Session 3

Working with ADO.NET and Entity Framework

Session Overview

- Describe ADO.NET
- Explain Entity Framework
- Describe data handling in ASP.NET MVC with a code-first database

Overview of ADO.NET

1
- ADO.NET is used by developers to work with data in Web applications.

2
- It is a core component of .NET Framework

3
- It helps in establishing a connection between an application and data sources

Data Layer (1-2)

Presentation Layer

The layer in which the user interface is created. In other words, it is the area that is directly visible or accessible to the client.

Business Layer

The layer in which validations/restrictions are defined.

Data Layer

The layer in which the data is stored. In simple terms, it is the storage space of the application.

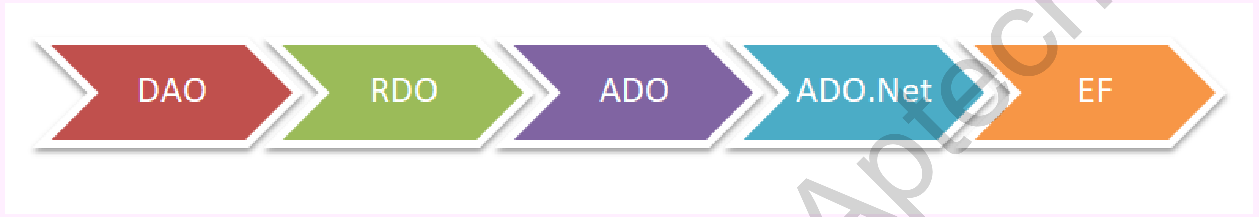
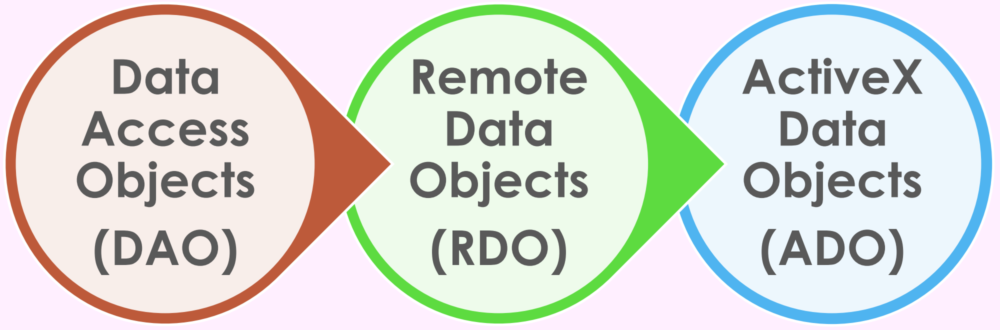


Figure 3.1: Microsoft Data Layers


Data Access Objects (DAO)
Remote Data Objects (RDO)
ActiveX Data Objects (ADO)

Data Layer (2-2)

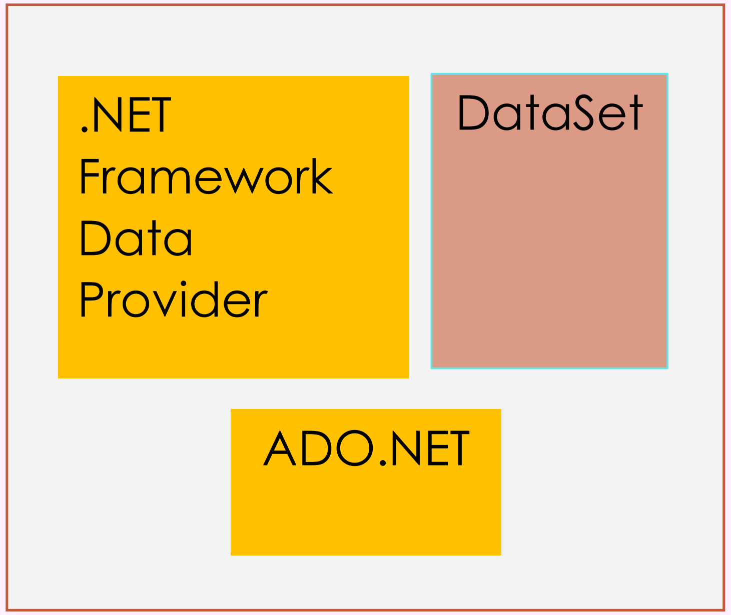


Figure 3.2: ADO.NET Components

Connection

Responsible for providing connectivity to a data source.

Command

Provides access to database commands.

DataAdapter

Acts as a bridge between the DataSet object and the data source.

DataReader

Provides a high-performance stream of data from the data source.

Entity Framework (1-2)

Entity Framework (EF) is an Object Relational Mapping (ORM) tool that is used to connect to the database.

In EF, Language Integrated Query (LINQ) is utilized to access the database and interact with auto-generated code.

EF establishes a bridge between the business entity and the data tables.

