Dr. Hajnal Éva: Haladó programozás

HALADÓ PROGRAMOZÁS

SQL server elérése Entity Framework módszerrel

Haladó Programozás

Adatbázis-elérési módszerek összehasonlítása
Adatbázisok elérése DbConnection/DbReader módszerrel
SQL server elérése DataSet módszerrel
SQL server elérése Entity Framework módszerrel

ORM fogalma

- Object Relational Mapping
  - Kétirányú leképezés az adatbázis SQL rendszere és a szoftver objektumorientált világa között.
  - A programban az adatbázis elemei programelemekre képződnek le.
  - Segítségével objektum-orientált adatbázis kezelő szoftvert készíthetünk (desktop, webes alkalmazást)

Entity Framework (LINQ: to SQL / to Entities)

- ADO.NET Entity Framework (+LINQ to Entites)
  - Teljes ORM
  - N:M kapcsolatokat is támogat
  - Alternatív adatbázis-dialektusokkal is működik/het (Oracle, MySQL...)
  - **Entity osztályok létrehozása: Data sources / Add new data source / Database / Entity Data Model**
  - **Újabb VS verzióknál: Project / Add New Item / Data / ADO.NET Entity Data Model**

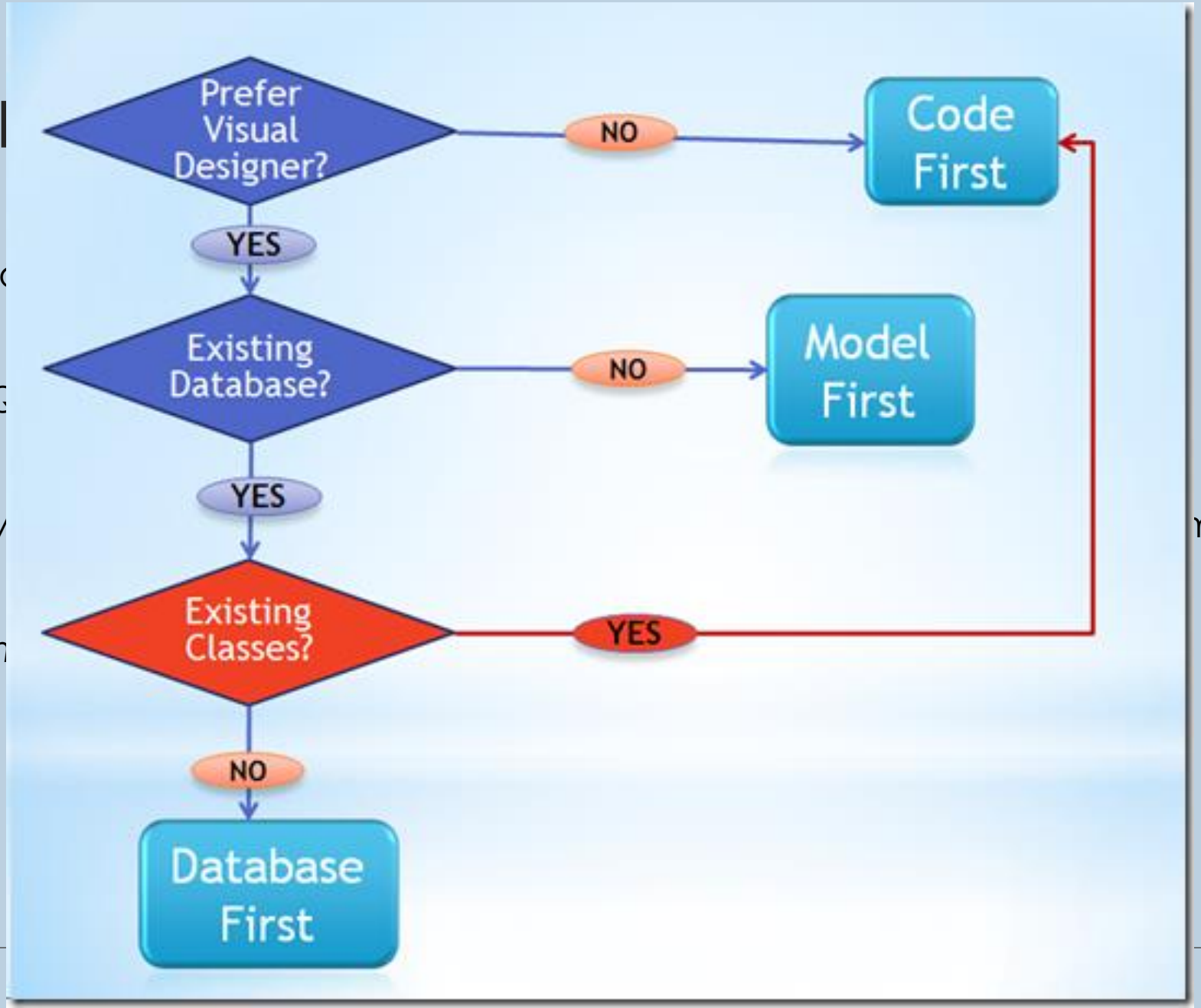
- LINQ to SQL
  - Formailag nagyon hasonló
  - **Régebbi módszer, NEM UGYANAZ!**
  - Csak közvetlen leképezést, és csak MSSQL dialektust támogat
  - Egyszerű és gyors, de nagyon korlátozottan használható („Rapid development")
  - Osztályok létrehozása: Project/Add Class/LINQ to SQL classes

Entity

- Code First: El
- Database/SG
- Model First: M

mot)

