

EF Core Code First

Entity Framework DB From Code



SoftUni Team
Technical Trainers



SoftUni
Foundation



Software University

<http://softuni.bg>

Table of Contents

1. Code First model
2. EF Core Components
3. EF Core Configuration
4. Database Migrations



sli.do

#csharp-db



Code First Model

Motivation for Code First approach

What is the Code First Model?

- **Code First** means to write the .NET classes and let EF Core create the **database** from the **mappings**



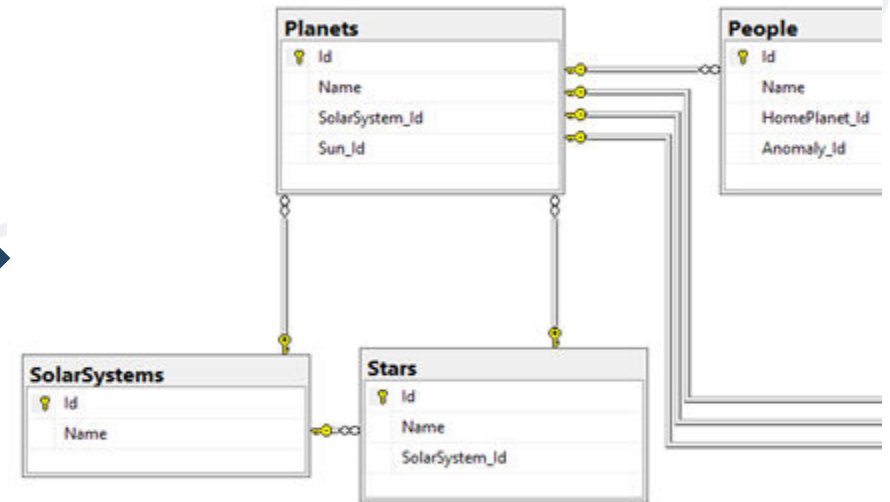
```

C# Planet.cs
└─ Planet
    └─ Planet()
        Id : int
        Name : string
        Sun : Star
        SolarSystem : SolarSystem
        OriginAnomalies : ICollection<Anomaly>

C# SolarSystem.cs
└─ SolarSystem
    Id : int
    Name : string

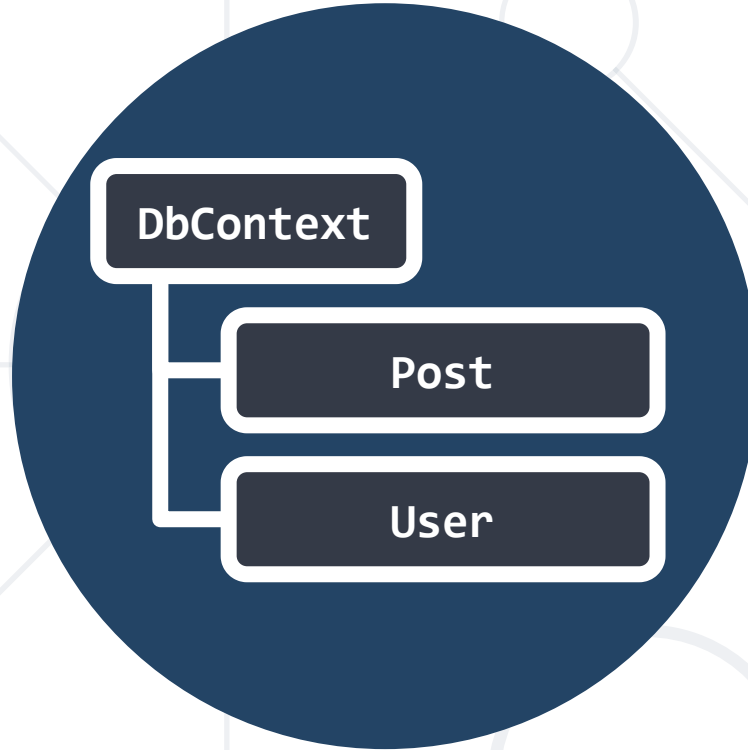
C# Star.cs
└─ Star
    Id : int
    Name : string
    SolarSystem : SolarSystem

```



Why Use Code First?

- Write code **without** having to define **mappings** in XML or **create** database **tables**
- Define objects in **C# format**
- Enables database persistence with no configuration
- Changes to code can be **reflected** (migrated) in the schema
- **Data Annotations** or **Fluent API** describe properties
 - Key, Required, MinLength, etc.