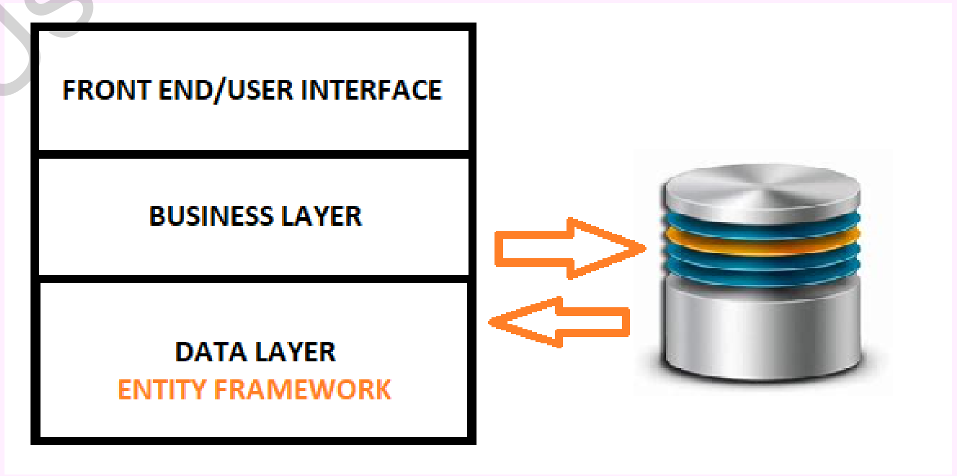
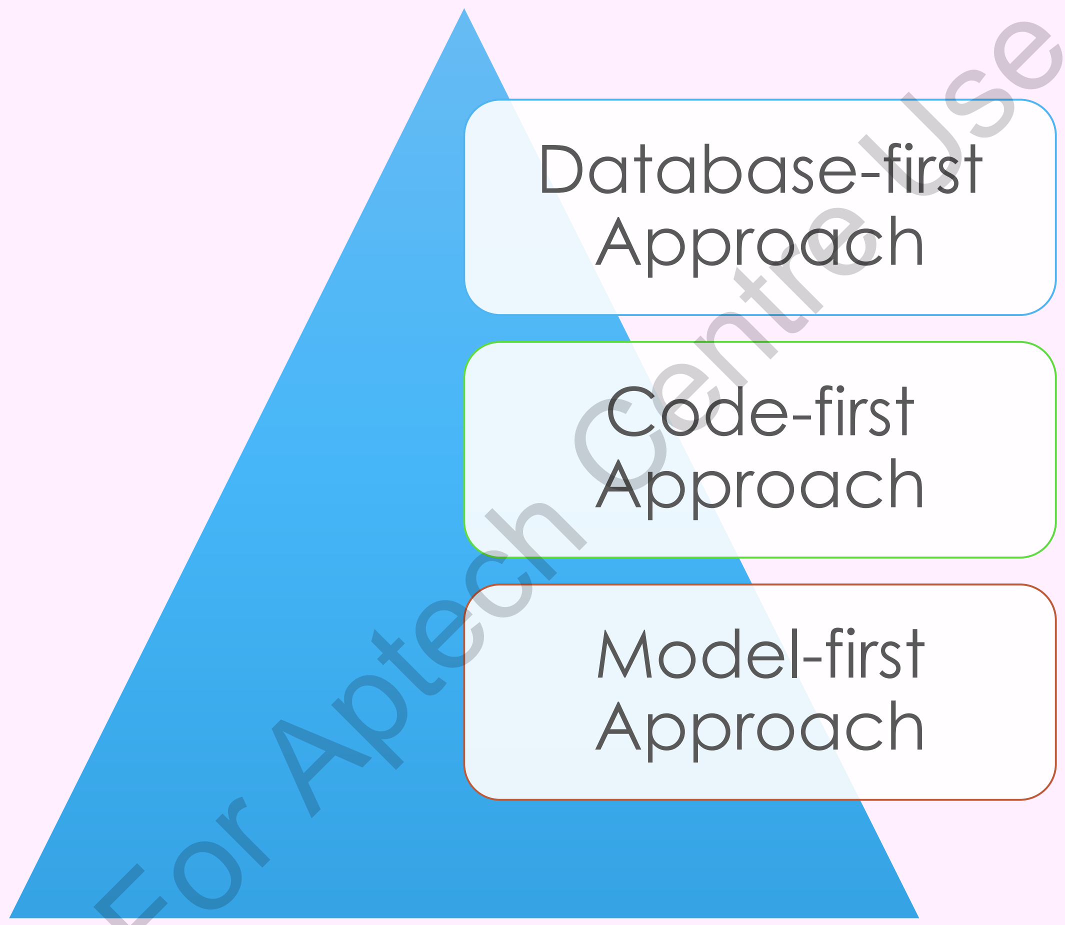


Figure 3.3: Entity Framework

Entity Framework (2-2)

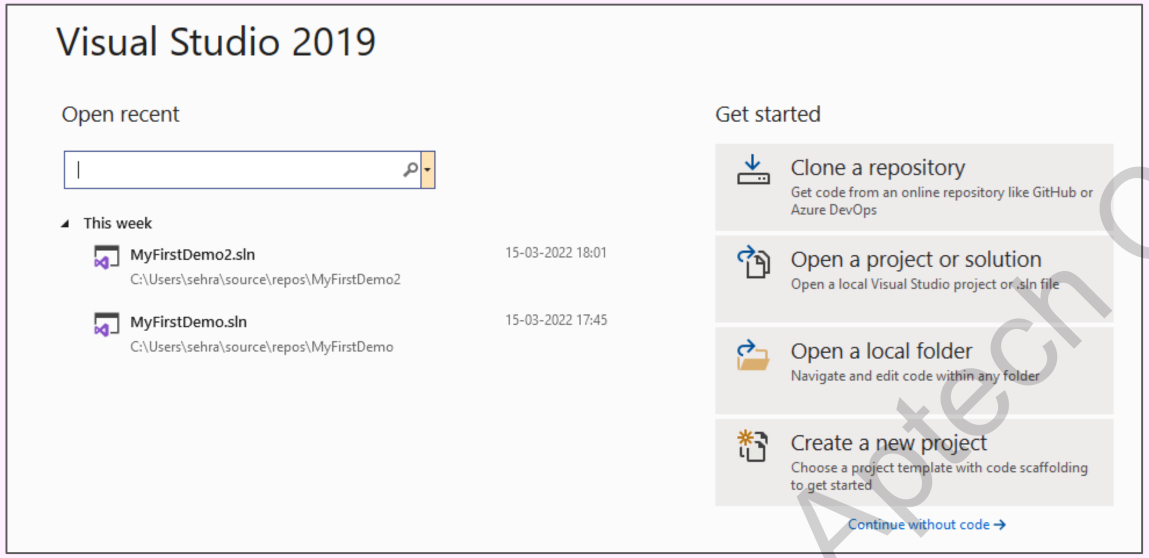
- EF uses LINQ queries instead of SQL queries and handles procedural and parameterized queries.
- EF enables caching to allow queries to be answered from the cache in case of repeat queries.
- EF allows concurrency and ensures that any changes that are being overridden are retrieved by another user.
- EF builds an Entity Data Model (EDM).
- EF does automatic transaction management while requesting or saving data.
- EF generates the required database commands for Create, Read, Update, and Delete (CRUD) operations and then, executes them.

Entity Framework Workflows



Data Handling in ASP.NET MVC with a Code-first Database(1-8)

Steps to create an application through a code-first approach in ASP.NET Core MVC:



Visual Studio 2019

Open recent

This week

MyFirstDemo2.sln — C:\Users\sehra\source\repos\MyFirstDemo2 — 15-03-2022 18:01

MyFirstDemo.sln — C:\Users\sehra\source\repos\MyFirstDemo — 15-03-2022 17:45

Get started

Clone a repository — Get code from an online repository like GitHub or Azure DevOps