(http://msdn.m



Entity Framework verziók

- EF1 = EF3.5 ➔ .NET 3.5
- EF4 ➔ .NET 4 „POCO support, lazy loading, testability improvements, customizable code generation and the Model First workflow”
- EF4.1 ➔ „first to be published on NuGet. This release included the simplified DbContext API and the Code First workflow” ➔ Jó!
- EF4.3 ➔ „Code First Migrations” ➔ Teljesen használható ORM!
- EF5, EF6, EF 6.1 ➔ https://msdn.microsoft.com/en-us/data/jj574253.aspx
- Frissíthető (nekünk nem kell, jó a beépített):
  > NuGet> Install-Package EntityFramework
  (esetleg: -Version x.x.x)
  > Entity Framework 6 Tools for Visual Studio:
  http://www.microsoft.com/en-us/download/details.aspx?id=40762

Start

- **NuGet Package Manager**
- **Install Entity Framework**

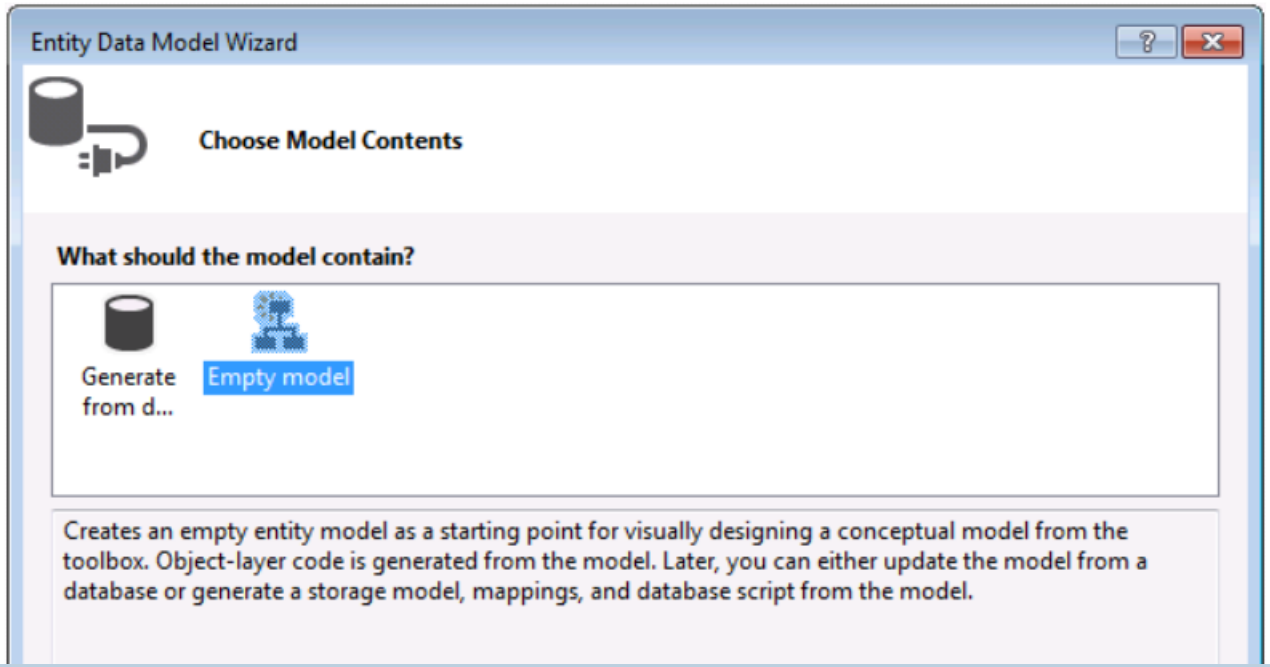
Model First Stratégia

1. Create application

