# EF Core – XML & JSON Processing

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 workshop, students will learn how to **import and export data** in an ASP.NET Core application using **Entity Framework Core**. The focus will be on working with **JSON and XML files**, utilizing serialization and deserialization techniques to transfer data between external files and the database.

## Setup Program.cs

* **Create a new scope** within the application's **dependency injection (DI) container**
* The **using statement** ensures that **resources are properly disposed of after execution**
* ServiceProvider is used to **access registered services** within the application's DI system

A screen shot of a computer program

AI-generated content may be incorrect.

## Import Data

**Setting Up the Files for Automatic Copying**

* By default, **non-code files (such as .json and .xml) are not copied** to the output directory
* If the file is missing, the application **will fail** when trying to read it
* Ensuring that the file is copied **every time the project runs** allows for consistent data loading

A screenshot of a computer

AI-generated content may be incorrect.

### ImportMoviesFromJson

#### Define the JSON File Path



#### Read the JSON File



#### Deserialize the JSON Content



#### Check if the Movies List is Valid

A group of black and blue letters

AI-generated content may be incorrect.

#### Extract Movie IDs



#### Check for Existing Movies in the Database



#### Add and Save New Movies

A close-up of a white background

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

### ImportCinemasMoviesFromJson

#### Updated Cinema-Movies JSON Document

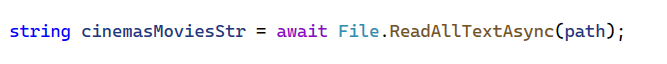
In the previous version of this workshop, we provided a **smaller** cinemaMovies.json file with fewer entries. However, to create a **more realistic and meaningful dataset**, we are now providing a **richer** version of the JSON file with **more cinema-movie relationships**

* **Replace the old cinemaMovies.json file** with the new, updated version provided in the workshop Resources.zip
* **Rebuild the application**

#### Define the JSON File Path



#### Read the JSON File



#### Deserialize the JSON Content



#### Check if the Data is Valid



#### Retrieve Existing Cinemas and Movies from the Database



#### Map JSON Data to CinemaMovie Entities

A screen shot of a computer program

AI-generated content may be incorrect.

#### Add and Save the New Relationships

A close up of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

### ImportTicketsFromXml

#### Updated Tickets XML Document

In the previous version of this workshop, we provided a **smaller** tickets.xml file with fewer entries. However, to create a **more realistic and meaningful dataset**, we are now providing a **richer** version of the XML file with **more tickets entries**

* **Replace the old tickets.xml file** with the new, updated version provided in the workshop Resources.zip
* **Rebuild the application**

#### Retrieve IDs from the Database

Before implementing ImportTicketsFromXml, we need to collect **CinemaId**, **MovieId**, and **UserId** from the database and update our **tickets.xml** file accordingly

* **Get Cinema IDs && Movies IDs from CinemasMovies**
  + **Assign random CinemaId/MovieId tuples to the tickets**

**A screenshot of a computer

AI-generated content may be incorrect.**

* **Get User IDs**
  + **Assign random UserIds to the tickets**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer code

AI-generated content may be incorrect.**

#### Defining the TicketDto Class

A screenshot of a computer program

AI-generated content may be incorrect.

#### Define the XML File Path



#### Read the XML File



#### Deserialize XML Content



#### Validate and Converted Data

A computer code with text

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

#### Insert and Save Tickets

A close-up of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

### ImportWatchListsFromXml

#### Adding the Method Call for Importing Watchlists

A screen shot of a computer

AI-generated content may be incorrect.

#### Run SQL Query to Get User IDs

A screenshot of a computer

AI-generated content may be incorrect.

#### Update Watchlists with Actual User IDs

A screenshot of a computer program

AI-generated content may be incorrect.

#### Create Required Dtos

A computer code with text

AI-generated content may be incorrect.

A screenshot of a computer code

AI-generated content may be incorrect.

#### Define the XML File Path



#### Read the XML File

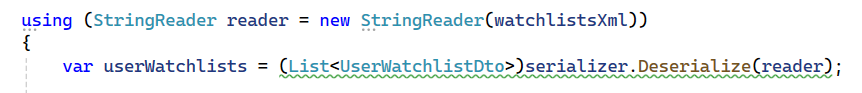


#### Deserialize XML Data

A close up of text

AI-generated content may be incorrect.

#### Deserialize and Iterate Over Users



#### Validate Users and Movies

A close up of text

AI-generated content may be incorrect.

A close up of a sign

AI-generated content may be incorrect.

#### Validate and Link Movies

A computer code with text

AI-generated content may be incorrect.

#### Check for Existing Entries

A computer screen shot of a program code

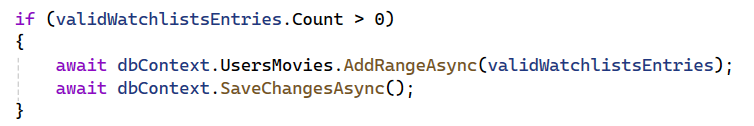
AI-generated content may be incorrect.

#### Create New Watchlist Entry

A screen shot of a computer program

AI-generated content may be incorrect.

#### Insert Valid Watchlist Entries



A screenshot of a computer

AI-generated content may be incorrect.

## Export Data

### Create an ExportResults Folder

A screenshot of a computer

AI-generated content may be incorrect.

### Update the Program.cs Methods calling Scope

A screen shot of a computer program

AI-generated content may be incorrect.

### ExportWatchlistsWithShowtimeCinemasToJson

#### Requirements

* Retrieve all users with their watchlists
* For each movie, check which cinemas are showing it
* If no cinema is showing the movie, display **"Not Available"**
* Save the data as a JSON file in the ExportResults folder

A screenshot of a computer program

AI-generated content may be incorrect.

### ExportUsersWithTheirTicketsToXml

#### Requirements

* Retrieve Users Who Have Purchased Tickets
* Each User Should Contain
  + **Username**
  + **Tickets List**
* Each Ticket Should Contain
  + **Movie Title**
  + **Cinema Details** → The cinema name and location where the movie is playing
  + **Price** → The cost of the ticket formatted as a decimal (e.g., "12.50")
* Export the Data into a Properly Formatted XML File

**A screenshot of a computer program

AI-generated content may be incorrect.**