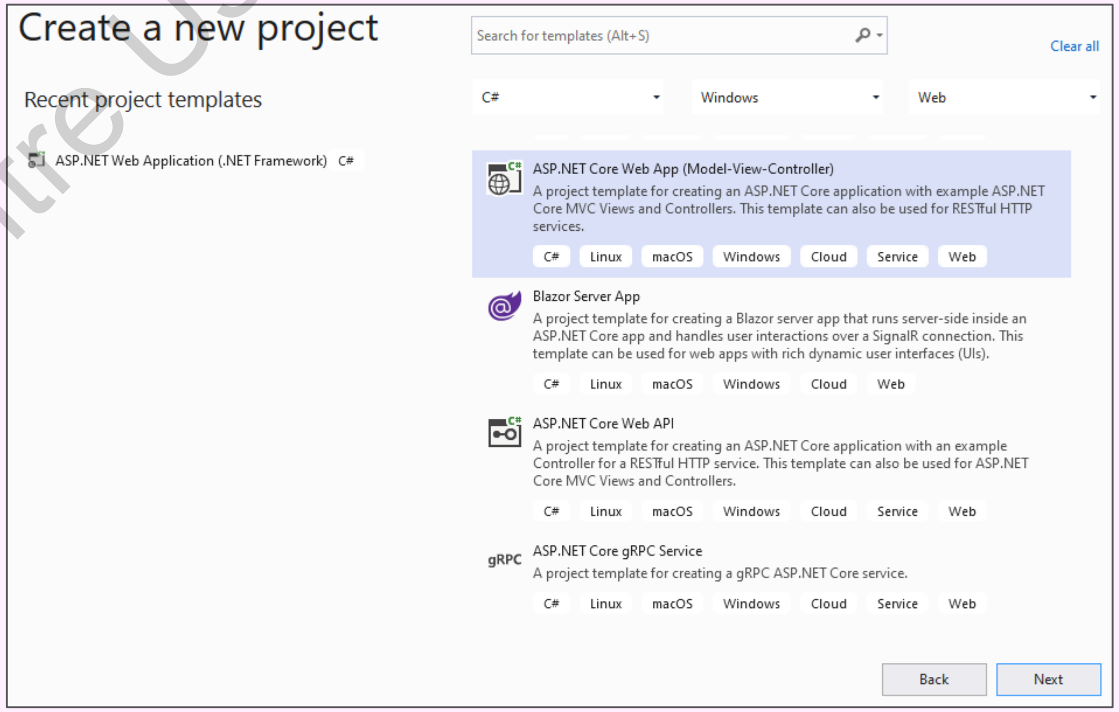
Open a project or solution — Open a local Visual Studio project or .sln file

Open a local folder — Navigate and edit code within any folder

Create a new project — Choose a project template with code scaffolding to get started

Continue without code →

Figure 3.4: Create New Project Using Visual Studio 2019



Create a new project

Search for templates (Alt+S) — Clear all

Recent project templates

ASP.NET Web Application (.NET Framework) C#

C# | Windows | Web

ASP.NET Core Web App (Model-View-Controller)
A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.
C# Linux macOS Windows Cloud Service Web

Blazor Server App
A project template for creating a Blazor server app that runs server-side inside an ASP.NET Core app and handles user interactions over a SignalR connection. This template can be used for web apps with rich dynamic user interfaces (UIs).
C# Linux macOS Windows Cloud Web

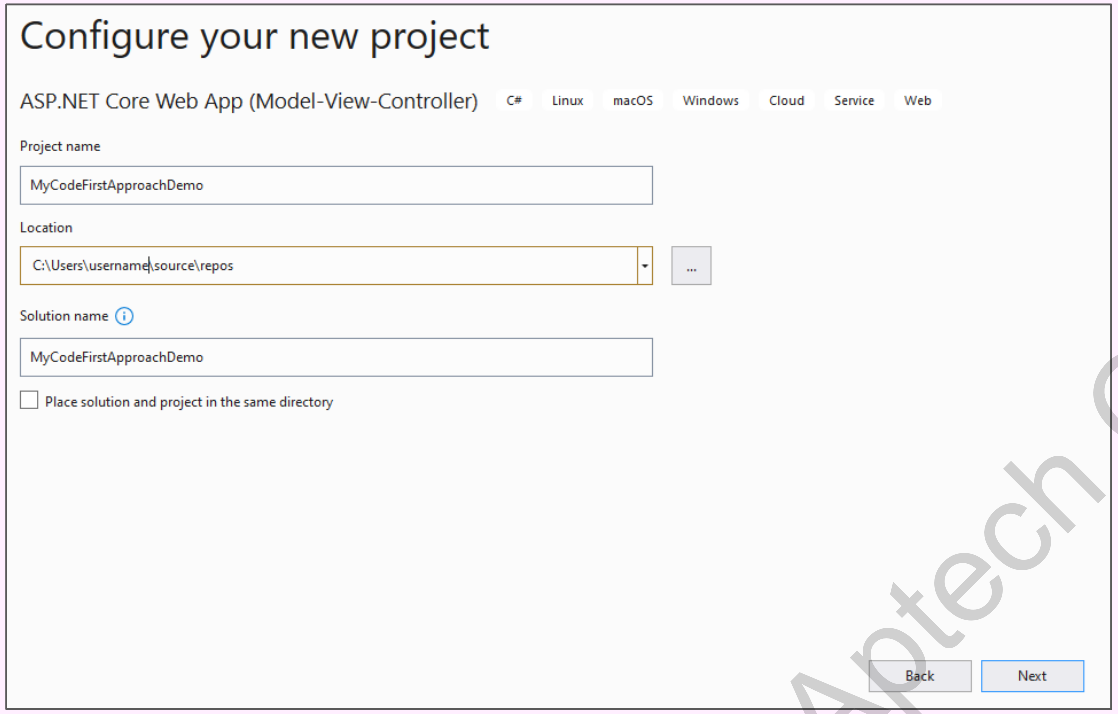
ASP.NET Core Web API
A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.
C# Linux macOS Windows Cloud Service Web

ASP.NET Core gRPC Service
A project template for creating a gRPC ASP.NET Core service.
C# Linux macOS Windows Cloud Service Web

Back | Next

Figure 3.5: ASP .NET Core Web App

Data Handling in ASP.NET MVC with a Code-first Database(2-8)



Configure your new project

ASP.NET Core Web App (Model-View-Controller)