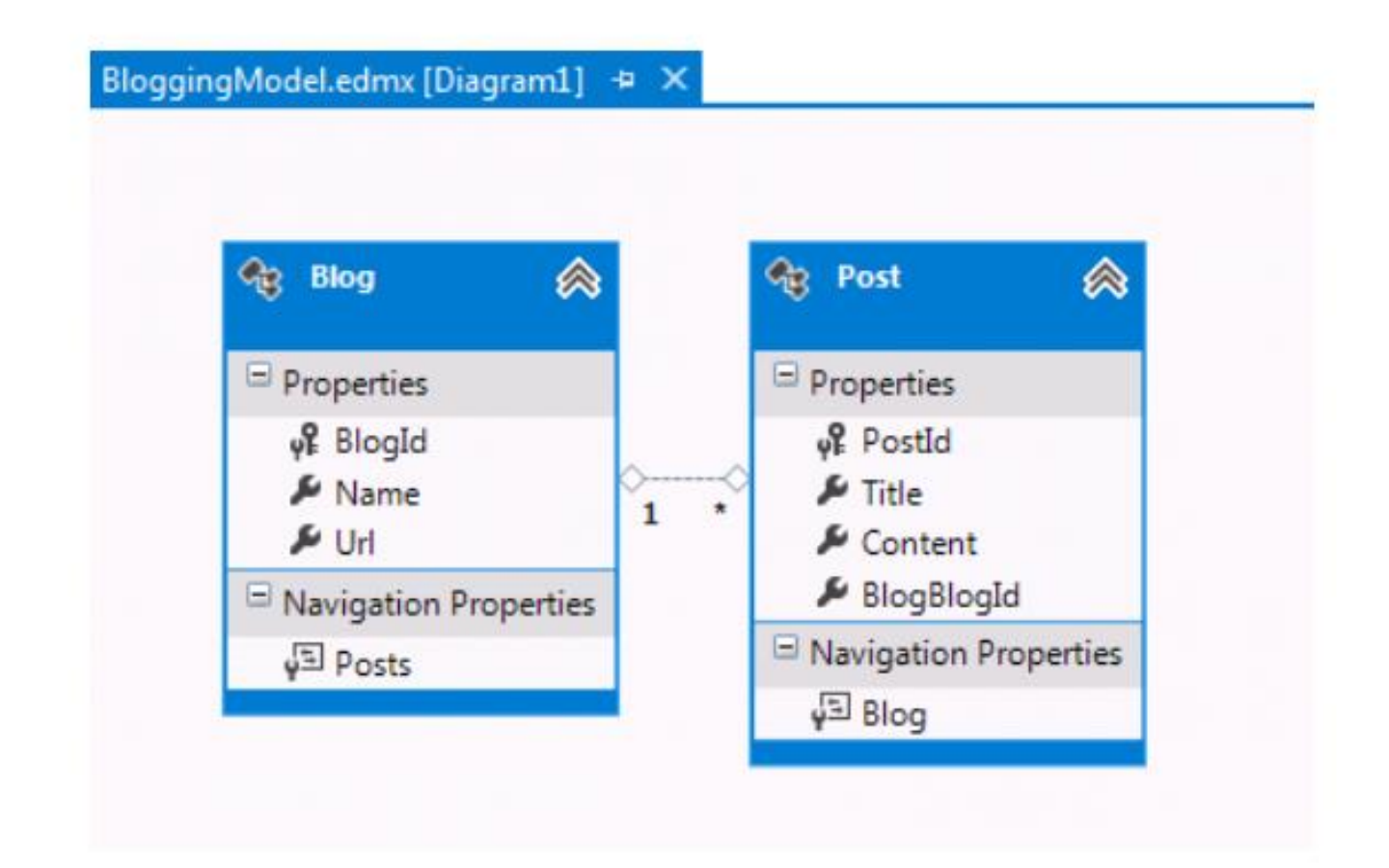
- Open Visual Studio
- **File -> New -> Project...**
- Select **Windows** from the left menu and **Console Application**
- Enter **ModelFirstSample** as the name
- Select **OK**

2. Cre

- **Project -> Add New Item...**
- Select **Data** from the left menu and then **ADO.NET Entity Data Model**
- Enter **BloggingModel** as the name and click **OK**, this launches the Entity Data Model Wizard
- Select **Empty Model** and click **Finish**



3. Add entities



4. Generating database

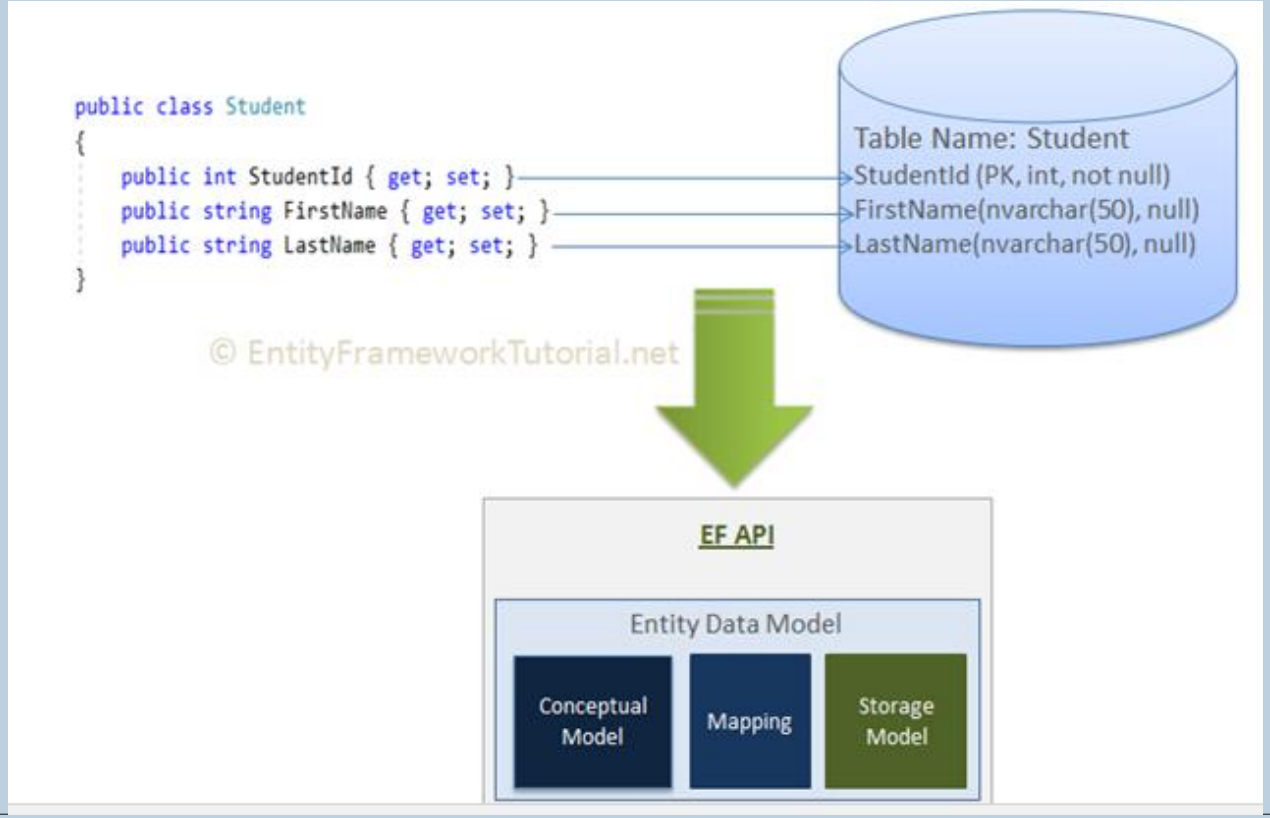
- Right-click on the design surface and select **Generate Database from Model...**
- Click **New Connection...** and specify either LocalDB or SQL Express, depending on which version of Visual Studio you are using, enter **ModelFirst.Blogging** as the database name.

