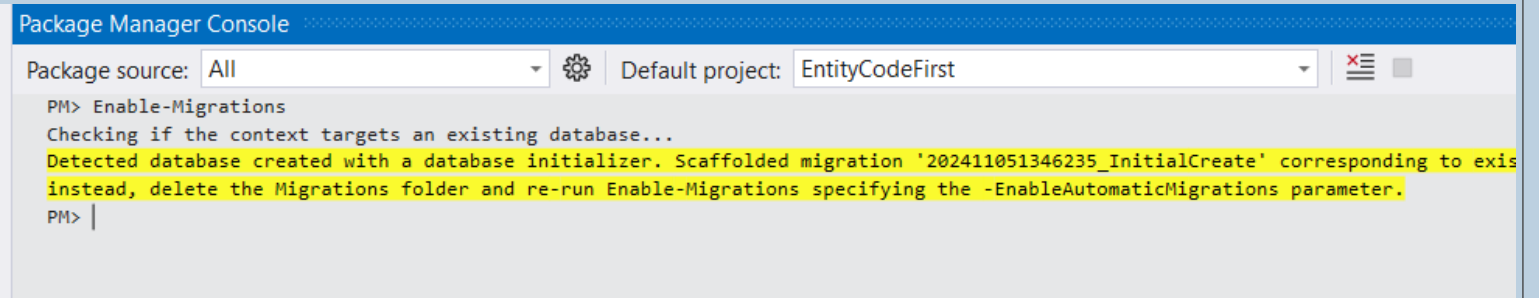
- Adatmigrációval jár- a kód már nem tükrözi az adatbázist- át kell görgetni a változtatást az adatbázisba

1. Engedélyezni a Migrációt
Package Manager konzol

Enable-Migrations

2. Elkészíteni a migrációs kódot
Add-Migration Neve

3. Lefuttatni a migrációs kódot
Update-Database



```
Package Manager Console
Package source: All
Default project: EntityCodeFirst
PM> Enable-Migrations
Checking if the context targets an existing database...
Detected database created with a database initializer. Scaffolded migration '202411051346235_InitialCreate' corresponding to existing database.
instead, delete the Migrations folder and re-run Enable-Migrations specifying the -EnableAutomaticMigrations parameter.
PM> |
```


Member of DbContext	Meaning in Life
Database	Provides access to database-related information and functionality, including execution of SQL statements.
Model	The metadata about the shape of entities, the relationships between them, and how they map to the database. Note: This property is usually not interacted with directly.
ChangeTracker	Provides access to information and operations for entity instances this DbContext is tracking.
DbSet<T>	Not truly a member of DbContext, but properties added to the custom derived DbContext class. The properties are of type DbSet<T> and are used to query and save instances of application entities. LINQ queries against DbSet<T> properties are translated into SQL queries.
Entry()	Provides access to change tracking information and operations for the entity, such as explicitly loading related entities or changing the EntityState. Can also be called on an untracked entity to change the state to tracked.
Set<TEntity>()	Creates an instance of the DbSet<T> property that can be used to query and persist data.
SaveChanges()/SaveChangesAsync()	Saves all entity changes to the database and returns the number of records affected. Executes in a transaction (implicit or explicit).
Add()/AddRange() Update()/UpdateRange() Remove()/RemoveRange()	Methods to add, update, and remove entity instances. Changes are persisted only when SaveChanges() is executed successfully. Async versions are available as well. Note: While available on the derived DbContext, these methods are usually called directly on the DbSet<T> properties.
Find()	Finds an entity of a type with the given primary key values. Async versions are available as well. Note: While available on the derived DbContext, these methods are usually called directly on the DbSet<T> properties.

Tervezési részletek- DbContext osztály

Tervezési részletek

- Hogyan készítek I úgy az osztályokat, hogy abból olyan adatbázis generálódjon, amit szeretnék?
- POCO (Plain Old Class Object)+ névkonvenciók alkalmazása

Saját konvenciók kialakítása



Data
annotációkkal

Fluent API
használatával

Adat annotációk

Annotációk példa- attributumok

```
public class Student
{
    public Student()
    { }
    [Key]
    public int SID { get; set; }

    [Column("Name", TypeName="ntext")]
    [MaxLength(20)]
    public string StudentName { get; set; }

    [NotMapped]
    public int? Age { get; set; }

    public int StdId { get; set; }

    [ForeignKey("StdId")]
    public virtual Standard Standard { get; set; }
}
```

- [KeyAttribute](#)
- [StringLengthAttribute](#)
- [MaxLengthAttribute](#)
- [ConcurrencyCheckAttribute](#)
- [RequiredAttribute](#)
- [TimestampAttribute](#)
- [ComplexTypeAttribute](#)
- [ColumnAttribute](#)
- [TableAttribute](#)
- [InversePropertyAttribute](#)
- [ForeignKeyAttribute](#)
- [DatabaseGeneratedAttribute](#)
- [NotMappedAttribute](#)

Data annotációk a ComponentModel névtérben

- Az Attribute nem íródik a használatkor
- Pl. [Key]

Fluent API

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    // Fluent API calls go here
}
```

Fluent API

```
public SchoolDBContext():  
base("SchoolDBConnectionString")  
{ }  
public DbSet<Student> Students { get; set; }  
public DbSet<Standard> Standards { get; set; }  
public DbSet<StudentAddress> StudentAddress { get; set; }  
protected override void OnModelCreating(DbModelBuilder  
modelBuilder)  
{ //Configure domain classes using modelBuilder here.. }
```

DBModelBuilder hívja a Fluent API-t.

Példák Fluent API használatára

- Context tulajdonságainak beállítása
- Kaszkádolt/korlátozott stb. tulajdonság beállítása
- Névkonvenciók beállítása

DAL példa – adattábla létrehozás

```
public class SchoolContext : DbContext
{
    // "SchoolContext" a connection string neve, ami az adatbázis
    // hozzáférést definiálja - a connectionstring a Web.config fileban van.
    public SchoolContext() : base("SchoolContext")
    { }

    public DbSet<Student> Students { get; set; }
    public DbSet<Enrollment> Enrollments { get; set; }
    public DbSet<Course> Courses { get; set; }
    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    { modelBuilder.Conventions.Remove<PluralizingTableNameConvention>();
      } }
```

Köszönöm a figyelmet!