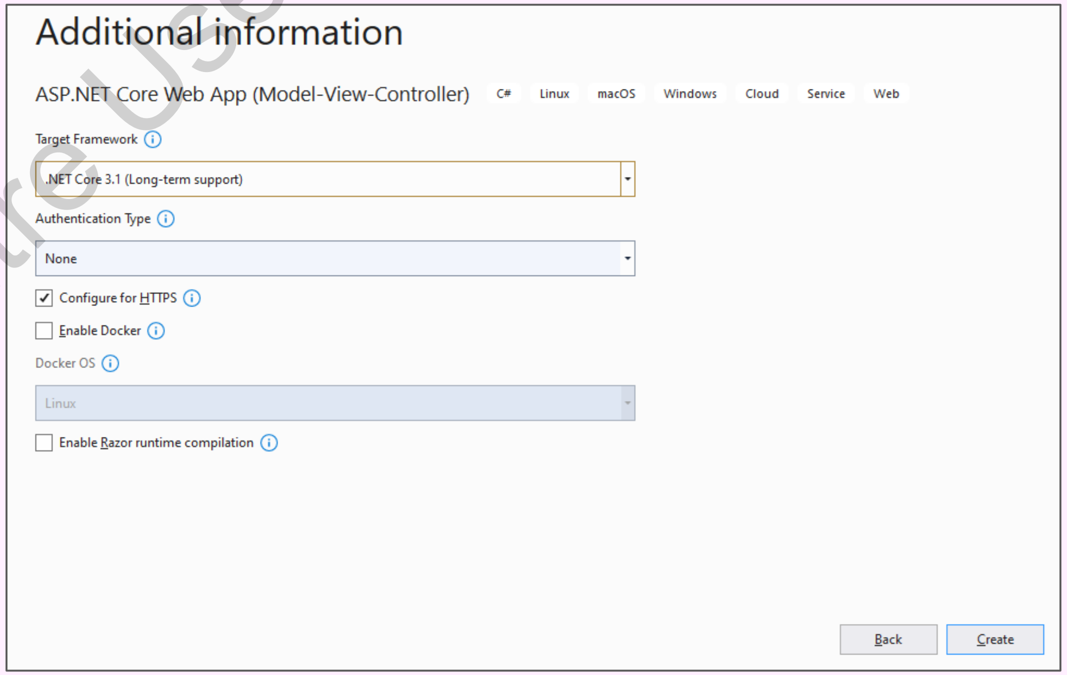
Project name: MyCodeFirstApproachDemo

Location: C:\Users\username\source\repos

Solution name: MyCodeFirstApproachDemo

☐ Place solution and project in the same directory

Figure 3.6: Configure Your New Project



Additional information

ASP.NET Core Web App (Model-View-Controller)

Target Framework: .NET Core 3.1 (Long-term support)

Authentication Type: None

☑ Configure for HTTPS

☐ Enable Docker

Docker OS: Linux

☐ Enable Razor runtime compilation

Figure 3.7: Specifying Additional Information

Data Handling in ASP.NET MVC with a Code-first Database(3-8)

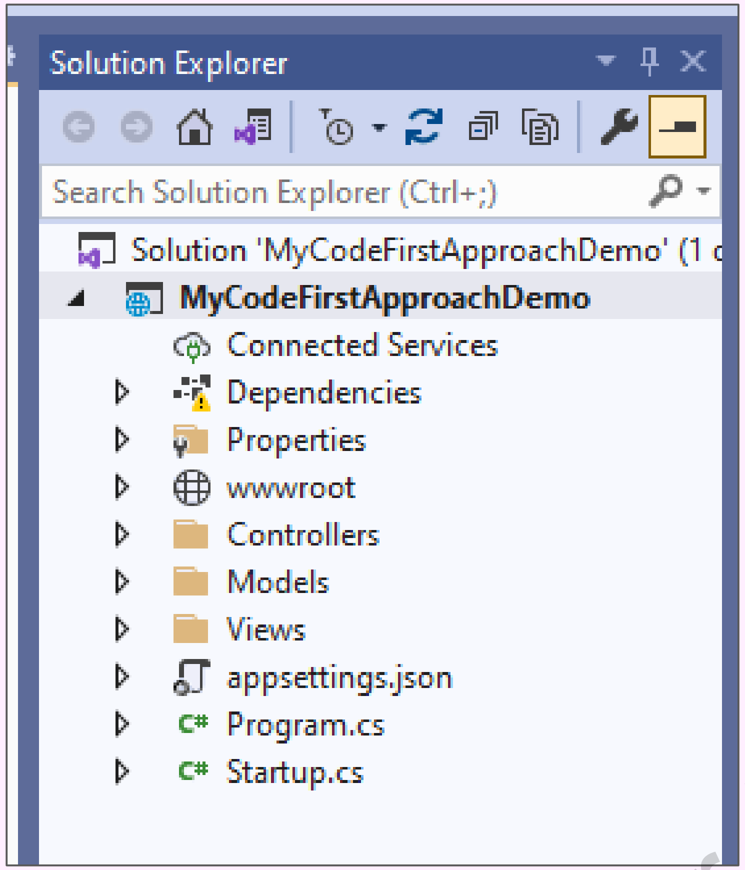
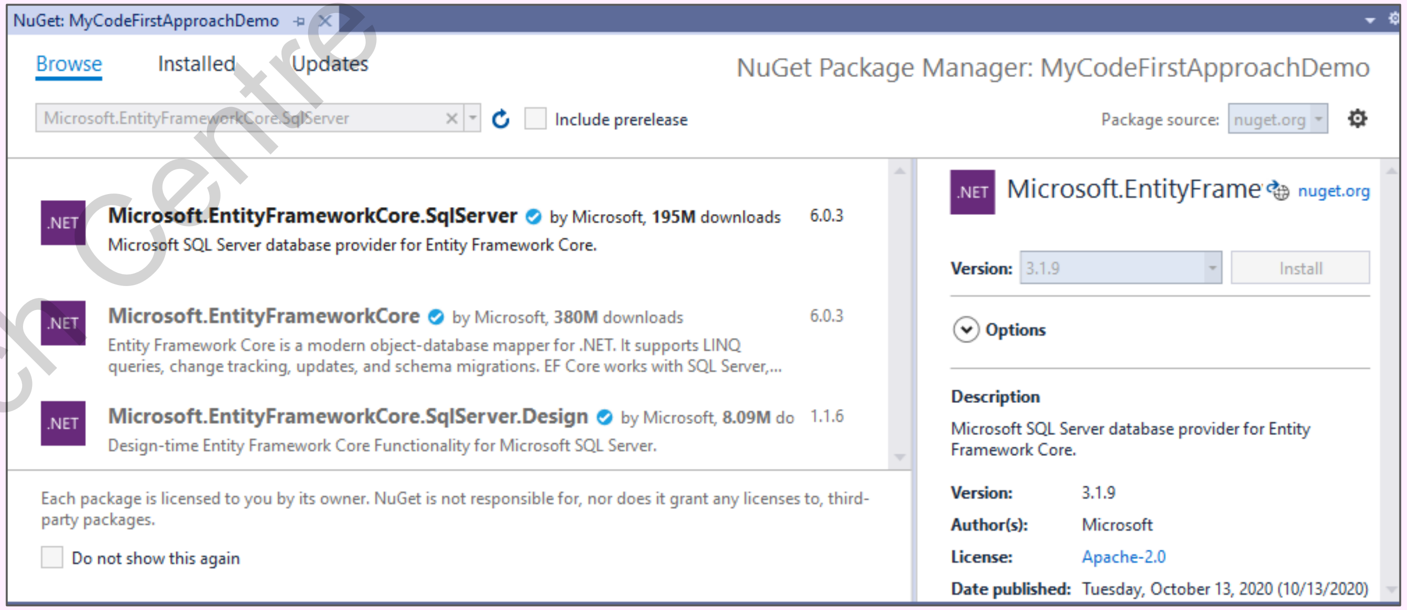


Figure 3.8: Solution Explorer Showing Dependencies Node


NuGet: MyCodeFirstApproachDemo
Browse
Installed
Updates
NuGet Package Manager: MyCodeFirstApproachDemo
Include prerelease
Package source: nuget.org
Version: 3.1.9
Install
Options

Microsoft.EntityFrameworkCore.SqlServer

Microsoft.EntityFrameworkCore.SqlServer by Microsoft, 195M downloads 6.0.3
Microsoft SQL Server database provider for Entity Framework Core.