5. Creation of code

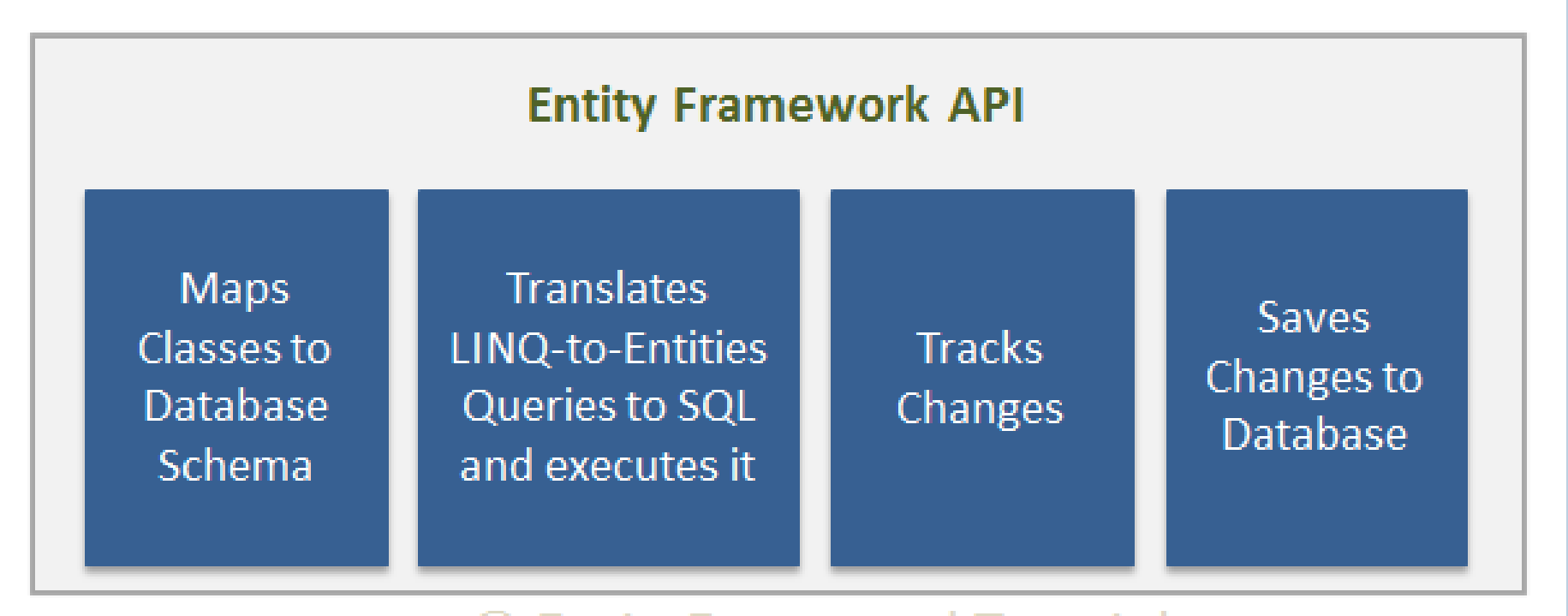
```
class Program
{
    static void Main(string[] args)
    {
        using (var db = new BloggingContext())
        {
            // Create and save a new Blog
            Console.Write("Enter a name for a new Blog: ");
            var name = Console.ReadLine();

            var blog = new Blog { Name = name };
            db.Blogs.Add(blog);
            db.SaveChanges();

            // Display all Blogs from the database
            var query = from b in db.Blogs
                        orderby b.Name
```

Entity Data Model

- The very first task of EF API is to build an Entity Data Model (EDM). EDM is an in-memory representation of the entire metadata: conceptual model, storage model, and mapping between them.

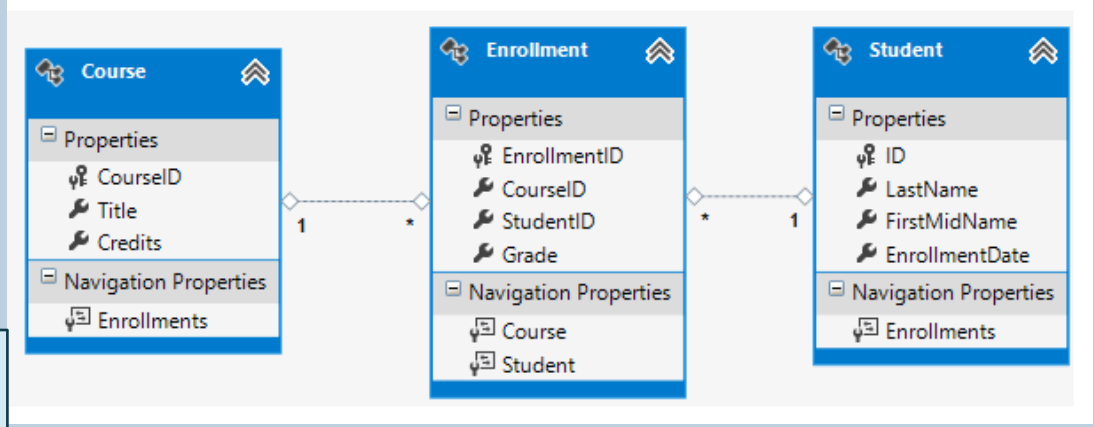


EF API


Entity Framework API
Maps Classes to Database Schema
Translates LINQ-to-Entities Queries to SQL and executes it
Tracks Changes
Saves Changes to Database

