M10 MATERI 1

1. **TUJUAN**

   CPMK : Mahasiswa dapat memahami dan mengimplementasikan ASP .Net Core.

   Sub-CPMK :

   a. Mahasiswa dapat memahami dan mengimplementasikan ASP .Net Core..

2. **DURASI WAKTU**

   2 pertemuan x 4 jam

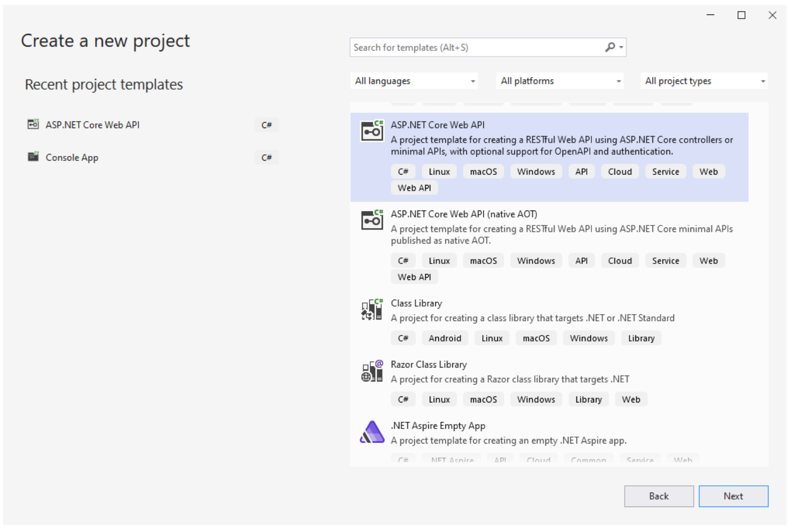
3. **DASAR TEORI**

   ASP .NET Core

4. Percobaan

- **Membuat Projek ASP .NET Core API Sederhana**

1. Buatlah projek **ASP.Net Core Web API** dengan nama **PRG4_M10_P1_XXX** (XXX adalah 3 angka paling belakang di NIM anda), kemudian klik Next.



Create a new project

Recent project templates

ASP.NET Core Web API C#

Console App C#

Search for templates (Alt+S)

All languages | All platforms | All project types

ASP.NET Core Web API
A project template for creating a RESTful Web API using ASP.NET Core controllers or minimal APIs, with optional support for OpenAPI and authentication.
C# Linux macOS Windows API Cloud Service Web Web API

ASP.NET Core Web API (native AOT)
A project template for creating a RESTful Web API using ASP.NET Core minimal APIs published as native AOT.
C# Linux macOS Windows API Cloud Service Web Web API

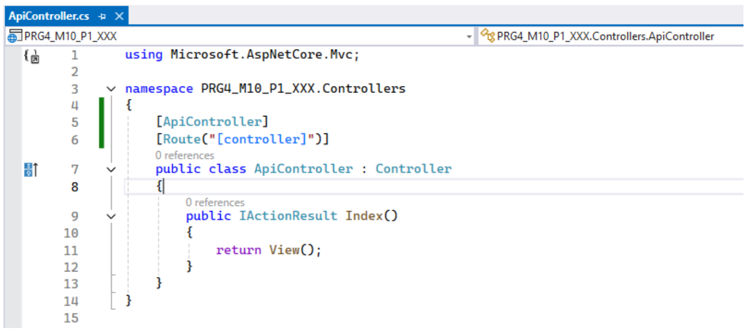
Class Library
A project for creating a class library that targets .NET or .NET Standard
C# Android Linux macOS Windows Library

Razor Class Library
A project for creating a Razor class library that targets .NET
C# Linux macOS Windows Library Web

.NET Aspire Empty App
A project template for creating an empty .NET Aspire app.

Back | Next

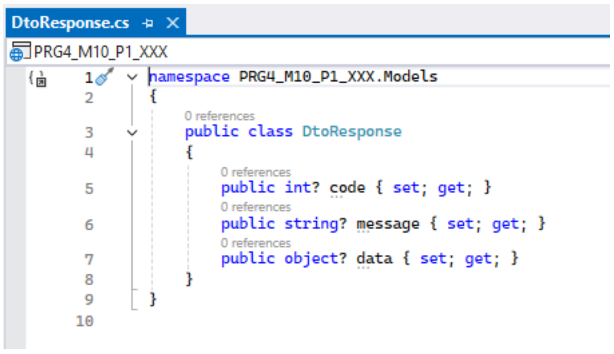
2. Buatlah controllers baru dengan nama "ApiControllers" untuk mengelola api yang akan dibuat.



ApiController.cs — PRG4_M10_P1_XXX — PRG4_M10_P1_XXX.Controllers.ApiController

```csharp
using Microsoft.AspNetCore.Mvc;

namespace PRG4_M10_P1_XXX.Controllers
{
    [ApiController]
    [Route("[controller]")]
    public class ApiController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

3. Selanjutnya, buatlah sebuah folder dengan nama "models", kemudian buatlah sebuah model dengan nama "DtoResponse" yang akan digunakan sebagai response dari api yang dibuat.

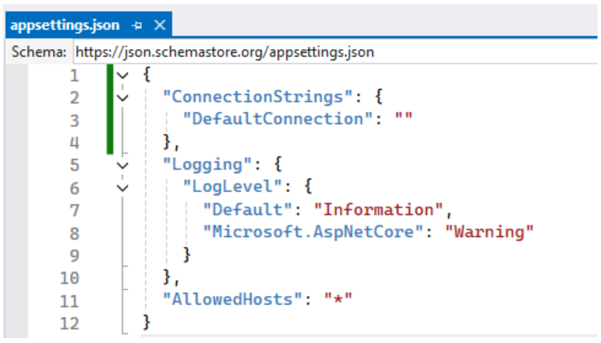


DtoResponse.cs — PRG4_M10_P1_XXX

```csharp
namespace PRG4_M10_P1_XXX.Models
{
    public class DtoResponse
    {
        public int? code { set; get; }
        public string? message { set; get; }
        public object? data { set; get; }
    }
}
```

- **Membuat Connection String**

1. Bukalah appsettings.json, kemudian tambahkan connection string seperti berikut. Connection string akan berisikan alamat dari database yang akan digunakan.



appsettings.json

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

```json
{
  "ConnectionStrings": {
    "DefaultConnection": ""
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*"
}
```

2. Connection String memiliki 2 jenis berdasarkan kredensial otentikasinya, yakni dengan **Windows Authentication** atau **SQL Server Authentication.**

[Windows Authentication]

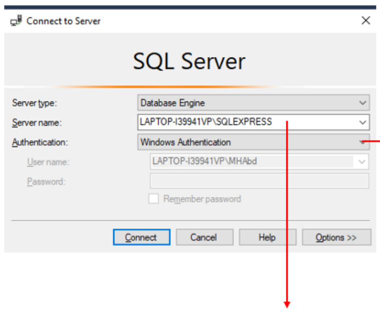
```
"WindowConnection": "Server=..;Database=..;Integrated Security=True;TrustServerCertificate=True;"
```

[SQL Server Authentication]

```
"SQLConnection": "Server=..;Database=..;User Id=..;Password=..;TrustServerCertificate=True;",
```

3. Selanjutnya ceklah SQL Server anda, kemudian pilihlah connection string sesuai dengan kredensial otentikasi yang anda miliki.

[Windows Authentication]



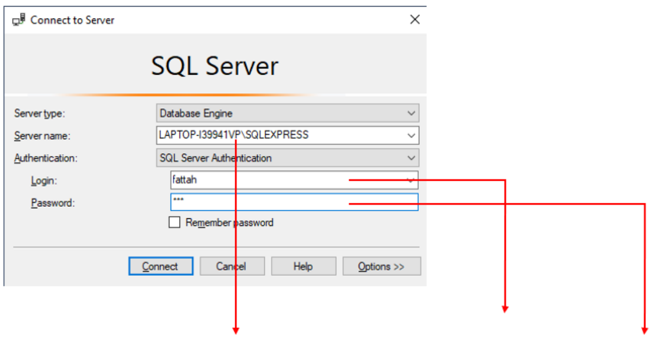
Connect to Server

SQL Server

Server type: Database Engine
Server name: LAPTOP-I39941VP\SQLEXPRESS
Authentication: Windows Authentication
User name: LAPTOP-I39941VP\MHAbd
Password:
Remember password

