Entity Framework

EF Core Code First

Lesson Objectives

- EF Code First Overview
- Code-First Conventions
- Database Initialization
- Database Initialization Strategies
- Seed data

Section 1

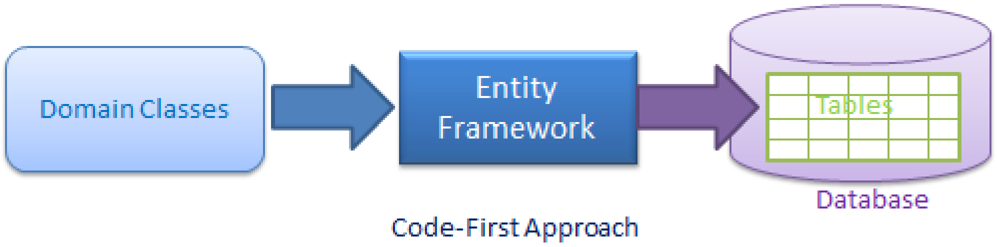
EF CODE FIRST OVERVIEW

EF Code First Overview

- Do not have an existing database for your application?
- Do not have much experience in SQL?
- Do not want to working with SQL directly?
- Prefer to writing entities and context class first?

EF Code First Overview

- Code-First is mainly useful in Domain Driven Design.
- In the Code-First approach,
  - ✓ developer focus on the domain of application
  - ✓ create classes for domain entity
  - ✓ EF generate database from code



Code-First Workflow

- Step 1: Create or modify domain classes
- Step 2: Configure these domain classes using Fluent-API or data annotation attributes
- Step 3: Create database using automated migration

Code-First Workflow

- Loop:
  - ✓ Step 2.x: Add or update the domain classes
  - ✓ Step 3.x: Update database schema using code first migration
- End loop

Section 2

CODE-FIRST CONVENTIONS

https://www.entityframeworktutorial.net/code-first/code-first-conventions.aspx

Default code first conventions

Default Convention For	Description
Schema	By default, EF creates all the DB objects into the **dbo** schema.
Table Name	<Entity Class Name> + 's' EF will create a DB table with the entity class name suffixed by 's' e.g. Student domain class (entity) would map to the Students table.
Primary key Name	1) Id 2) <Entity Class Name> + "Id" (case insensitive) EF will create a primary key column for the property named Id or <Entity Class Name> + "Id" (case insensitive).
Foreign key property Name	By default EF will look for the foreign key property with the same name as the principal entity primary key name. If the foreign key property does not exist, then EF will create an FK column in the Db table with <Dependent Navigation Property Name> + "_" + <Principal Entity Primary Key Property Name> e.g. EF will create Grade_GradeId foreign key column in the Students table if the Student entity does not contain foreignkey property for Grade.
Null column	EF creates a null column for all reference type properties and nullable primitive properties e.g. string, Nullable<int>, Student, Grade (all class type properties)
Not Null Column	EF creates NotNull columns for Primary Key properties and non-nullable value type properties e.g. int, float, decimal, datetime etc.
DB Columns order	EF will create DB columns in the same order like the properties in an entity class. However, primary key columns would be moved first.
Properties mapping to DB	By default, all properties will map to the database. Use the [NotMapped] attribute to exclude property or class from DB mapping.
Cascade delete	Enabled by default for all types of relationships.

Default code first conventions

- Examples:

✓ public string Description { get; set; }	=> mapped
✓ public string Description { get { return Name; } }	=> **not** mapped
✓ public string Description { set { value = Name; } }	=> **not** mapped
✓ public string Description;	=> **not** mapped
✓ private string Description { get; set; }	=> **not** mapped
✓ protected string Description { get; set; }	=> **not** mapped

Data type mapped

C# Data Type	Mapping to SQL Server Data Type	C# Data Type	Mapping to SQL Server Data Type
int	int	byte	tinyint
string	nvarchar(Max)	short	smallint
decimal	decimal(18,2)	long	bigint
float	real	double	float
byte[]	varbinary(Max)	char	*No mapping*
datetime	datetime	sbyte	*No mapping (throws exception)*
bool	bit	object	*No mapping*

Relationship Convention

- EF core infers the One-to-Many, One-to-One relationship using the navigation property by default convention.
- You need to configure them either using Fluent API or DataAnnotation.

➢ There are no default conventions available in Entity Framework Core which automatically configure a many-to-many relationship. You must configure it using Fluent API.