Model first stratégia használata

1. **Modell kialakítása**



2. **Adatbázis kialakítás a modell alapján**
3. **Kód kialakítás az adatbázis alapján**

```
public class Student
{
public int ID { get; set; }
public string LastName { get; set; }
public string FirstMidName { get; set; }
public DateTime EnrollmentDate { get; set; }
//navigációs tulajdonság, az 1:N kapcsolatot
képezi le
public virtual ICollection<Enrollment>
Enrollments { get; set; } }
```

Course osztály

```
public class Course
{
//Elsődleges kulcs manuálisan beállítható legyen
[DatabaseGenerated(DatabaseGeneratedOption.None)]
public int CourseID { get; set; } public string Title { get; set; } public
int Credits { get; set; } public virtual ICollection<Enrollment>
Enrollments { get; set; }
 }
```

Data **A**ccess **L**ayer (DAL)

- Az a forrás file, ami az adatbázis hozzáférést megvalósítja
- Legfontosabb osztálya: System.Data.Entity.DbContext
- Adatelem DbSet, egy táblának megfeleltethető gyűjtemény


DATABASE FIRST
STRATÉGIA

LocalDB létrehozása

- In-Solution
  - Project / Add New Item / Service-based Database (ez SQL server file lesz. Local Database: SQL server compact file)
  - Nekünk mindkettő jó, service-based-et használunk: EmpDept
  - Előbb feltöltjük az adatbázist, majd azután generálunk osztályokat
- In-Profile
  - Server Explorer -> Right click (Data Connections) -> Add Connection (Microsoft SQL Server + .Net provider for SQL Server <OK>)
  - Server name = (localdb)\v11.0 , Database name = EMPDEPT <OK>
  - "Database does not exist. Attempt to create?" <YES>
  - Akár a „Create New SQL Server Database" is jó, ugyanilyen lépésekkel

Adatbázis feltöltése

- Új DB-re jobbkatt, New query, a létrehozó SQL -ből mindent copypaste, Execute, ez után a query bezárható (Új db /tables- re jobbkatt, refresh: táblák megjelennek)
- Project, Add new Item, ADO.NET Entity Data Model <ADD>; Generate from database <NEXT>; az MDF file legyen a legördülő menüben + save connection settings <NEXT>; EF6.0 <NEXT>; Mindegyik tábla mellett pipa + Model namespace = EmpDeptModel <FINISH>
- Konfigurációtól függően: Template can harm your computer, click ok to run ... <OK> <OK>

(http://msdn.microsoft.com/en-us/data/jj206878)

- Eredmény: automatikusan generált osztályok (mint DataSetnél), csak ezek nagyrészt generikus osztályok típusparaméterezett változatai ➔ ~30KB a két táblás adatbázis

1. Inicializálás

```
ED = new EmpDeptEntities();
Console.WriteLine("Connect OK");


var reszleg = ED.DEPT.First();
Console.WriteLine(reszleg.DNAME);
var dolg = from dolgozo in ED.EMP where     dolgozo.ENAME.Contains("E") select
dolgozo;
Console.WriteLine(dolg.Count());
```

2. INSERT

```
var ujdolg = new EMP()
{
    ENAME = "BELA",
    MGR = null,
    DEPTNO = 20,
    EMPNO = 1000
};
ED.EMP.Add(ujdolg); // régen: AddObject
ED.SaveChanges();
Console.WriteLine("Insert OK");
```

3. UPDATE

```
var valaki = ED.EMP.Single(x => x.EMPNO == 1000);

valaki.ENAME = "JOZSI";

ED.SaveChanges();

Console.WriteLine("Update OK");
```

4. DELETE

```
var valaki = ED.EMP.Single(x => x.EMPNO == 1000);

ED.EMP.Remove(valaki); // régen: DeleteObject

ED.SaveChanges();

Console.WriteLine("Delete OK");
```

5. SELECT

```
string s = "", sep="";
foreach (var dolg in
    ED.EMP.Where(dolgozo => dolgozo.SAL >= 3000))
{
    s += sep + dolg.ENAME; sep = ",";
}
Console.WriteLine(s);


Console.WriteLine(
    string.Join(";", ED.EMP.Select(x => x.ENAME))
);
```

6. Megjelenítés GUI-n

```
{
    dataGridView1.Columns.Clear();
    dataGridView1.Rows.Clear();
    dataGridView1.ItemsSource = null;
    var dolgozok = from dolgozo in NE.EMP
                   orderby dolgozo.ENAME select dolgozo;
    dataGridView1.ItemsSource = dolgozok.ToList();
}
```

- Így bindelhető DataContexten keresztül is
- ItemsSource csak dolgozok.ToList() lehet, vagy teljes tábla esetén ED.EMP.Load() után ED.EMP.Local