```
"WindowConnection": "Server=server_anda;Database=database_anda;Integrated Security=True;TrustServerCertificate=True;"
```

[SQL Server Authentication]



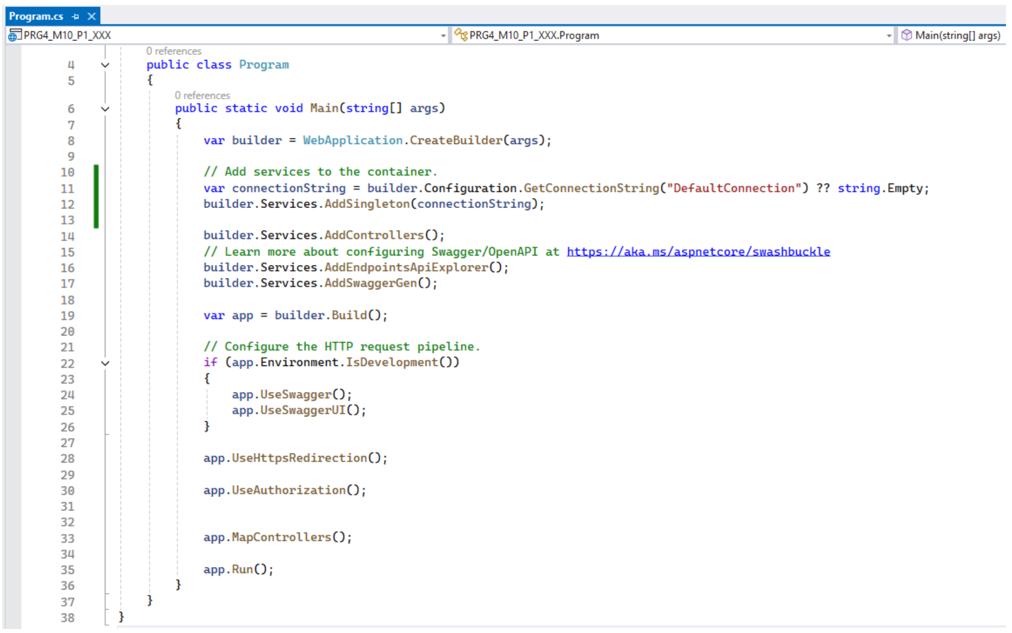
Connect to Server

SQL Server

Server type: Database Engine
Server name: LAPTOP-I39941VP\SQLEXPRESS
Authentication: SQL Server Authentication
Login: fattah
Password: ***
Remember password

```
"SQLConnection": "Server=server_anda;Database=database_anda;User Id=id_anda;Password=password_anda;TrustServerCertificate=True;",
```

4. Kemudian tambahkan connection string pada “Program.cs”.



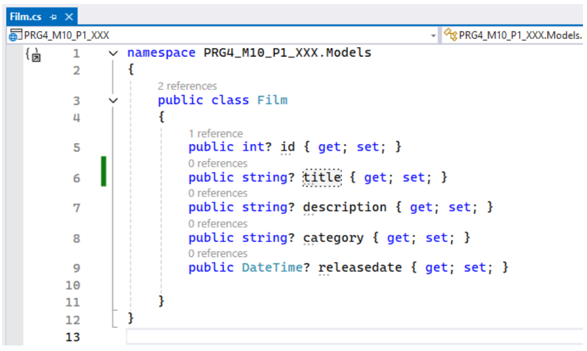
```csharp
public class Program
{
    public static void Main(string[] args)
    {
        var builder = WebApplication.CreateBuilder(args);

        // Add services to the container.
        var connectionString = builder.Configuration.GetConnectionString("DefaultConnection") ?? string.Empty;
        builder.Services.AddSingleton(connectionString);

        builder.Services.AddControllers();
        // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
        builder.Services.AddEndpointsApiExplorer();
        builder.Services.AddSwaggerGen();

        var app = builder.Build();

        // Configure the HTTP request pipeline.
        if (app.Environment.IsDevelopment())
        {
            app.UseSwagger();
            app.UseSwaggerUI();
        }

        app.UseHttpsRedirection();

        app.UseAuthorization();


        app.MapControllers();

        app.Run();
    }
}
```

- **Membuat Database dan Model Baru**

1. Buatlah sebuah database baru dengan menggunakan SQL Server dengan nama “Pemrograman 4”, kemudian buatlah tabel baru dengan nama “film”.

Field	Data Type	Constraint
id	int	primary key, identity(1,1)
title	varchar(50)	null
description	varchar(max)	null
category	varchar(50)	null
releasedate	datetime	null

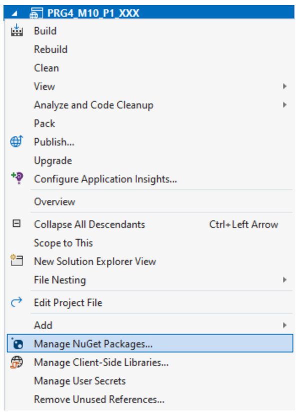
2. Buatlah sebuah model baru dengan nama “film” seperti berikut.



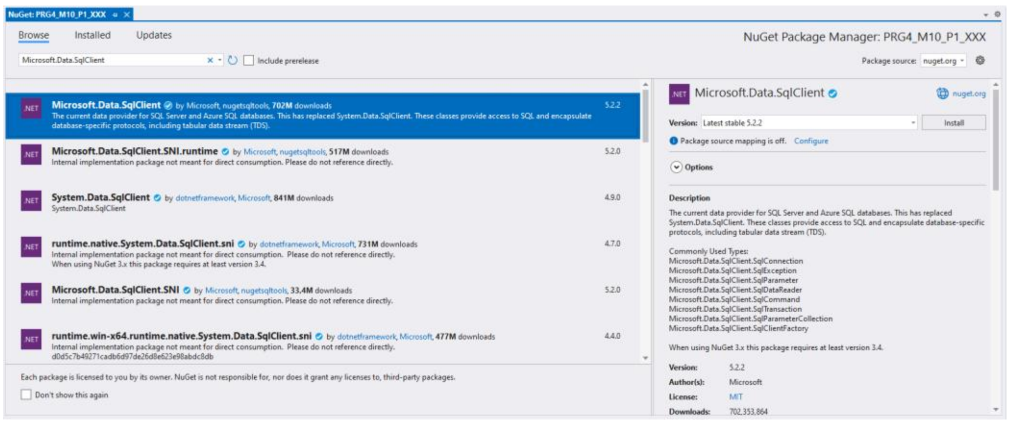
```csharp
namespace PRG4_M10_P1_XXX.Models
{
    public class Film
    {
        public int? id { get; set; }
        public string? title { get; set; }
        public string? description { get; set; }
        public string? category { get; set; }
        public DateTime? releasedate { get; set; }