Microsoft.EntityFrameworkCore by Microsoft, 380M downloads 6.0.3
Entity Framework Core is a modern object-database mapper for .NET. It supports LINQ queries, change tracking, updates, and schema migrations. EF Core works with SQL Server,...

Microsoft.EntityFrameworkCore.SqlServer.Design by Microsoft, 8.09M do 1.1.6
Design-time Entity Framework Core Functionality for Microsoft SQL Server.

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.

Do not show this again

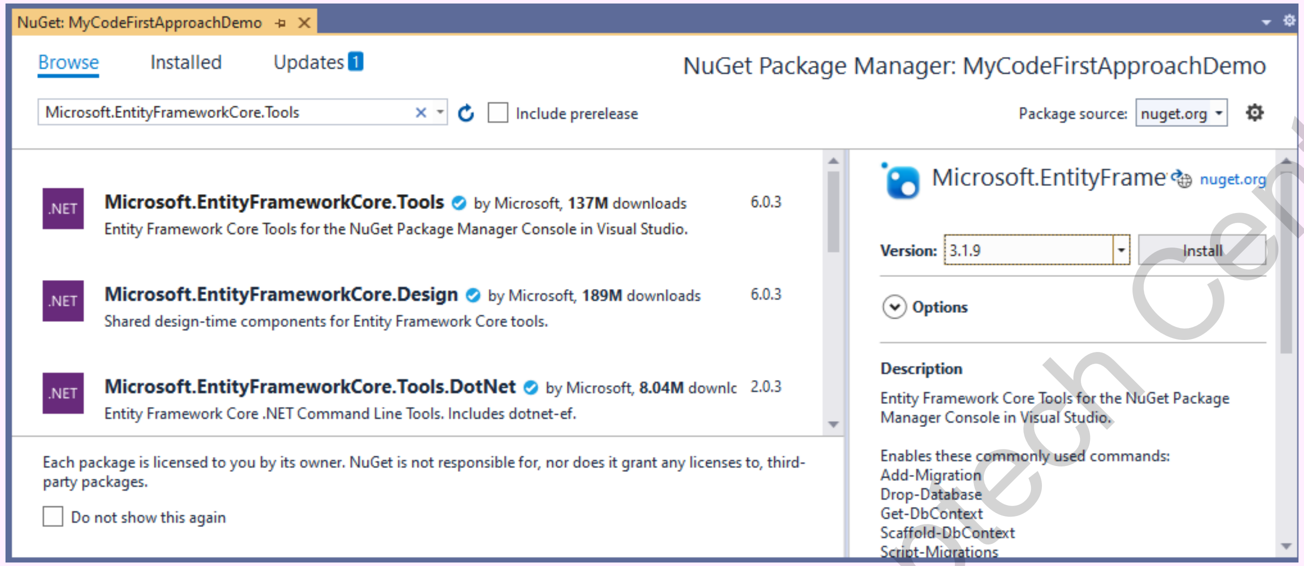
Microsoft.EntityFrame nuget.org

Description
Microsoft SQL Server database provider for Entity Framework Core.

Version:	3.1.9
Author(s):	Microsoft
License:	Apache-2.0
Date published:	Tuesday, October 13, 2020 (10/13/2020)

Figure 3.9: NuGet Package Manager

Data Handling in ASP.NET MVC with a Code-first Database(4-8)


NuGet: MyCodeFirstApproachDemo
Browse
Installed
Updates 1
NuGet Package Manager: MyCodeFirstApproachDemo
Include prerelease
Package source: nuget.org
Microsoft.EntityFrame nuget.org
Install
Options
Description

Search: Microsoft.EntityFrameworkCore.Tools

Microsoft.EntityFrameworkCore.Tools by Microsoft, 137M downloads 6.0.3
Entity Framework Core Tools for the NuGet Package Manager Console in Visual Studio.

Microsoft.EntityFrameworkCore.Design by Microsoft, 189M downloads 6.0.3
Shared design-time components for Entity Framework Core tools.

Microsoft.EntityFrameworkCore.Tools.DotNet by Microsoft, 8.04M downlc 2.0.3
Entity Framework Core .NET Command Line Tools. Includes dotnet-ef.

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.

Do not show this again

Version: 3.1.9

Entity Framework Core Tools for the NuGet Package Manager Console in Visual Studio.

Enables these commonly used commands:
Add-Migration
Drop-Database
Get-DbContext
Scaffold-DbContext
Script-Migrations