+1 JOIN

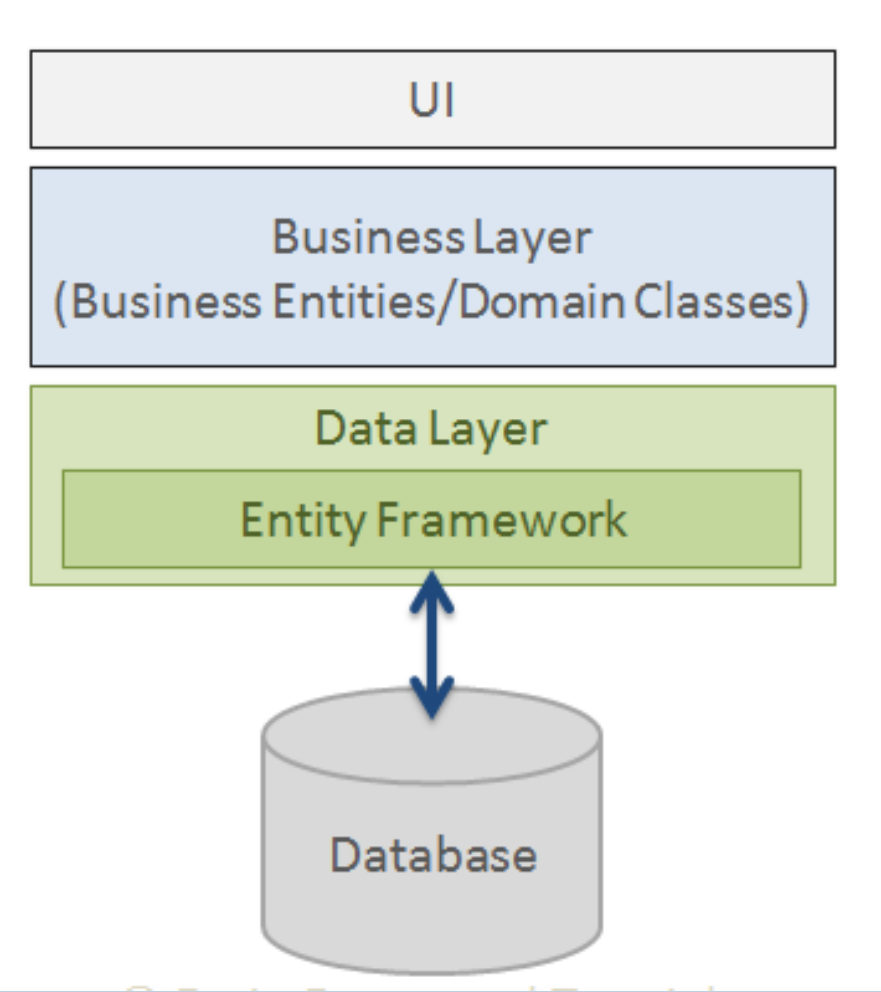
```
var dolgozok = from dolgozo in NE.EMP
   join reszleg in NE.DEPT
   on dolgozo.DEPTNO equals reszleg.DEPTNO
   select new {
       dolgozo.ENAME, dolgozo.SAL, reszleg.DNAME
   };
```

VAGY Lazy Loading kihasználásával:

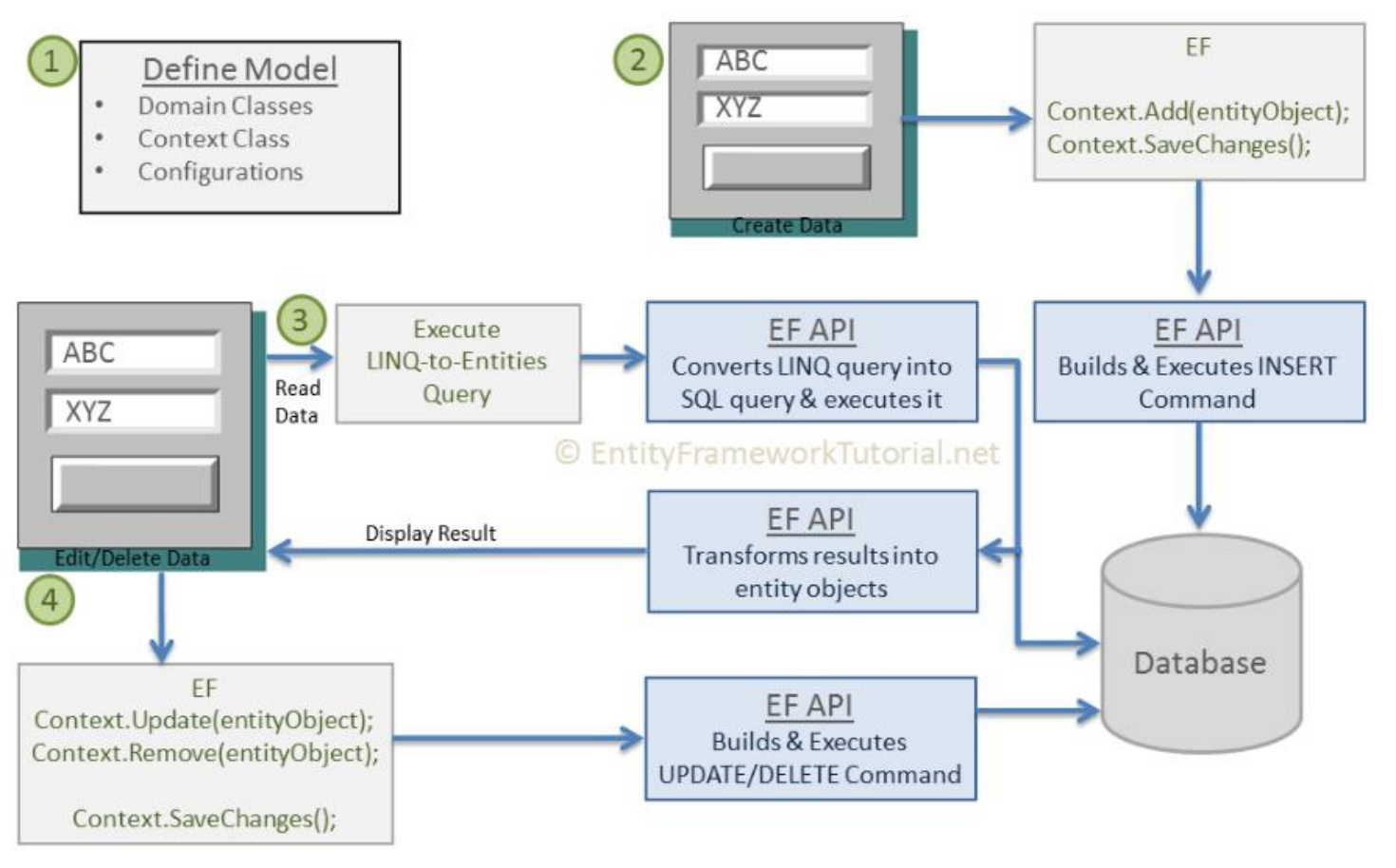
```
var dolgozok = from dolgozo in NE.EMP
   select new
   {
       dolgozo.ENAME,dolgozo.SAL,dolgozo.DEPT.DNAME
   };
```