    }
}
```

- **Konektivitas database dengan ADO .NET**

1. Bukalah Nuget Package Manager (NPM) dengan cara klik kanan projek anda > NPM.



2. Kemudian pilih Browse, lalu carilah Microsoft.Data.SqlClient, selanjutnya installah versi terbaru.



Search: Microsoft.Data.SqlClient

Package source: nuget.org

Microsoft.Data.SqlClient by Microsoft, nugetsqltools, 702M downloads — 5.2.2
The current data provider for SQL Server and Azure SQL databases. This has replaced System.Data.SqlClient. These classes provide access to SQL and encapsulate database-specific protocols, including tabular data stream (TDS).

Microsoft.Data.SqlClient.SNI.runtime by Microsoft, nugetsqltools, 517M downloads — 5.2.0
Internal implementation package not meant for direct consumption. Please do not reference directly.

System.Data.SqlClient by dotnetframework, Microsoft, 841M downloads — 4.9.0
System.Data.SqlClient

runtime.native.System.Data.SqlClient.sni by dotnetframework, Microsoft, 731M downloads — 4.7.0
Internal implementation package not meant for direct consumption. Please do not reference directly.
When using NuGet 3.x this package requires at least version 3.4.

Microsoft.Data.SqlClient.SNI by Microsoft, nugetsqltools, 33.4M downloads — 5.2.0
Internal implementation package not meant for direct consumption. Please do not reference directly.

runtime.win-x64.runtime.native.System.Data.SqlClient.sni by dotnetframework, Microsoft, 477M downloads — 4.4.0
Internal implementation package not meant for direct consumption. Please do not reference directly.
d0d5c7b49271cadb6d97de26d8e623e98abdc8db

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

Don't show this again

Microsoft.Data.SqlClient

Version: Latest stable 5.2.2

Package source mapping is off.

Description

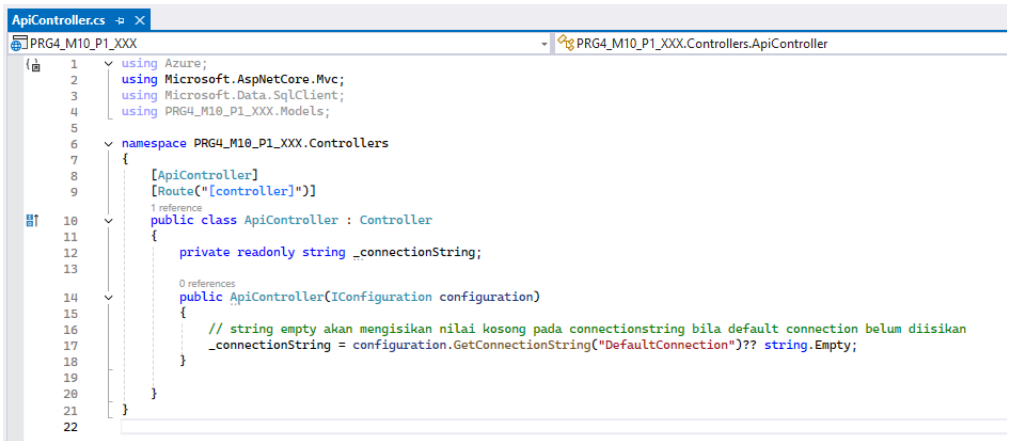
The current data provider for SQL Server and Azure SQL databases. This has replaced System.Data.SqlClient. These classes provide access to SQL and encapsulate database-specific protocols, including tabular data stream (TDS).

Commonly Used Types:
Microsoft.Data.SqlClient.SqlConnection
Microsoft.Data.SqlClient.SqlException
Microsoft.Data.SqlClient.SqlParameter
Microsoft.Data.SqlClient.SqlDataReader
Microsoft.Data.SqlClient.SqlCommand
Microsoft.Data.SqlClient.SqlTransaction
Microsoft.Data.SqlClient.SqlParameterCollection
Microsoft.Data.SqlClient.SqlClientFactory

When using NuGet 3.x this package requires at least version 3.4.

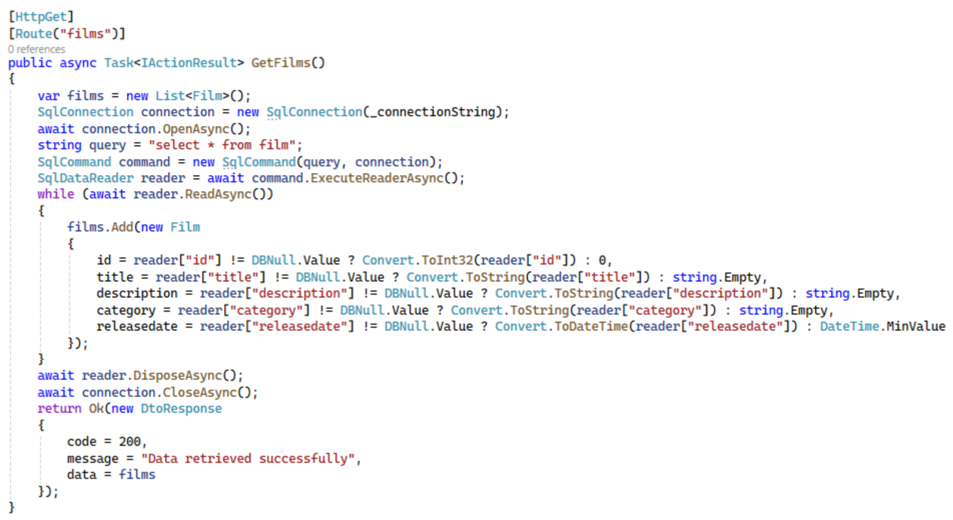
Version:	5.2.2
Author(s):	Microsoft
License:	MIT
Downloads:	702,353,864

3. Selanjutnya tambahkan connection string pada controller yang sebelumnya dibuat.



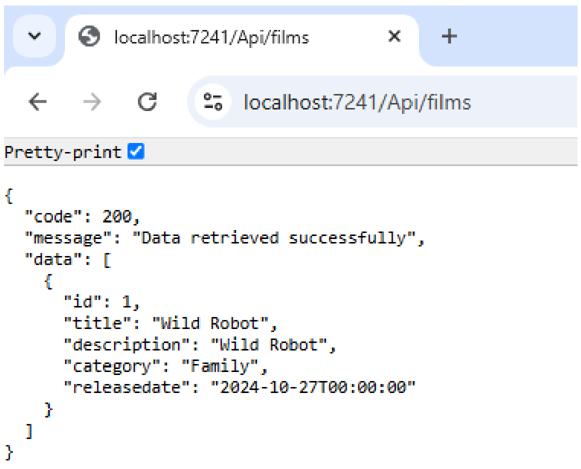
```csharp
using Azure;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Data.SqlClient;
using PRG4_M10_P1_XXX.Models;

namespace PRG4_M10_P1_XXX.Controllers
{
    [ApiController]
    [Route("[controller]")]
    1 reference
    public class ApiController : Controller
    {
        private readonly string _connectionString;

        0 references
        public ApiController(IConfiguration configuration)
        {
            // string empty akan mengisikan nilai kosong pada connectionstring bila default connection belum diisikan
            _connectionString = configuration.GetConnectionString("DefaultConnection")?? string.Empty;
        }

    }
}
```

4. Kemudian buatlah sebuah endpoint seperti berikut!



```
[HttpGet]
[Route("films")]
0 references
public async Task<IActionResult> GetFilms()
{
    var films = new List<Film>();
    SqlConnection connection = new SqlConnection(_connectionString);
    await connection.OpenAsync();
    string query = "select * from film";
    SqlCommand command = new SqlCommand(query, connection);
    SqlDataReader reader = await command.ExecuteReaderAsync();
    while (await reader.ReadAsync())
    {
        films.Add(new Film
        {
            id = reader["id"] != DBNull.Value ? Convert.ToInt32(reader["id"]) : 0,
            title = reader["title"] != DBNull.Value ? Convert.ToString(reader["title"]) : string.Empty,
            description = reader["description"] != DBNull.Value ? Convert.ToString(reader["description"]) : string.Empty,
            category = reader["category"] != DBNull.Value ? Convert.ToString(reader["category"]) : string.Empty,
            releasedate = reader["releasedate"] != DBNull.Value ? Convert.ToDateTime(reader["releasedate"]) : DateTime.MinValue
        });
    }
    await reader.DisposeAsync();
    await connection.CloseAsync();
    return Ok(new DtoResponse
    {
        code = 200,
        message = "Data retrieved successfully",
        data = films
    });
}
```

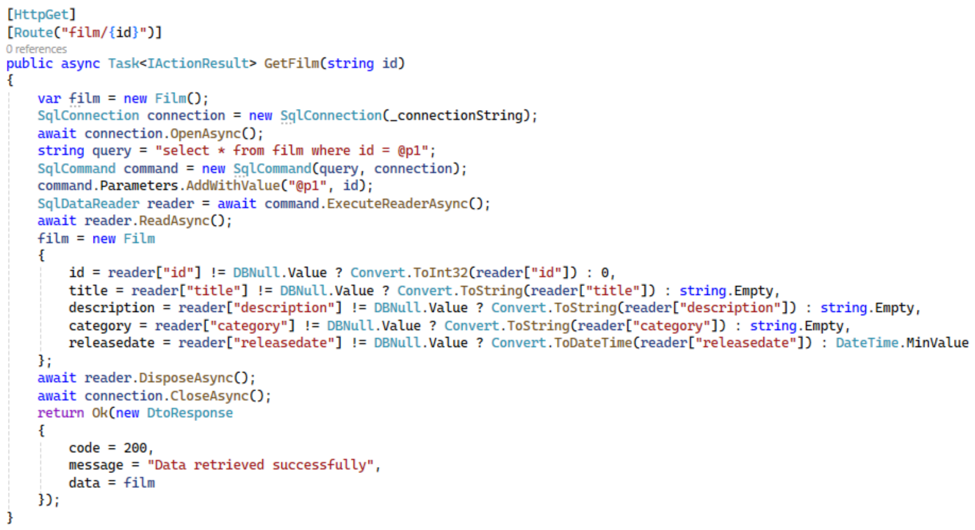
5. Terakhir, jalankan projek dan isilah data pada tabel database, lalu uji coba endpoint tersebut!