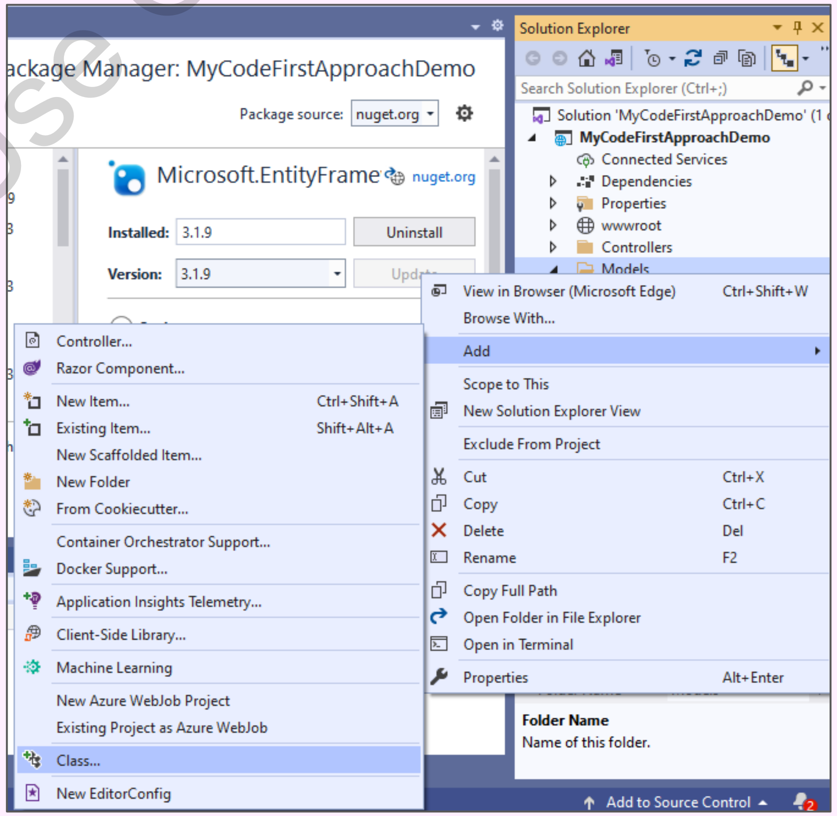
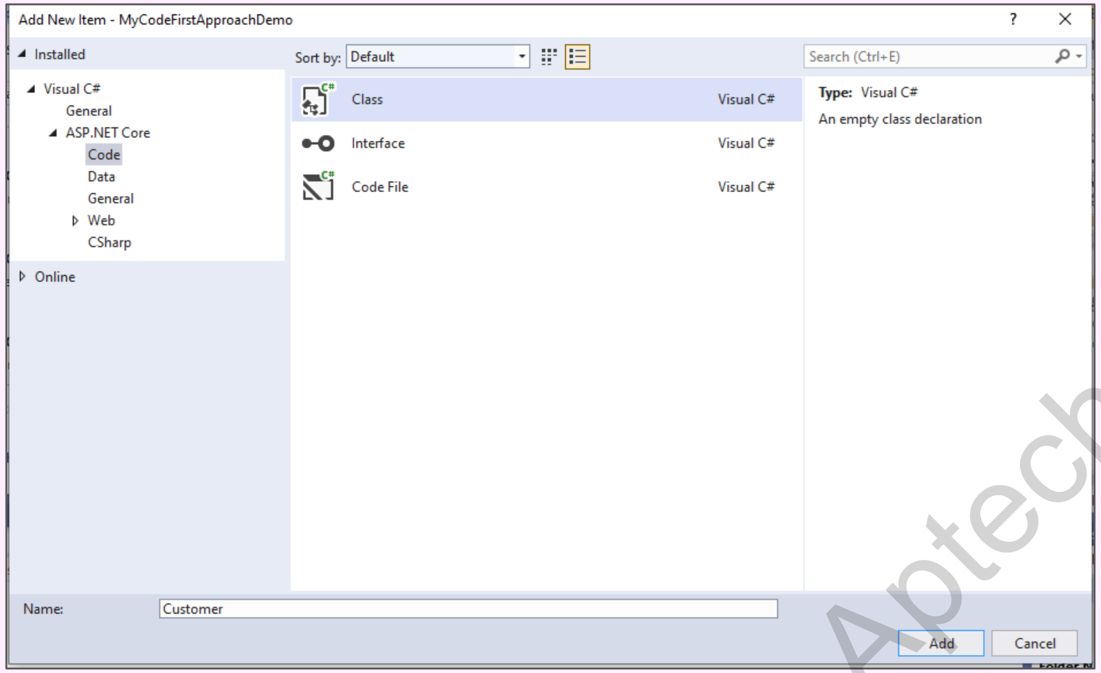


Figure 3.10: EF Tools

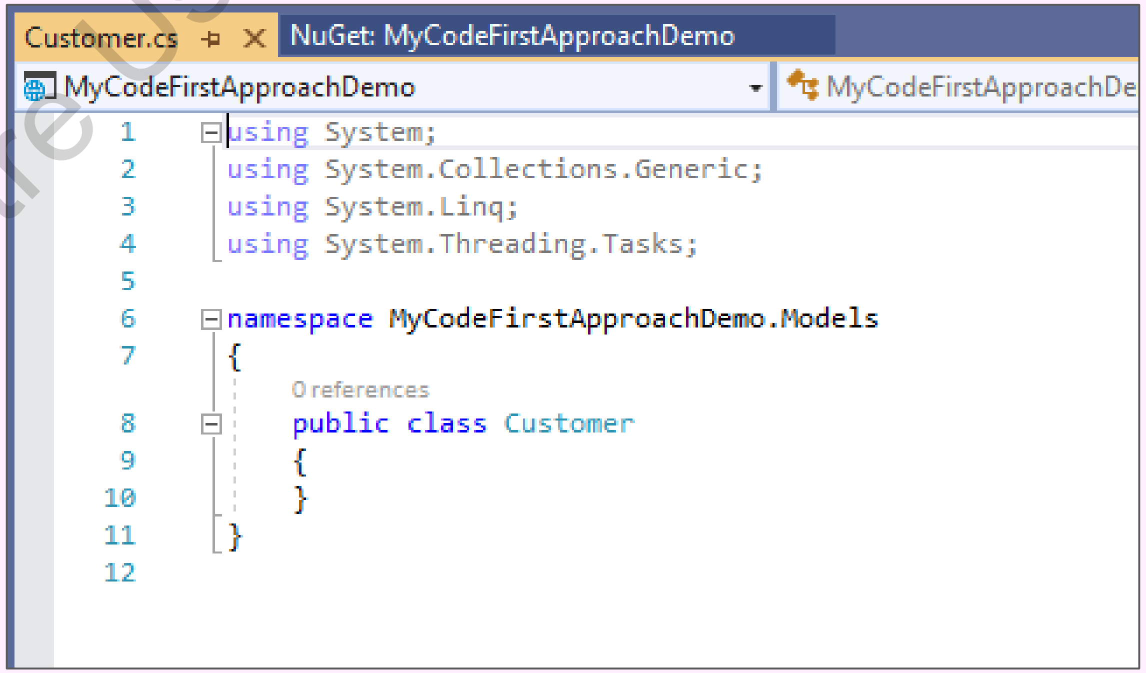
Figure 3.11: Add a Class

Data Handling in ASP.NET MVC with a Code-first Database(5-8)