ORM használatának előnyei

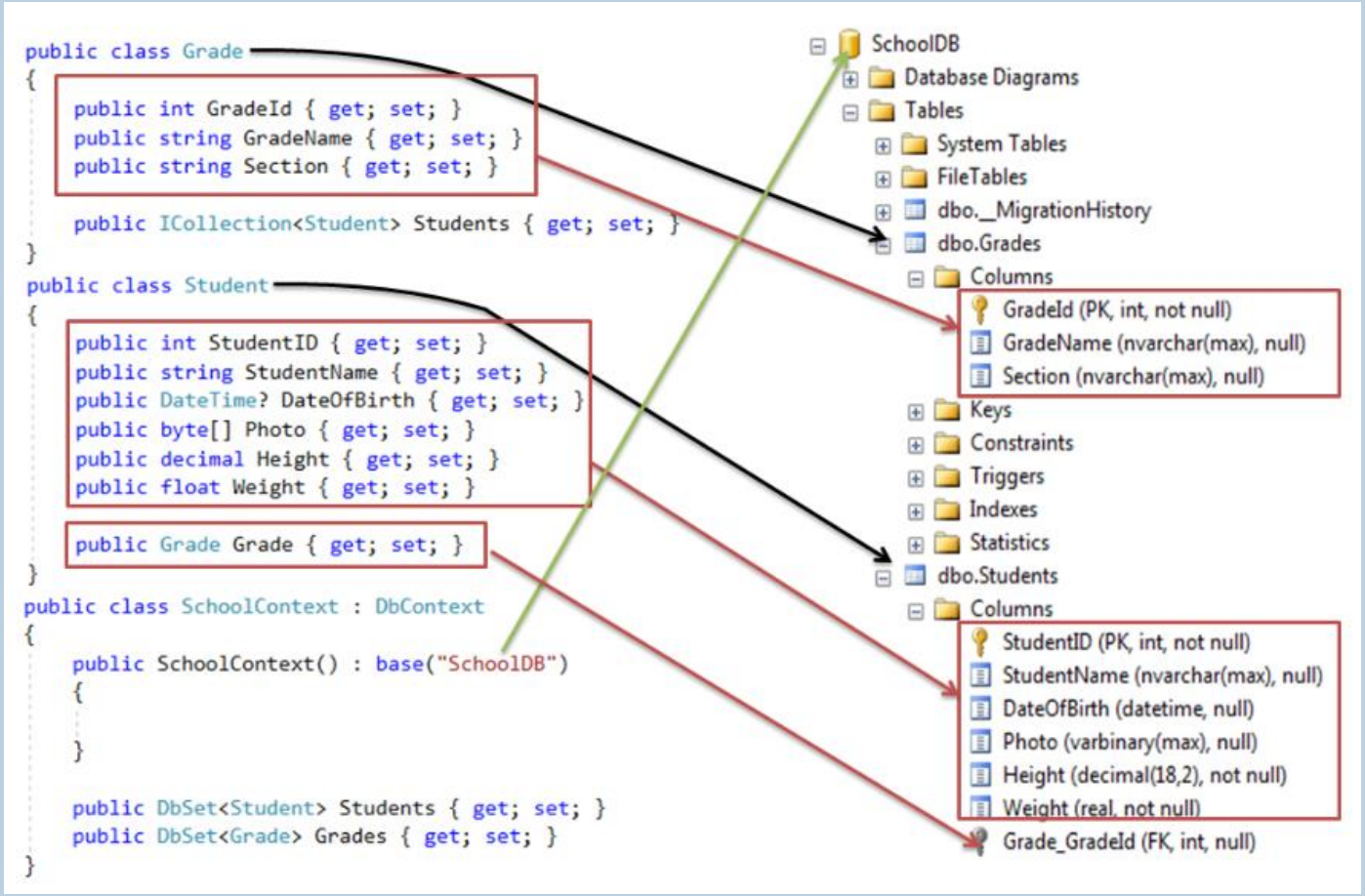
- A kódban sehol sem használunk dialektus függő SQL utasításokat
  - Bármikor dialektust/szervert válthatunk a kód átírása nélkül
- A string formátumú SQL utasítások helyett a fordítás közben szintaktikailag ellenőrzött lekérdező formátumot használunk
  - A fordítás közben kiderül, ha a bárhol szintaktikai hiba van
- A string formátumú SQL paraméterek (string összefűzés) helyett változókat használunk lekérdezés paraméterként
  - SQL injection elkerülése
- A lekérdezések eredménye nem általános anonymous típus / objektum / asszociatív tömb
  - Helyette erősen típusos ismert típusú érték / példány / lista
- Az ORM rétegre +1 réteg elhelyezésével könnyedén megoldható az adatforrás tetszőleges cseréje és a kód tesztelése
  - Repository Pattern, Dependency Injection Pattern