```
{
  "code": 200,
  "message": "Data retrieved successfully",
  "data": [
    {
      "id": 1,
      "title": "Wild Robot",
      "description": "Wild Robot",
      "category": "Family",
      "releasedate": "2024-10-27T00:00:00"
    }
  ]
}
```

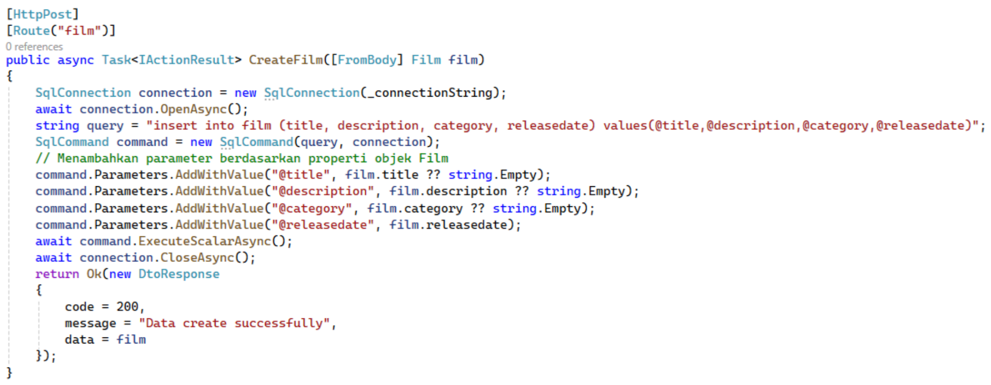
- **Membuat Read, Create, Update dan Delete menggunakan ADO .NET**

[Read]

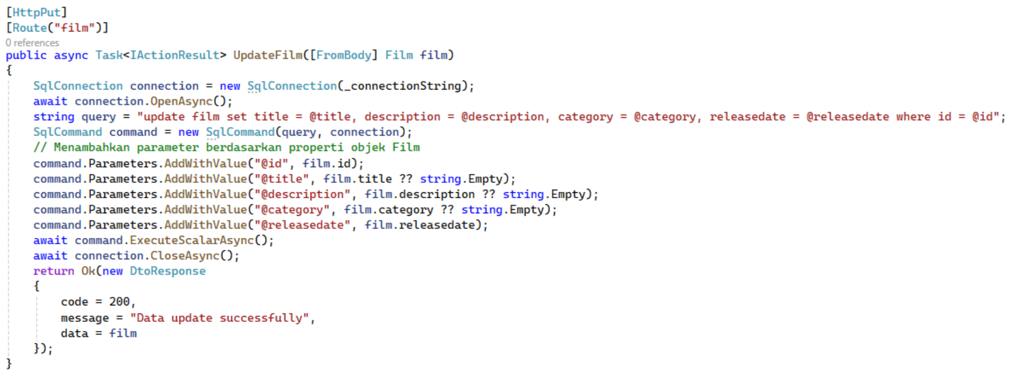


```
[HttpGet]
[Route("film/{id}")]
0 references
public async Task<IActionResult> GetFilm(string id)
{
    var film = new Film();
    SqlConnection connection = new SqlConnection(_connectionString);
    await connection.OpenAsync();
    string query = "select * from film where id = @p1";
    SqlCommand command = new SqlCommand(query, connection);
    command.Parameters.AddWithValue("@p1", id);
    SqlDataReader reader = await command.ExecuteReaderAsync();
    await reader.ReadAsync();
    film = new Film
    {
        id = reader["id"] != DBNull.Value ? Convert.ToInt32(reader["id"]) : 0,
        title = reader["title"] != DBNull.Value ? Convert.ToString(reader["title"]) : string.Empty,
        description = reader["description"] != DBNull.Value ? Convert.ToString(reader["description"]) : string.Empty,
        category = reader["category"] != DBNull.Value ? Convert.ToString(reader["category"]) : string.Empty,
        releasedate = reader["releasedate"] != DBNull.Value ? Convert.ToDateTime(reader["releasedate"]) : DateTime.MinValue
    };
    await reader.DisposeAsync();
    await connection.CloseAsync();
    return Ok(new DtoResponse
    {
        code = 200,
        message = "Data retrieved successfully",
        data = film
    });
}
```

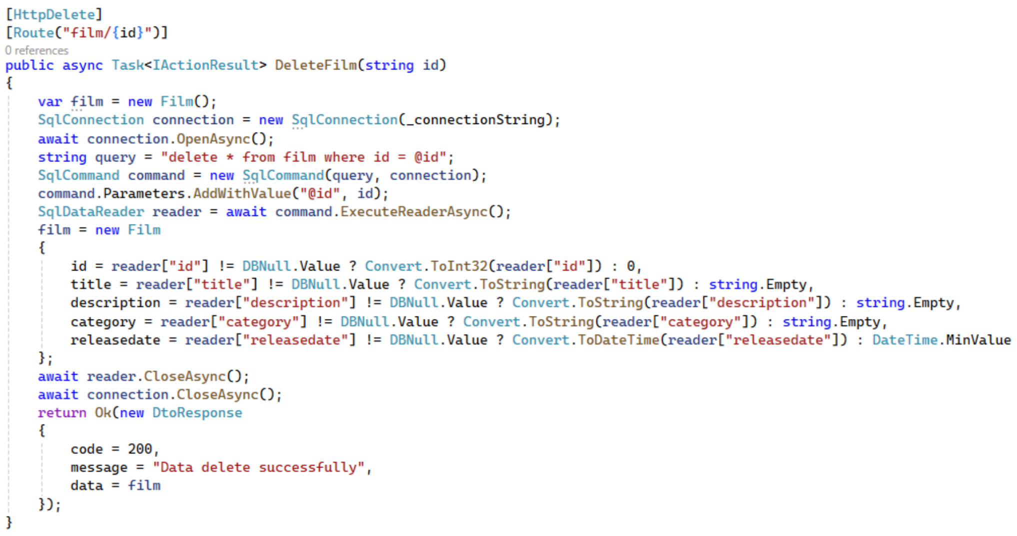
[Create]