Name: Customer

Figure 3.12: Change Default Name of Class

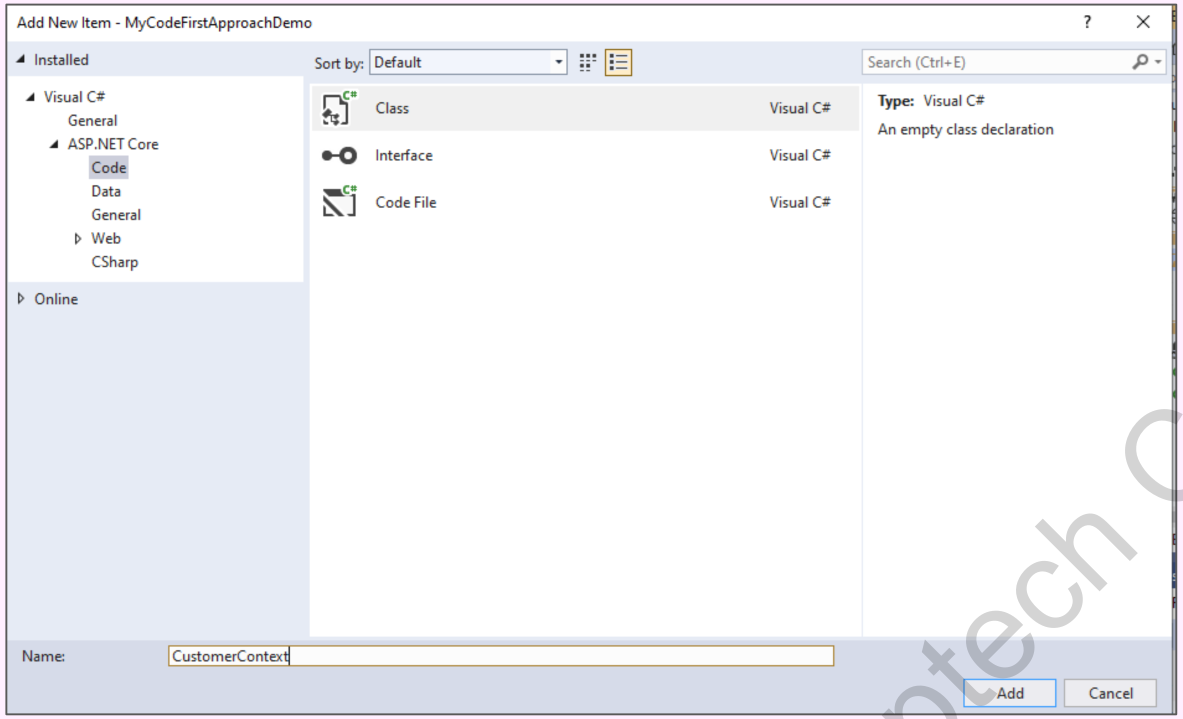


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace MyCodeFirstApproachDemo.Models
{
    0 references
    public class Customer
    {
    }
}
```

Figure 3.13: Default Auto-generated Class Template

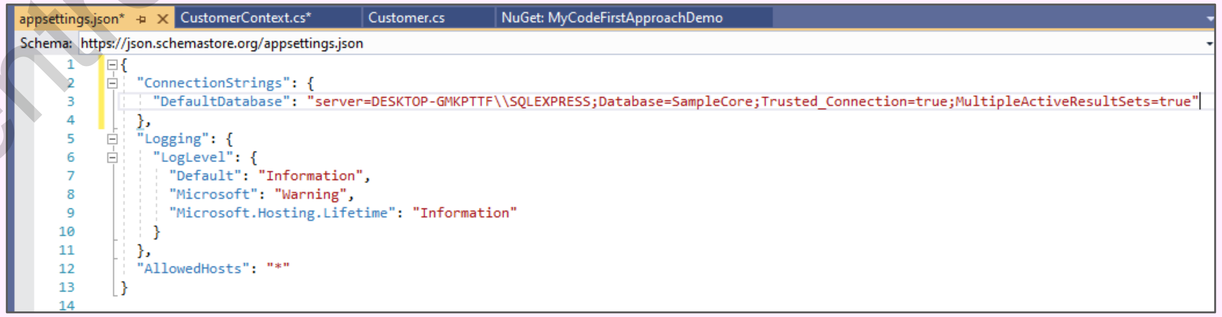
Data Handling in ASP.NET MVC with a Code-first Database(6-8)



Type: Visual C#
An empty class declaration

Name: CustomerContext

Figure 3.14: Adding Database Context

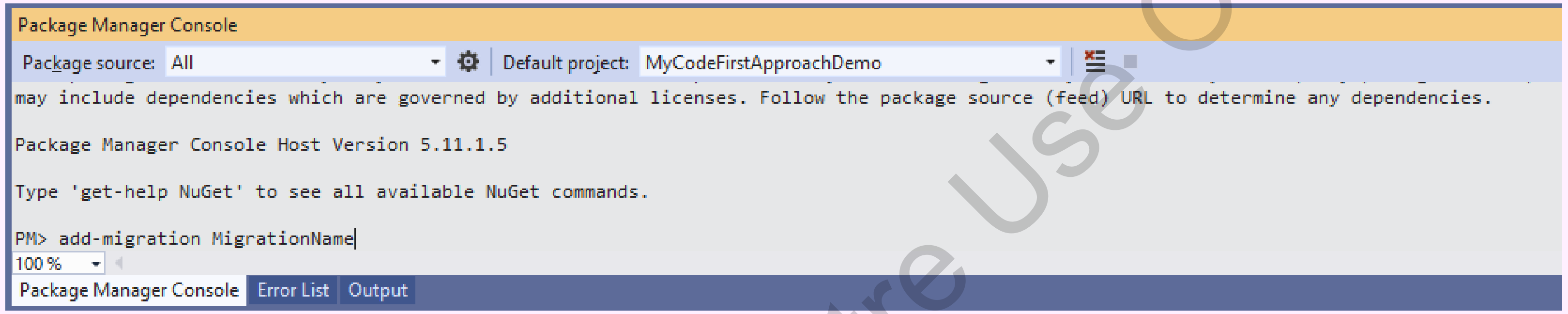


Schema: https://json.schemastore.org/appsettings.json

```json
{
  "ConnectionStrings": {
    "DefaultDatabase": "server=DESKTOP-GMKPTTF\\SQLEXPRESS;Database=SampleCore;Trusted_Connection=true;MultipleActiveResultSets=true"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*"
}
```

Figure 3.15: Configuration File

Data Handling in ASP.NET MVC with a Code-first Database(7-8)



Package Manager Console

Package source: All | Default project: MyCodeFirstApproachDemo

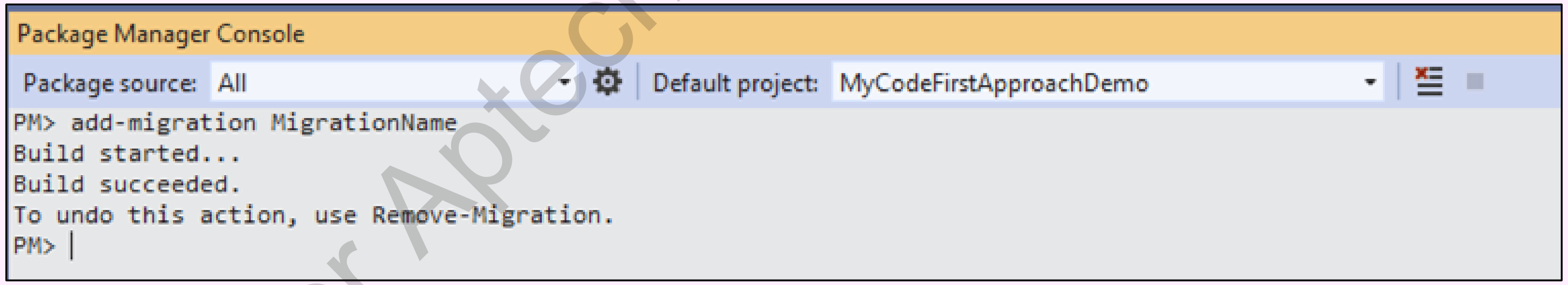
```
may include dependencies which are governed by additional licenses. Follow the package source (feed) URL to determine any dependencies.