Conventions for One-to-Many Relationships

- Convention 1:
  - ✓ includes a reference navigation property of parent in the child entity class
- Convention 2:
  - ✓ includes a collection navigation property of children in the parent entity class
- Convention 3:
  - ✓ includes navigation properties at both ends will also result in a one-to-many relationship
- Convention 4:
  - ✓ a fully defined relationship at both ends will create a one-to-many relationship

Conventions for One-to-Many Relationships

```
public class Student
{
    public int Id { get; set; }
    public string Name { get; set; }

    public int GradeId { get; set; }     [4]
    public Grade Grade { get; set; }    [1, 3, 4]
}

public class Grade
{

    public int GradeId { get; set; }
    public string GradeName { get; set; }

    public ICollection<Student> Student { get; set; } [2, 3, 4]
}
```

Section 3

DATABASE INITIALIZATION

Database Initialization

The DBContext connects to the database using the Database Providers. These Providers requires a connection string to connect to the database.

Database Initialization

Once you install the database provider, then you can configure the database provider using the extension method provided them using the DbContextOptionsBuilder

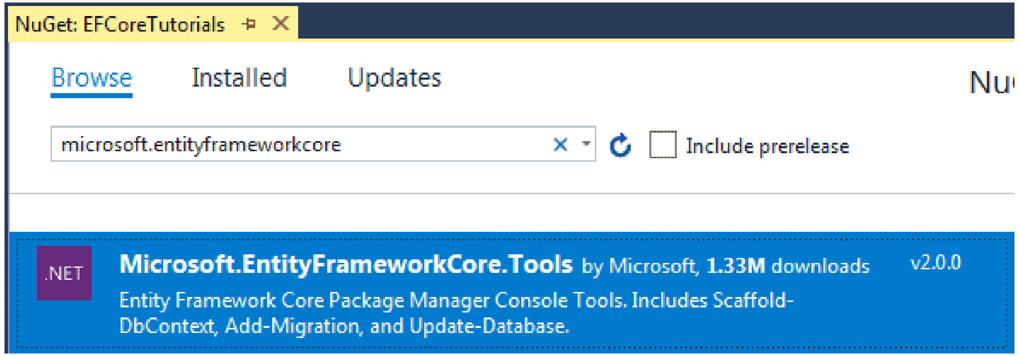
For Example to use SQL Server install the package

```

Install-Package Microsoft.EntityFrameworkCore.SqlServer

```

Database Initialization

In order to execute EF Core commands from Package Manager Console, search for the

Microsoft.EntityFrameworkCore.Tools package from NuGet UI and install it as shown below



microsoft.entityframeworkcore

Microsoft.EntityFrameworkCore.Tools by Microsoft, **1.33M** downloads v2.0.0

Entity Framework Core Package Manager Console Tools. Includes Scaffold-DbContext, Add-Migration, and Update-Database.

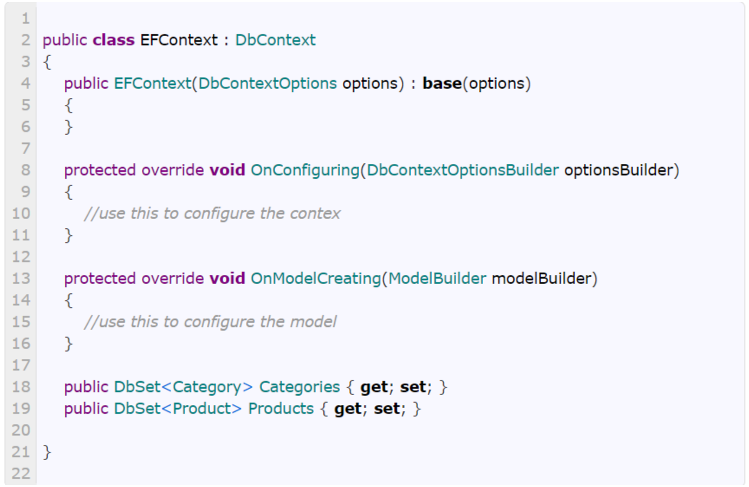
Section 4

DATABASE INITIALIZATION STRATEGIES

Create Class DBContext

To use DBContext, we need to create a context class and derive it from the DbContext base class

The following is the example of the Context class (EFContext):

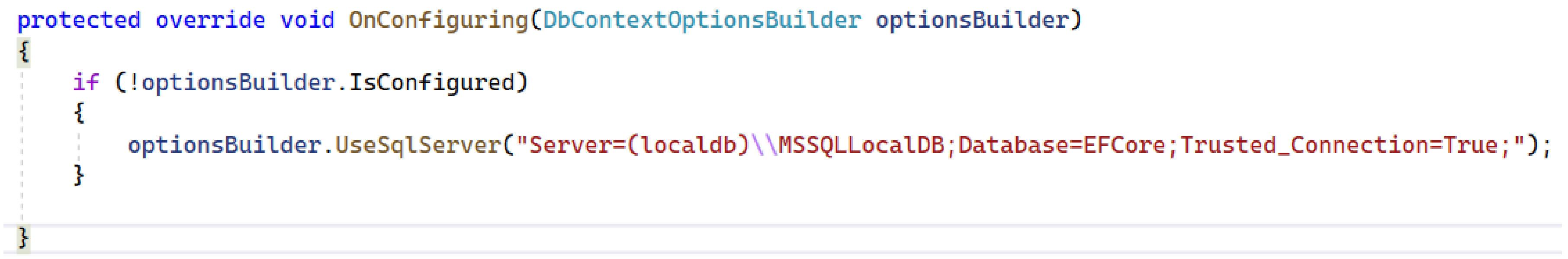


```
public class EFContext : DbContext
{
    public EFContext(DbContextOptions options) : base(options)
    {
    }

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        //use this to configure the contex
    }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        //use this to configure the model
    }

    public DbSet<Category> Categories { get; set; }
    public DbSet<Product> Products { get; set; }

}
```