```
[HttpPost]
[Route("film")]
0 references
public async Task<IActionResult> CreateFilm([FromBody] Film film)
{
    SqlConnection connection = new SqlConnection(_connectionString);
    await connection.OpenAsync();
    string query = "insert into film (title, description, category, releasedate) values(@title,@description,@category,@releasedate)";
    SqlCommand command = new SqlCommand(query, connection);
    // Menambahkan parameter berdasarkan properti objek Film
    command.Parameters.AddWithValue("@title", film.title ?? string.Empty);
    command.Parameters.AddWithValue("@description", film.description ?? string.Empty);
    command.Parameters.AddWithValue("@category", film.category ?? string.Empty);
    command.Parameters.AddWithValue("@releasedate", film.releasedate);
    await command.ExecuteScalarAsync();
    await connection.CloseAsync();
    return Ok(new DtoResponse
    {
        code = 200,
        message = "Data create successfully",
        data = film
    });
}
```

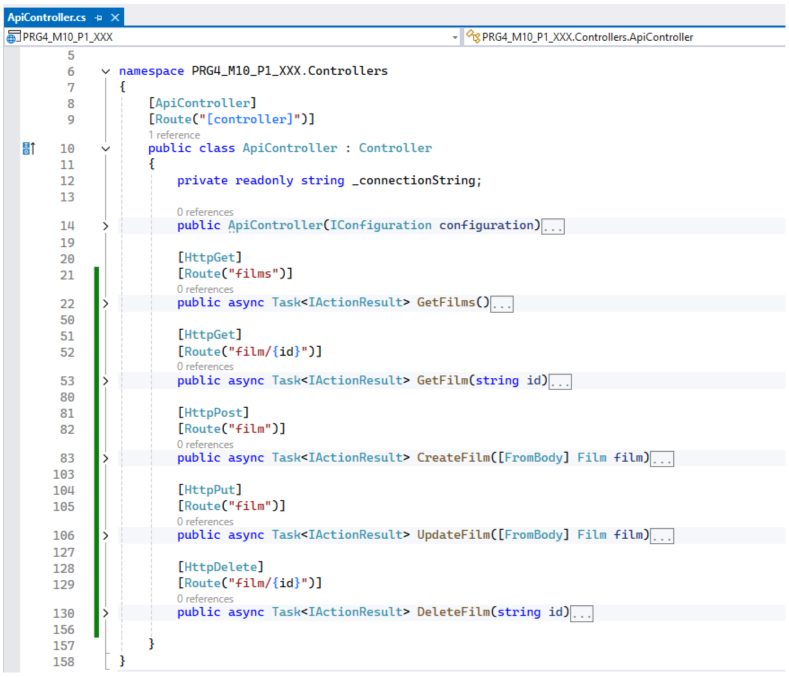
[Update]


```
[HttpPut]
[Route("film")]
0 references
public async Task<IActionResult> UpdateFilm([FromBody] Film film)
{
    SqlConnection connection = new SqlConnection(_connectionString);
    await connection.OpenAsync();
    string query = "update film set title = @title, description = @description, category = @category, releasedate = @releasedate where id = @id";
    SqlCommand command = new SqlCommand(query, connection);
    // Menambahkan parameter berdasarkan properti objek Film
    command.Parameters.AddWithValue("@id", film.id);
    command.Parameters.AddWithValue("@title", film.title ?? string.Empty);
    command.Parameters.AddWithValue("@description", film.description ?? string.Empty);
    command.Parameters.AddWithValue("@category", film.category ?? string.Empty);
    command.Parameters.AddWithValue("@releasedate", film.releasedate);
    await command.ExecuteScalarAsync();
    await connection.CloseAsync();
    return Ok(new DtoResponse
    {
        code = 200,
        message = "Data update successfully",
        data = film
    });
}
```

[Delete]

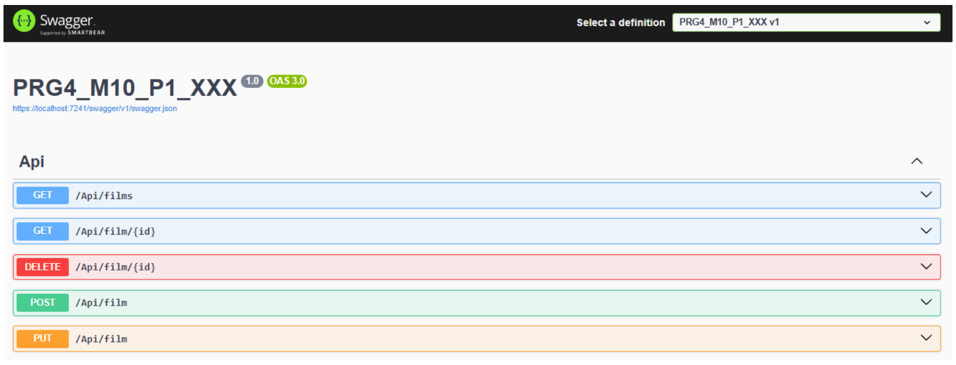

```
[HttpDelete]
[Route("film/{id}")]
0 references
public async Task<IActionResult> DeleteFilm(string id)
{
    var film = new Film();
    SqlConnection connection = new SqlConnection(_connectionString);
    await connection.OpenAsync();
    string query = "delete * from film where id = @id";
    SqlCommand command = new SqlCommand(query, connection);
    command.Parameters.AddWithValue("@id", id);
    SqlDataReader reader = await command.ExecuteReaderAsync();
    film = new Film
    {
        id = reader["id"] != DBNull.Value ? Convert.ToInt32(reader["id"]) : 0,
        title = reader["title"] != DBNull.Value ? Convert.ToString(reader["title"]) : string.Empty,
        description = reader["description"] != DBNull.Value ? Convert.ToString(reader["description"]) : string.Empty,
        category = reader["category"] != DBNull.Value ? Convert.ToString(reader["category"]) : string.Empty,
        releasedate = reader["releasedate"] != DBNull.Value ? Convert.ToDateTime(reader["releasedate"]) : DateTime.MinValue
    };
    await reader.CloseAsync();
    await connection.CloseAsync();
    return Ok(new DtoResponse
    {
        code = 200,
        message = "Data delete successfully",
        data = film
    });
}
```

1. Tambahkan endpoint diatas pada api sebelumnya sehingga menjadi seperti berikut!



```csharp
namespace PRG4_M10_P1_XXX.Controllers
{
    [ApiController]
    [Route("[controller]")]
    1 reference
    public class ApiController : Controller
    {
        private readonly string _connectionString;

        0 references
        public ApiController(IConfiguration configuration)[...]

        [HttpGet]
        [Route("films")]
        0 references
        public async Task<IActionResult> GetFilms()[...]

        [HttpGet]
        [Route("film/{id}")]
        0 references
        public async Task<IActionResult> GetFilm(string id)[...]

        [HttpPost]
        [Route("film")]
        0 references
        public async Task<IActionResult> CreateFilm([FromBody] Film film)[...]

        [HttpPut]
        [Route("film")]
        0 references
        public async Task<IActionResult> UpdateFilm([FromBody] Film film)[...]