UI
Business Layer
(Business Entities/Domain Classes)
Data Layer
Entity Framework
Database

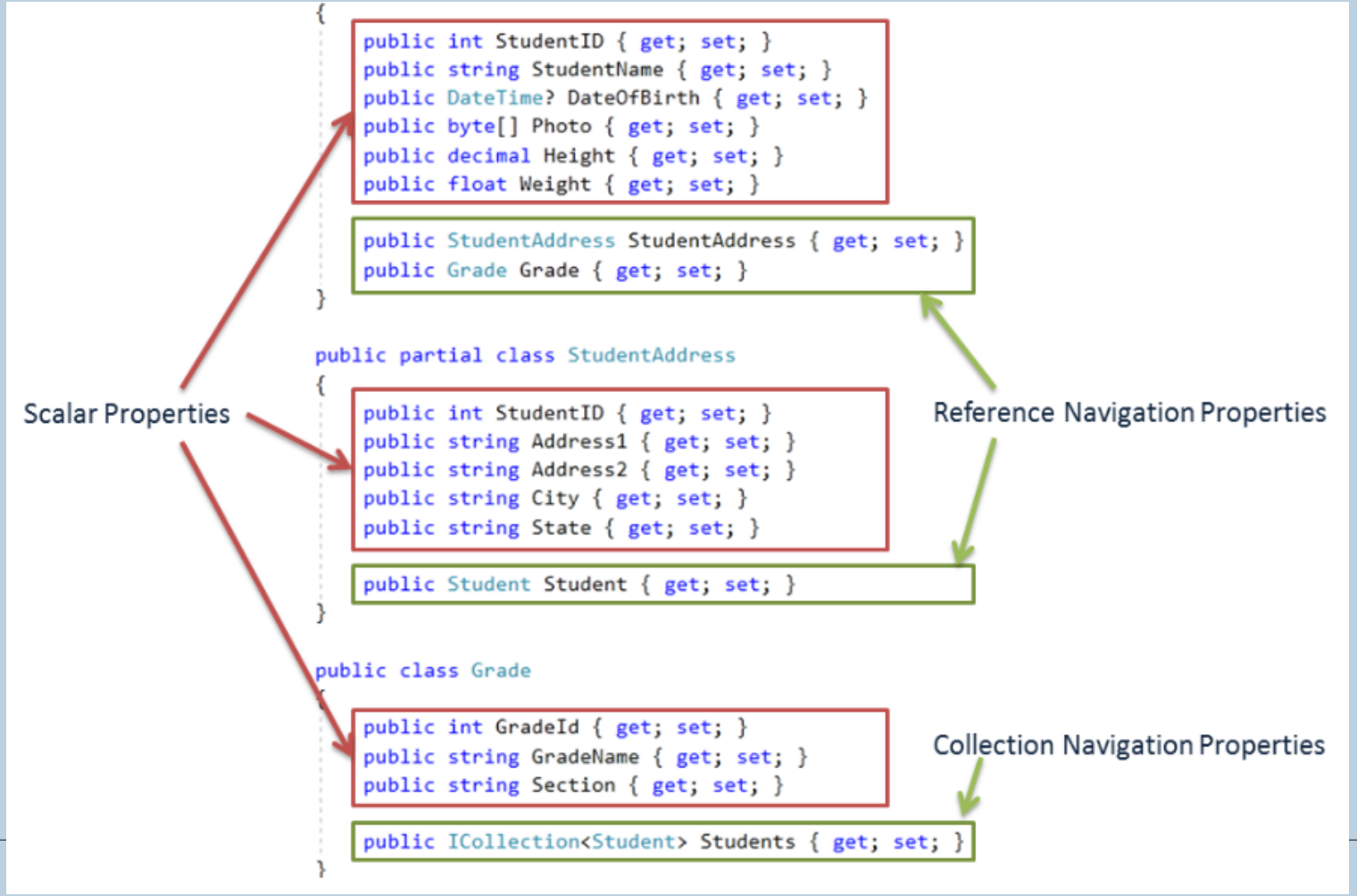
EF munkamenet



Az adatbázis létrehozásának névkonvenciói



Táblák közötti kapcsolatok a navigációs tulajdonságokkal



EF Névkonvenciók

Konvenció	Leírás
Schema	Összes objektumból egy-egy táblát generál.
Table Name	<Entity Class Name> + 's'
Elsődleges kulcs neve	1) Id 2) <Entity Class Name> + "Id" (case insensitive)
Idegen kulcs neve	1. EF megkeresi az elsődleges kulccsal megegyező nevet a kapcsoló osztályban. 2. Ha nincs, akkor EF létrehoz egy FK oszlopot az adattáblában a <Dependent Navigation Property Name> + "_" + <Principal Entity Primary Key Property Name> felhasználásával
Null column	Null lehet az összes referencia típusú változónak és a nullázható egyszerű típus
Not Null Column	Az elsődleges kulcsnak és a nemnullázható egyszerű típusoknak pl. int, float, decimal, datetime
DB oszlopok sorrendje	Osztály adatainak a sorrendjében
Adatok kapcsolása az adatbázishoz	Alapértelmezetten az összes. Kivételképzés: [NotMapped] attributummal
Cascade delete	Alapértelmezett

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!