# EF Core – ASP.NET Essentials II

Problems for exercises for the ["Entity Framework Core" course @ SoftUni](https://softuni.bg/trainings/4842/entity-framework-core-february-2025)

A movie ticket and popcorn

Description automatically generated

## Overview

In this exercise, students will focus on implementing the **backend logic** of the CinemaApp project. The primary objective is to build **CRUD (Create, Read, Update, Delete) operations** for managing data while **working directly with DbContext via Dependency Injection (DI).** This will provide a hands-on understanding of handling database interactions efficiently in an ASP.NET Core MVC application.

### Learning Objectives

By completing this exercise, students will:

* Implement **CRUD functionality** in controllers for key entities
* Work with **DbContext** using **Dependency Injection**
* Understand how to **structure controller logic** for **interacting with the database**
* Learn how to **manage data import and export** using helper methods

### Best Practices and Approach

A common **best practice in ASP.NET Core applications** is to use:

* **Repository Pattern**
* **Service Layer**

to **separate concerns** and **avoid direct interaction with DbContext in controllers**.

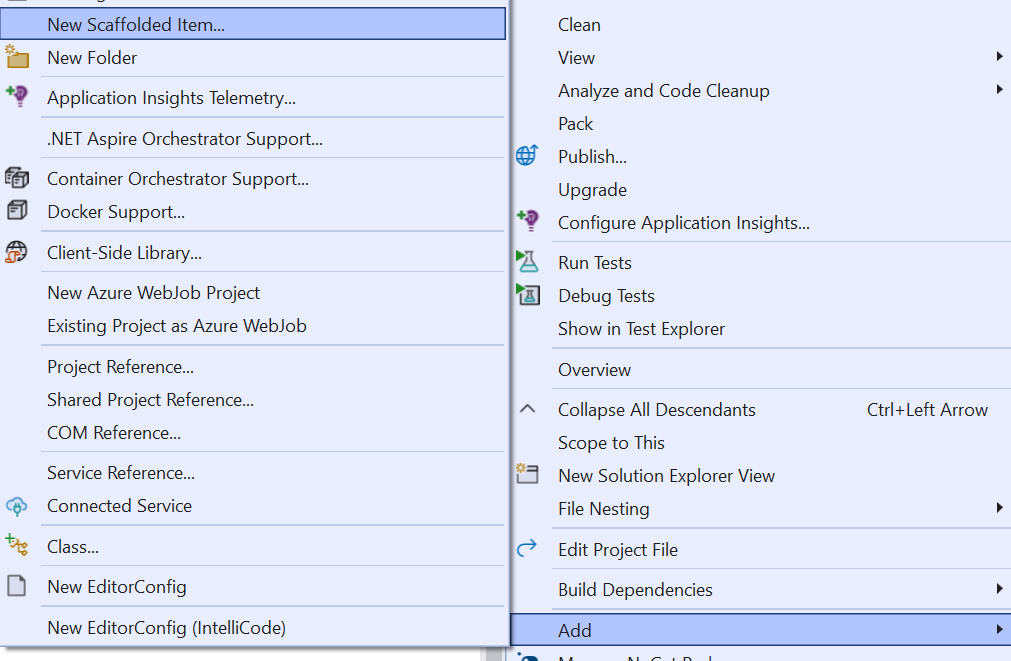
However, **to keep the application simple** and **avoid introducing too much information at the beginning**, we will directly interact with DbContext in this exercise.

Later, students can **refactor** the application to follow best practices.

## Identity Setup

### Scaffolding Identity

* Right-click on the **Project** in **Solution Explorer**
* Select **Add > New Scaffolded Item**
* Choose **Identity**
* Select **Account: Login, Register, Logout**
* Choose CinemaAppDbContext as the data context
* Click **Add**



A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

### Change IdentityUser to ApplicationUser

After scaffolding, the Identity pages (such as Register.cshtml and Login.cshtml) will **use IdentityUser by default**. However, if you have a custom ApplicationUser class, you must **update all** occurrences of IdentityUser in the scaffolded files.

* Update Register.cshtml.cs, Login.cshtml.cs and Logout.cshtml.cs
* **Apply Migrations**

### Register a User and Extract User IDs from SSMS

* **Editing** \_LoginPartial.cshtmlA screenshot of a computer

  AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

By default, the \_LoginPartial.cshtml file injects UserManager<IdentityUser> and SignInManager<IdentityUser>.

However, since we are using ApplicationUser, you must **update this file** to prevent runtime errors.

* **Remove Password Restrictions**

**A screen shot of a computer code

AI-generated content may be incorrect.**

* **Run the Application** and navigate to the **Register page:**A screenshot of a computer

  AI-generated content may be incorrect.

A screenshot of a login form

AI-generated content may be incorrect.

A close up of a sign

AI-generated content may be incorrect.

A screenshot of a computer code

AI-generated content may be incorrect.

## Creating a DataProcessor static class

### Seeding Files for Import

* Additional movies to import:  
   
* CinemasMovies relations to import:  
  
* Tickets to import:  
  

Contains a **list of predefined movies** that will be imported into the database

The tickets.xml file includes ticket data with **placeholders for `UserId` elements**.

**These values are initially \*\*empty\*\*** and can be updated once **users are registered** through the default \*\*ASP.NET Identity\*\* registration functionality.

This ensures that ticket ownership can be correctly assigned.

### Assigning Users to Tickets in the XML File

Now that we have users created in the database, we need to **update the tickets.xml** file by **assigning Id values** to each ticket.



### Where to Place the DataProcessor Class?

**The DataProcessor class should be placed in a logical and reusable location within your ASP.NET Core project**

**A screenshot of a computer

AI-generated content may be incorrect.**

### Creating DataProcessor Class Empty Methods

A screenshot of a computer program

AI-generated content may be incorrect.

A computer code with many different colored text

AI-generated content may be incorrect.

### Create a Folder for Data Files

**A screenshot of a computer

AI-generated content may be incorrect.**

## Triggering DataProcessor When the App Starts

* **To execute the DataProcessor at startup, we need to modify Program.cs to call DataSeeder.**
* **This will ensure that the data is imported when the application starts.**

**A screen shot of a computer program

AI-generated content may be incorrect.**

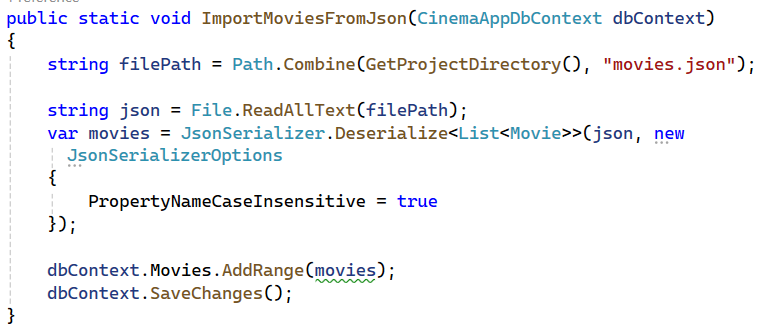
#### Comment Out the Following Lines

A screen shot of a computer program

AI-generated content may be incorrect.

* We will **uncomment one by one** the DataProcessor methods, and seed the new entries in the database

#### Import Movies From Json



#### Ensure Movies are Imported Successfully

A screenshot of a computer

AI-generated content may be incorrect.

## Next Steps for Import:

#### ImportCinemasMoviesFromJson

* Create a CinemaMovieDTO

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

#### ImportTicketsFromXML