        [HttpDelete]
        [Route("film/{id}")]
        0 references
        public async Task<IActionResult> DeleteFilm(string id)[...]
    }
}
```

2. Kemudian, cobalah setiap endpoint yang sudah dibuat.

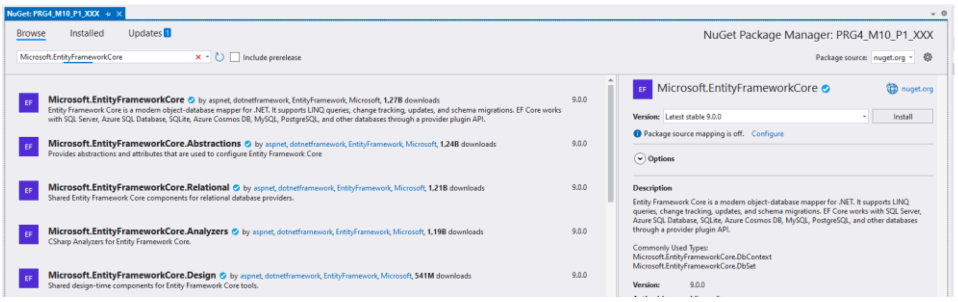


GET /Api/films
GET /Api/film/{id}
DELETE /Api/film/{id}
POST /Api/film
PUT /Api/film

- ## Konektivitas database dengan Entity Framework

1. Buka kembali Nuget Package Manager, kemudian installah library berikut.

   [1] Microsoft.EntityFrameworkCore



Microsoft.EntityFrameworkCore by aspnet, dotnetframework, EntityFramework, Microsoft, 1,27B downloads 9.0.0
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, Azure SQL Database, SQLite, Azure Cosmos DB, MySQL, PostgreSQL, and other databases through a provider plugin API.

Microsoft.EntityFrameworkCore.Abstractions by aspnet, dotnetframework, EntityFramework, Microsoft, 1,24B downloads 9.0.0
Provides abstractions and attributes that are used to configure Entity Framework Core

Microsoft.EntityFrameworkCore.Relational by aspnet, dotnetframework, EntityFramework, Microsoft, 1,21B downloads 9.0.0
Shared Entity Framework Core components for relational database providers.

Microsoft.EntityFrameworkCore.Analyzers by aspnet, dotnetframework, EntityFramework, Microsoft, 1,19B downloads 9.0.0
CSharp Analyzers for Entity Framework Core.

Microsoft.EntityFrameworkCore.Design by aspnet, dotnetframework, EntityFramework, Microsoft, 541M downloads 9.0.0
Shared design-time components for Entity Framework Core tools.

Microsoft.EntityFrameworkCore

Version: Latest stable 9.0.0

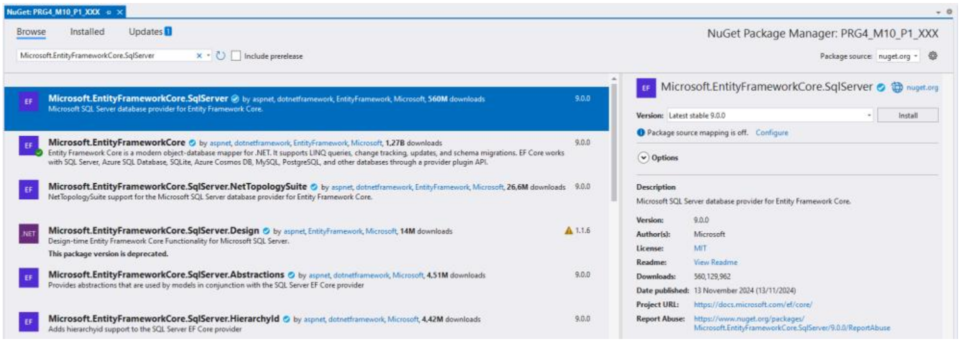
Package source mapping is off.

Description
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, Azure SQL Database, SQLite, Azure Cosmos DB, MySQL, PostgreSQL, and other databases through a provider plugin API.

Commonly Used Types:
Microsoft.EntityFrameworkCore.DbContext
Microsoft.EntityFrameworkCore.DbSet

Version: 9.0.0

   [2] Microsoft.EntityFrameworkCore.SqlServer


NuGet: PRG4_M10_P1_XXX

Browse | Installed | Updates 1

Microsoft.EntityFrameworkCore.SqlServer ☐ Include prerelease

NuGet Package Manager: PRG4_M10_P1_XXX

Package source: nuget.org

Microsoft.EntityFrameworkCore.SqlServer by aspnet, dotnetframework, EntityFramework, Microsoft, **560M** downloads Microsoft SQL Server database provider for Entity Framework Core.	9.0.0
Microsoft.EntityFrameworkCore by aspnet, dotnetframework, EntityFramework, Microsoft, **1,27B** downloads 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, Azure SQL Database, SQLite, Azure Cosmos DB, MySQL, PostgreSQL, and other databases through a provider plugin API.	9.0.0
Microsoft.EntityFrameworkCore.SqlServer.NetTopologySuite by aspnet, dotnetframework, EntityFramework, Microsoft, **26,6M** downloads NetTopologySuite support for the Microsoft SQL Server database provider for Entity Framework Core.	9.0.0
Microsoft.EntityFrameworkCore.SqlServer.Design by aspnet, EntityFramework, Microsoft, **14M** downloads Design-time Entity Framework Core Functionality for Microsoft SQL Server. **This package version is deprecated.**	⚠ 1.1.6
Microsoft.EntityFrameworkCore.SqlServer.Abstractions by aspnet, dotnetframework, Microsoft, **4,51M** downloads Provides abstractions that are used by models in conjunction with the SQL Server EF Core provider	9.0.0
Microsoft.EntityFrameworkCore.SqlServer.HierarchyId by aspnet, dotnetframework, Microsoft, **4,42M** downloads Adds hierarchyid support to the SQL Server EF Core provider	9.0.0

Microsoft.EntityFrameworkCore.SqlServer nuget.org

Version: Latest stable 9.0.0 Install

Package source mapping is off. Configure

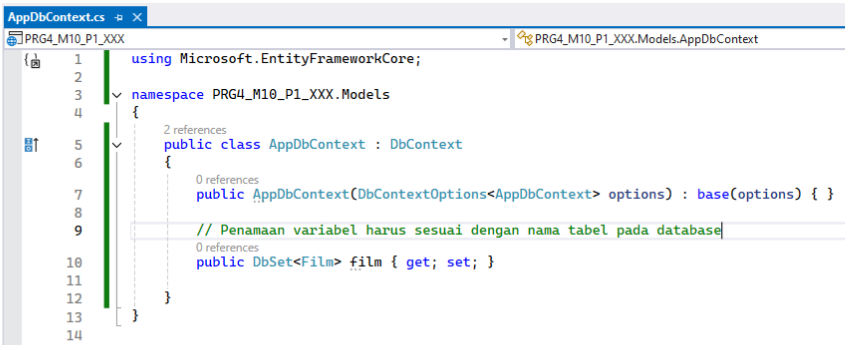
Options

Description

Microsoft SQL Server database provider for Entity Framework Core.

Version:	9.0.0
Author(s):	Microsoft
License:	MIT
Readme:	View Readme
Downloads:	560,129,962
Date published:	13 November 2024 (13/11/2024)
Project URL:	https://docs.microsoft.com/ef/core/
Report Abuse:	https://www.nuget.org/packages/Microsoft.EntityFrameworkCore.SqlServer/9.0.0/ReportAbuse

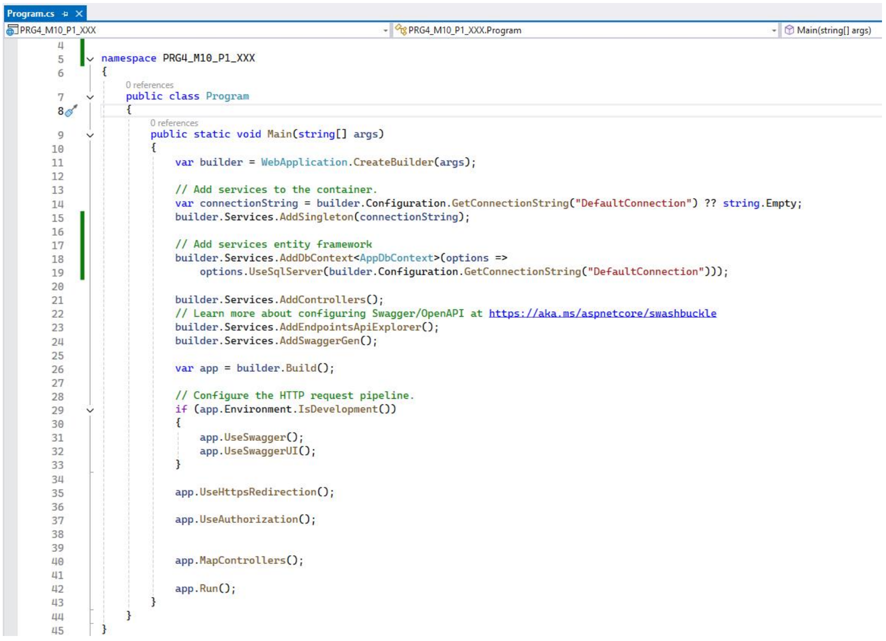
2. Selanjutnya buatlah file AppDbContext.cs pada folder models. Fungsi AppDbContext adalah untuk menghubungkan konseptual data aplikasi dengan physical data database sehingga dapat melakukan operasi CRUD dan menjalankan query.


AppDbContext.cs

PRG4_M10_P1_XXX | PRG4_M10_P1_XXX.Models.AppDbContext