EF Core Components

Overview of system objects

- Bunch of normal C# classes (POCO)
 - May contain **navigation properties** for **table relationships**

```
public class PostAnswer
{
    public int Id { get; set; }
    public string Content { get; set; }
    public int PostId { get; set; }
    public Post Post { get; set; }
}
```

Primary key

Foreign key

Navigation property

- Recommended to be in a **separate class library**

Domain Classes (Models) (2)

- Another example of a domain class (model)

```
public class Post
{
    public int Id { get; set; }
    public string Content { get; set; }
    public int AuthorId { get; set; }
    public User Author { get; set; }

    public IList<Reply> Replies { get; set; }
}
```

Navigation
property

One-to-Many
relation

- Maps a **collection** of **entities** from a **table**
- Set operations: **Add**, **Attach**, **Remove**, **Find**
- **DbContext** contains multiple **DbSet<T>** properties

```
public class DbSet<TEntity> :  
System.Data.Entity.Infrastructure.DbQuery<TEntity>  
    where TEntity : class  
    Member of System.Data.Entity
```

```
public DbSet<Post> Posts { get; set; }
```

- Usually named after the database e.g. **BlogDbContext**, **ForumDbContext**
- Inherits from **DbContext**
- Manages model classes using **DbSet<T>** type
- Implements **identity tracking, change tracking**
- Provides **API** for **CRUD** operations and **LINQ-based** data access
- Recommended to be in a separate class library
 - Don't forget to reference the EF Core library + any providers
- Use several **DbContext** if you have too much models

Defining DbContext Class – Example

EF Reference

```
using Microsoft.EntityFrameworkCore;  
using CodeFirst.Data.Models;
```

Models Namespace

```
public class ForumDbContext : DbContext  
{  
    public DbSet<Category> Categories { get; set; }  
    public DbSet<Post> Posts { get; set; }  
    public DbSet<PostAnswer> PostAnswers { get; set; }  
    public DbSet<User> Users { get; set; }  
}
```

CRUD Operations with EF Code First

```
var db = new ForumDbContext();
var category = new Category { Name = "Database course" };
db.Categories.Add(category);

var post = new Post();
post.Title = "Homework Deadline";
post.Content = "Please extend the homework deadline";
post.Type = PostType.Normal;
post.Category = category;
post.Tags.Add(new Tag { Text = "homework" });
post.Tags.Add(new Tag { Text = "deadline" });

db.Posts.Add(post);

db.SaveChanges();
```



EF Core Configuration

NuGet Packages, Configuration

Code First with EF Core: Setup

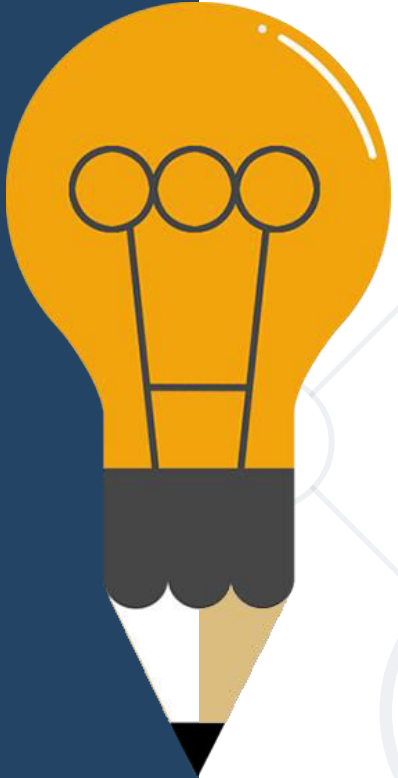
- To add EF Core support to a project in Visual Studio:

- Install it from **Package Manager Console**

```
Install-Package Microsoft.EntityFrameworkCore
```

- EF Core is modular – any **data providers** must be installed too

```
Install-Package Microsoft.EntityFrameworkCore.SqlServer
```



How to Connect to SQL Server?