Package Manager Console Host Version 5.11.1.5

Type 'get-help NuGet' to see all available NuGet commands.

PM> add-migration MigrationName
```

Figure 3.16: Migration



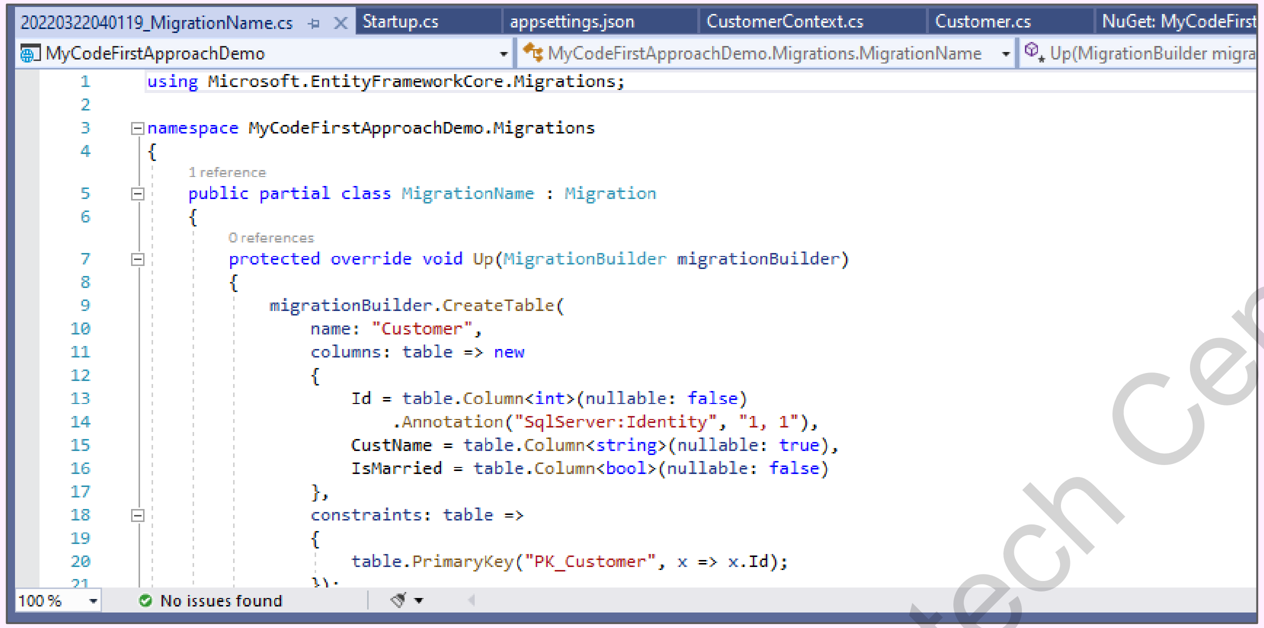
Package Manager Console

Package source: All | Default project: MyCodeFirstApproachDemo

```
PM> add-migration MigrationName
Build started...
Build succeeded.
To undo this action, use Remove-Migration.
PM>
```

Figure 3.17: Build Message

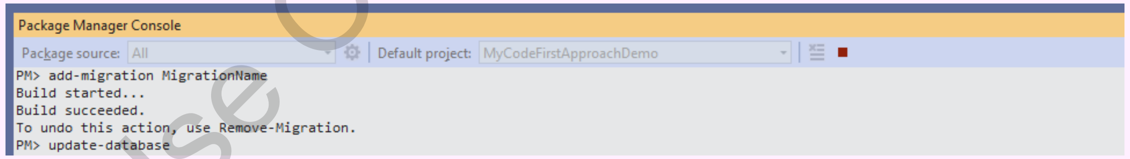
Data Handling in ASP.NET MVC with a Code-first Database(8-8)


20220322040119_MigrationName.cs
Startup.cs
appsettings.json
CustomerContext.cs
Customer.cs
NuGet: MyCodeFirst
MyCodeFirstApproachDemo
MyCodeFirstApproachDemo.Migrations.MigrationName
Up(MigrationBuilder migra

```
using Microsoft.EntityFrameworkCore.Migrations;

namespace MyCodeFirstApproachDemo.Migrations
{
    1 reference
    public partial class MigrationName : Migration
    {
        0 references
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "Customer",
                columns: table => new
                {
                    Id = table.Column<int>(nullable: false)
                        .Annotation("SqlServer:Identity", "1, 1"),
                    CustName = table.Column<string>(nullable: true),
                    IsMarried = table.Column<bool>(nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_Customer", x => x.Id);
                });
```

Figure 3.18: MigrationName Class



```
PM> add-migration MigrationName
Build started...
Build succeeded.
To undo this action, use Remove-Migration.
PM> update-database
```

Figure 3.19: update-database Command

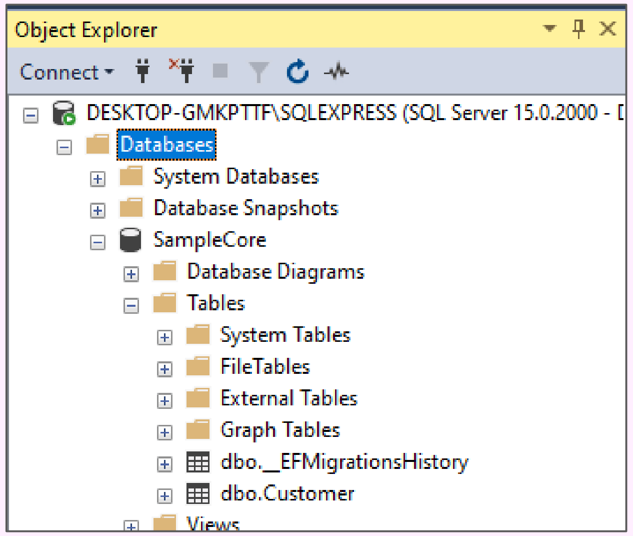


Figure 3.20: Database and Table Created in SQL Server 2019

Summary

- ADO.NET is a core component of the .NET framework and is used for establishing a connection between an application and data sources.
- Data layer is the storage space where data is stored.
- RDO is an object-oriented Data Access interface that combines the simple capabilities of DAO with the low power and flexibility of ODBC.
- ADO is a layer that allows application code to access any data that is stored in a generic manner without the requirement for database implementation.
- The Entity System is an Object-Relational Mapping (ORM)-based database management framework.
- EF is an improved version of ADO.NET that provides programmers with a completely automated database interface.
- EF uses LINQ queries instead of SQL queries.
- In the code-first approach, domain classes are created first.
- EF wizard generates the database tables based on the code.