```csharp
using Microsoft.EntityFrameworkCore;

namespace PRG4_M10_P1_XXX.Models
{
    2 references
    public class AppDbContext : DbContext
    {
        0 references
        public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) { }

        // Penamaan variabel harus sesuai dengan nama tabel pada database
        0 references
        public DbSet<Film> film { get; set; }

    }
}
```

3. Kemudian, daftarkan AppDbContext pada “Program.cs”


Program.cs

PRG4_M10_P1_XXX | PRG4_M10_P1_XXX.Program | Main(string[] args)

```csharp

namespace PRG4_M10_P1_XXX
{
    0 references
    public class Program
    {
        0 references
        public static void Main(string[] args)
        {
            var builder = WebApplication.CreateBuilder(args);

            // Add services to the container.
            var connectionString = builder.Configuration.GetConnectionString("DefaultConnection") ?? string.Empty;
            builder.Services.AddSingleton(connectionString);

            // Add services entity framework
            builder.Services.AddDbContext<AppDbContext>(options =>
                options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

            builder.Services.AddControllers();
            // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
            builder.Services.AddEndpointsApiExplorer();
            builder.Services.AddSwaggerGen();

            var app = builder.Build();

            // Configure the HTTP request pipeline.
            if (app.Environment.IsDevelopment())
            {
                app.UseSwagger();
                app.UseSwaggerUI();
            }

            app.UseHttpsRedirection();

            app.UseAuthorization();


            app.MapControllers();

            app.Run();
        }
    }
}
```

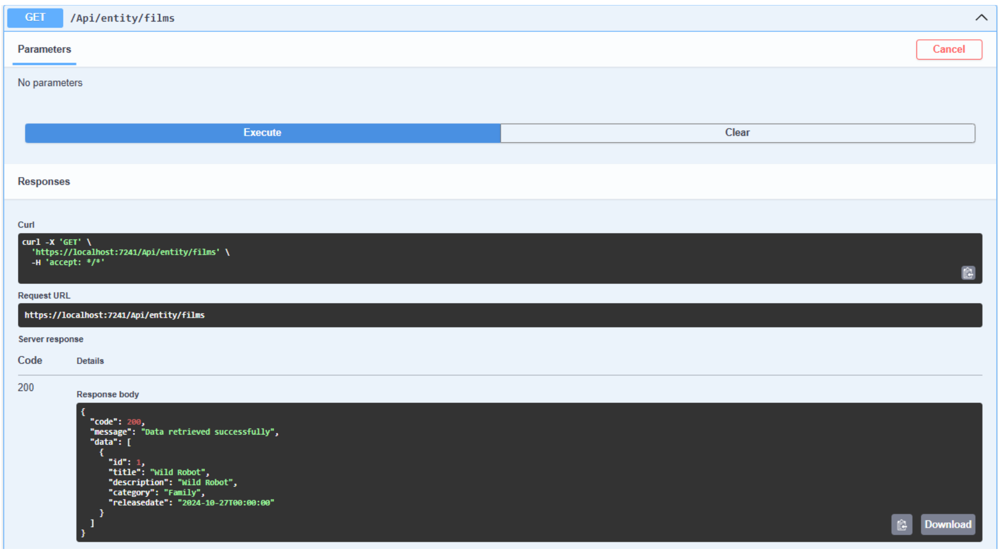
4. Selanjutnya tambahkan AppDbContext pada controller yang sebelumnya dibuat.

```
public class ApiController : Controller
{
    private readonly string _connectionString;
    private readonly AppDbContext _context;


    0 references
    public ApiController(IConfiguration configuration, AppDbContext context)
    {
        // string empty akan mengisikan nilai kosong pada connectionstring bila default connection belum diisikan
        _connectionString = configuration.GetConnectionString("DefaultConnection")?? string.Empty;

        // menambahkan AppDbContext
        _context = context;
    }
```

5. Kemudian tambahkan endpoint seperti berikut!