- One way to connect is to create a **Configuration** class with your connection string:

```
public static class Configuration
{
    public const string ConnectionString = "Server=.;Database=...;";
}
```

- Then add the connection string in the **OnConfiguring** method in the **DbContext** class

```
protected override void OnConfiguring(DbContextOptionsBuilder builder)
{
    if (!builder.IsConfigured)
        builder.UseSqlServer(Configuration.ConnectionString);
}
```

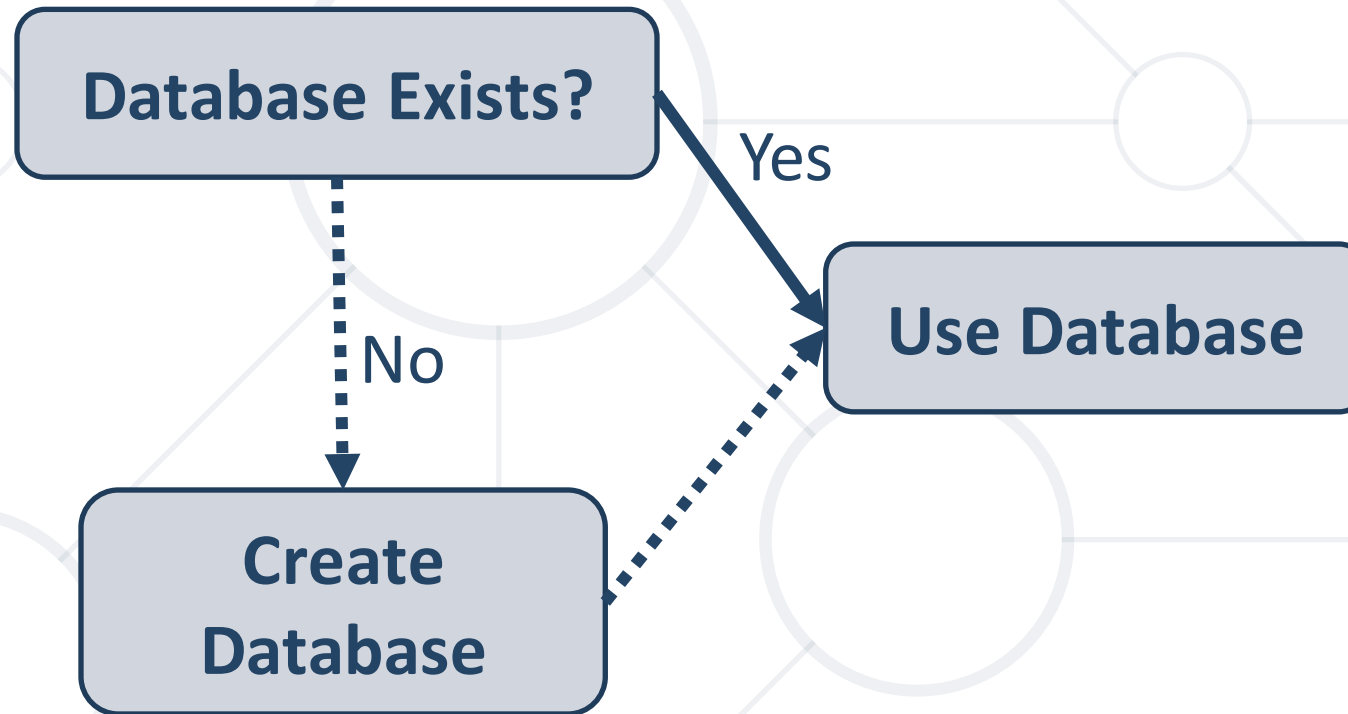

- The **OnModelCreating** Method let us use the Fluent API to describe our **table relations** to **EF Core**

```
protected override void OnModelCreating(ModelBuilder builder)
{
    builder.Entity<Category>()
        .HasMany(c => c.Posts)
        .WithOne(p => p.Category);

    builder.Entity<Post>()
        .HasMany(p => p.Replies)
        .WithOne(r => r.Post);

    builder.Entity<User>()
        .HasMany(u => u.Posts)
        .WithOne(p => p.Author);
}
```


Database Connection Workflow





Database Migrations

What are Database Migrations?

- 
- Updating database schema **without losing data**
 - Adding/dropping tables, columns, etc.
 - Migrations in EF Core keep their **history**
 - Entity Classes, DB Context versions are all **preserved**
 - **Automatically** generated



```
└─ Migrations
   ├── 20171102161155_Initial.cs
   ├── 20171102161155_Initial.Designer.cs
   └── ForumDbContextModelSnapshot.cs
```

- To use migrations in EF Core, we use the Add-Migration command from the **Package Manager Console**

```
Add-Migration {MigrationName}
```