Adding a database provider

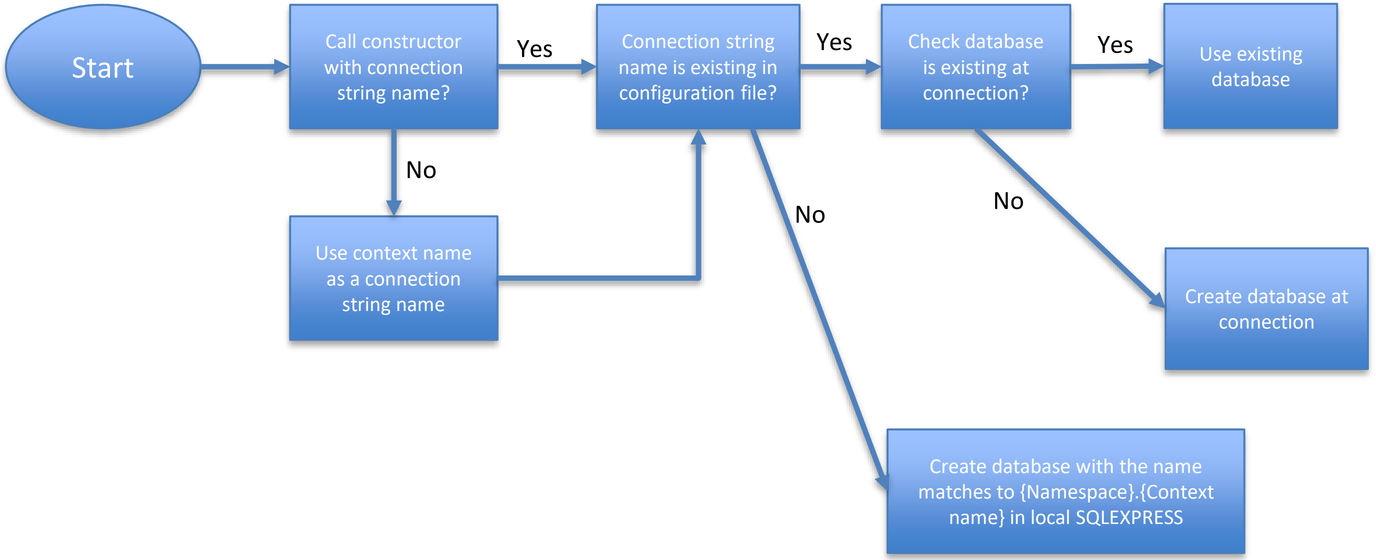
Then use the UseSqlServer extension method to register the SQL Server database provider

By overriding the OnConfigure method



```
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
{
    if (!optionsBuilder.IsConfigured)
    {
        optionsBuilder.UseSqlServer("Server=(localdb)\\MSSQLLocalDB;Database=EFCore;Trusted_Connection=True;");
    }

}
```

Database Initialization


Start
Call constructor with connection string name?
Yes
Connection string name is existing in configuration file?
Yes
Check database is existing at connection?
Yes
Use existing database
No
Use context name as a connection string name
No
No
Create database at connection
Create database with the name matches to {Namespace}.{Context name} in local SQLEXPRESS

Section 5

SEED DATA

Seed Data

- To insert data into database tables during the database initialization process.
- Use to provide some test data for the application
- Use to create some default master data for the application

Seed Data steps

- Step 1: Override the OnModelCreating method

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
  {
    //Configure domain classes using modelBuilder here
  }
```

- Step 2: Reference to the EntityTypeBuilder
  - ✓ modelBuilder.Entity<T>()
- Step 3: Use the HasData method to supply the list
  - ✓ modelBuilder.Entity<T>()

```
             .HasData( new T())
```

Once you have added the initial data, you should use migrations to update the database

Summary

- Use EF Code First approach: create entities class and DbContext first, then generate database
- Code-First Conventions: EF uses conventions to create database, tables, columns
- There are strategies to create database
- Always consider to use appropriate strategy

Thank you