```
[HttpGet]
[Route("entity/films")]
0 references
public async Task<IActionResult> GetEntityFilms()
{
     var films = await _context.film.ToListAsync();
     return Ok(new DtoResponse
     {
          code = 200,
          message = "Data retrieved successfully",
          data = films
     });
}
```

6. Terakhir, jalankan projek dan isilah data pada tabel database, lalu uji coba endpoint tersebut.



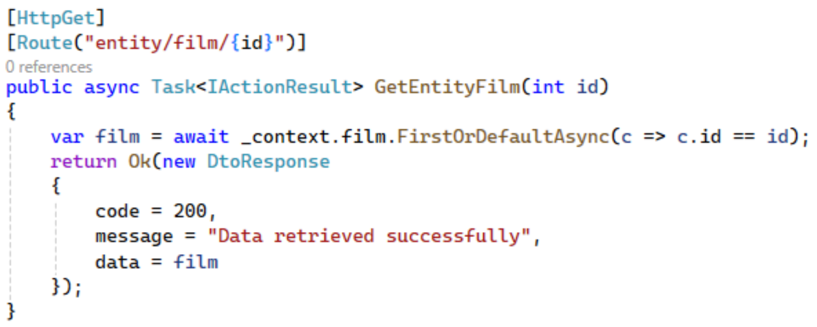
Curl
```
curl -X 'GET' \
  'https://localhost:7241/Api/entity/films' \
  -H 'accept: */*'
```
Request URL
```
https://localhost:7241/Api/entity/films
```
Server response

Code Details

200 Response body
```
{
  "code": 200,
  "message": "Data retrieved successfully",
  "data": [
    {
      "id": 1,
      "title": "Wild Robot",
      "description": "Wild Robot",
      "category": "Family",
      "releasedate": "2024-10-27T00:00:00"
    }
  ]
}
```

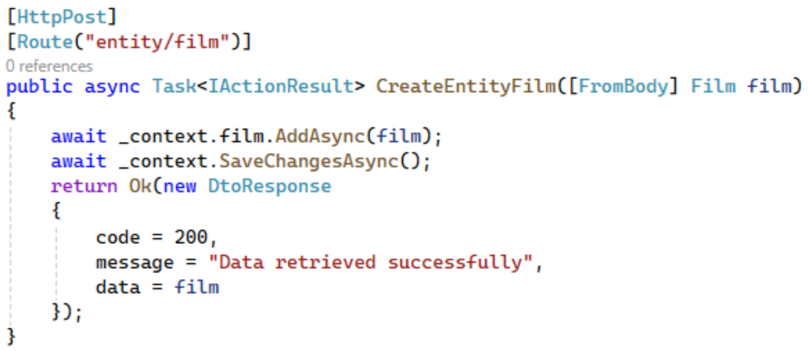
- **Membuat Read, Create, Update dan Delete menggunakan Entity Framework**

  [Read]



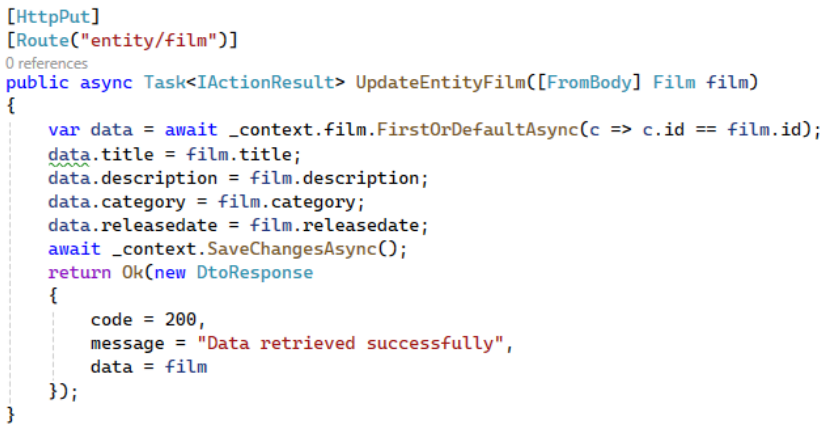
```
[HttpGet]
[Route("entity/film/{id}")]
0 references
public async Task<IActionResult> GetEntityFilm(int id)
{
    var film = await _context.film.FirstOrDefaultAsync(c => c.id == id);
    return Ok(new DtoResponse
    {
        code = 200,
        message = "Data retrieved successfully",
        data = film
    });
}
```

[Create]



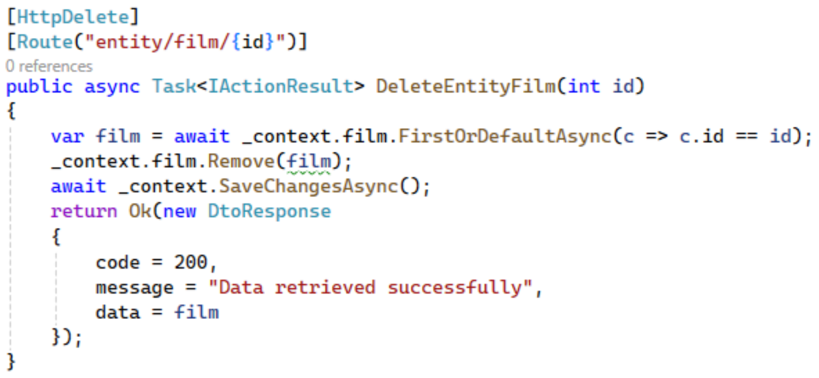
```
[HttpPost]
[Route("entity/film")]
0 references
public async Task<IActionResult> CreateEntityFilm([FromBody] Film film)
{
    await _context.film.AddAsync(film);
    await _context.SaveChangesAsync();
    return Ok(new DtoResponse
    {
        code = 200,
        message = "Data retrieved successfully",
        data = film
    });
}
```

[Update]



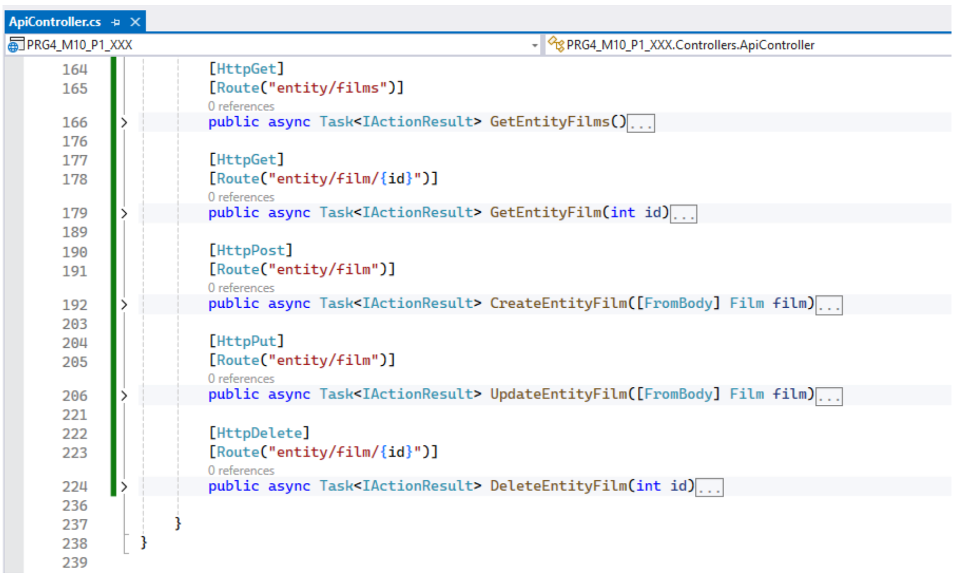
```
[HttpPut]
[Route("entity/film")]
0 references
public async Task<IActionResult> UpdateEntityFilm([FromBody] Film film)
{
    var data = await _context.film.FirstOrDefaultAsync(c => c.id == film.id);
    data.title = film.title;
    data.description = film.description;
    data.category = film.category;
    data.releasedate = film.releasedate;
    await _context.SaveChangesAsync();
    return Ok(new DtoResponse
    {
        code = 200,
        message = "Data retrieved successfully",
        data = film
    });
}
```

[Delete]\



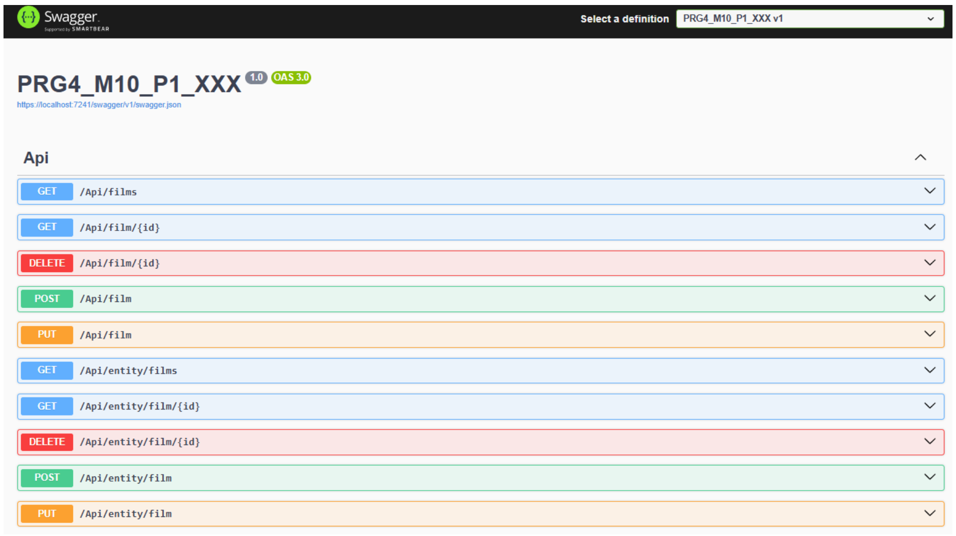
```
[HttpDelete]
[Route("entity/film/{id}")]
0 references
public async Task<IActionResult> DeleteEntityFilm(int id)
{
    var film = await _context.film.FirstOrDefaultAsync(c => c.id == id);
    _context.film.Remove(film);
    await _context.SaveChangesAsync();
    return Ok(new DtoResponse
    {
        code = 200,
        message = "Data retrieved successfully",
        data = film
    });
}
```

1. Tambahkan endpoint diatas pada api sebelumnya sehingga menjadi seperti berikut!



```
ApiController.cs  ⊐  ×
PRG4_M10_P1_XXX                                        PRG4_M10_P1_XXX.Controllers.ApiController
        [HttpGet]
        [Route("entity/films")]
        0 references
        public async Task<IActionResult> GetEntityFilms()[...]

        [HttpGet]
        [Route("entity/film/{id}")]
        0 references
        public async Task<IActionResult> GetEntityFilm(int id)[...]

        [HttpPost]
        [Route("entity/film")]
        0 references
        public async Task<IActionResult> CreateEntityFilm([FromBody] Film film)[...]

        [HttpPut]
        [Route("entity/film")]
        0 references
        public async Task<IActionResult> UpdateEntityFilm([FromBody] Film film)[...]

        [HttpDelete]
        [Route("entity/film/{id}")]
        0 references
        public async Task<IActionResult> DeleteEntityFilm(int id)[...]

    }
}
```

2. Kemudian, cobalah setiap endpoint yang sudah dibuat.



PRG4_M10_P1_XXX 1.0 OAS 3.0

https://localhost:7241/swagger/v1/swagger.json

Api

```
GET     /Api/films
GET     /Api/film/{id}
DELETE  /Api/film/{id}
POST    /Api/film
PUT     /Api/film
GET     /Api/entity/films
GET     /Api/entity/film/{id}
DELETE  /Api/entity/film/{id}
POST    /Api/entity/film
PUT     /Api/entity/film
```

1. Latihan

- Lengkapilah API yang anda buat sebelumnya dengan.
  1. Menggunakan Best Practices
  2. Menggunakan Exception Handling