- To undo a migration, we use **Remove-Migration**

```
Remove-Migration
```

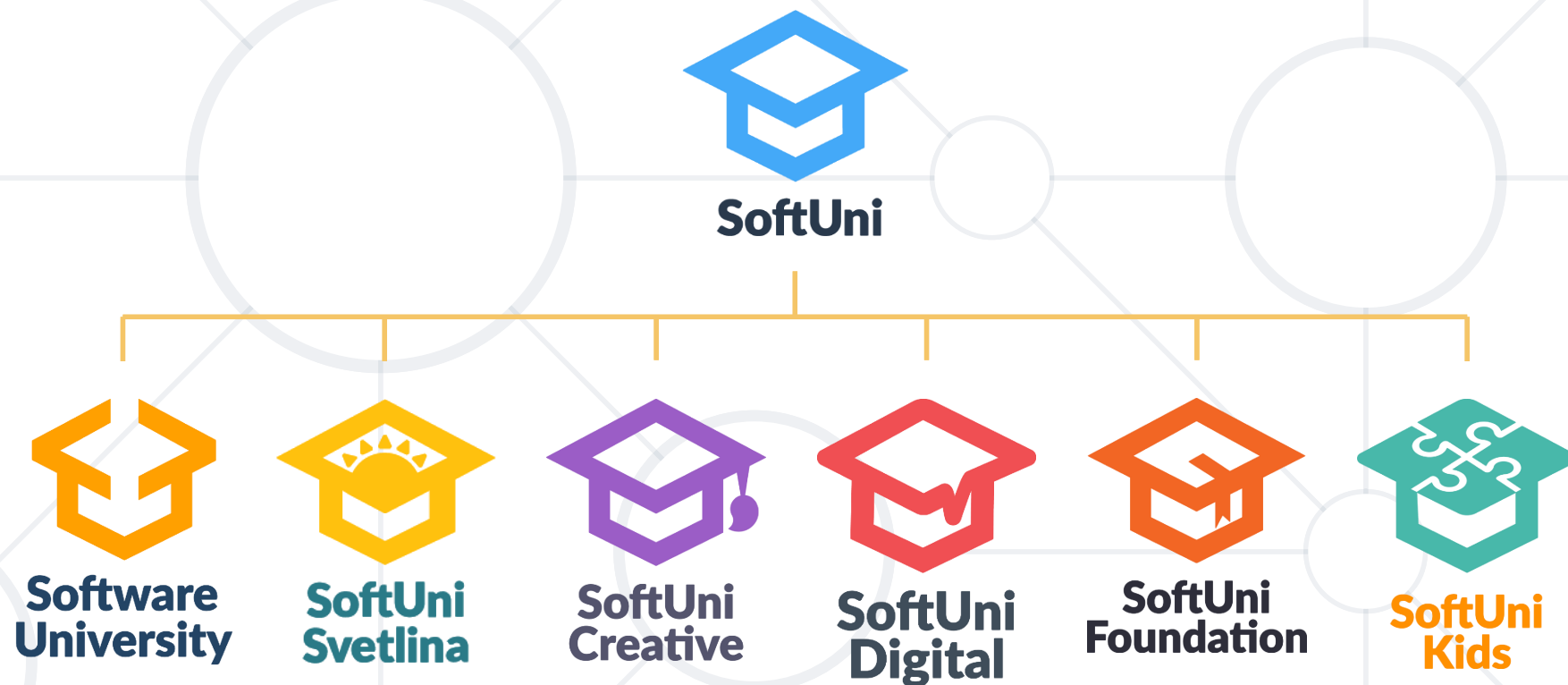
- Commit changes to the database, using **Update-Database**

```
Update-Database
```

- Code First **increases productivity** by centralizing maintenance
- **Classes** represent real world objects with their **properties** and **behaviour**
- Entity Framework Core uses **data classes** (POCOs) to represent DB objects
- We can use **Database Migrations** to update our database without losing our data



Questions?



SoftUni Diamond Partners



XSsoftware



SBTech
we know sports



telenor



SoftwareGroup
doing it right

NETPEAK



SmartIT



Postbank

Решения за твоето утре



INDEAVR

Serving the high achievers



INFRAGISTICS®



STEMO®
Computer Systems & Software

SUPERHOSTING.BG

SoftUni Organizational Partners



OneBit
SOFTWARE



WORLD
OF
MYTHS

- This course (slides, examples, demos, videos, homework, etc.) is licensed under the "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International" license



Trainings @ Software University (SoftUni)

- Software University – High-Quality Education and Employment Opportunities
 - softuni.bg
- Software University Foundation
 - <http://softuni.foundation/>
- Software University @ Facebook
 - facebook.com/SoftwareUniversity
- Software University Forums
 - forum.softuni.bg

