Entity Framework Core — CRUD

Czas: 2 × 45 min | praca indywidualna | Wykład 2.13

Cel: Zbudujesz bibliotekę książek opartą na EF Core z SQLite: model, kontekst, migracje i pełne operacje CRUD z interfejsu.

Po zajęciach potrafisz:

- konfiguruje EF Core z dostawcą SQLite
- tworzy i stosuje migracje schematu bazy
- wykonuje operacje Add, Find, Where, Update, Remove i SaveChanges
- stosuje krótko żyjący kontekst w operacjach ViewModelu

Sprzęt i materiały

- komputer z Windows 11 i dostępem do internetu
- pendrive lub konto w repozytorium do zapisania projektu

Oprogramowanie

- **Visual Studio Community 2026 (obciążenie „Programowanie aplikacji klasycznych platformy .NET")** — darmowe dla uczniów i nauczycieli; w pracowni zainstalowane visualstudio.microsoft.com/pl/downloads
- **.NET 10 SDK (LTS)** — instalowany razem z Visual Studio; sprawdzenie: `dotnet --version` dotnet.microsoft.com/pl-pl/download/dotnet/10.0
- **Pakiet NuGet CommunityToolkit.Mvvm 8.4** — instalacja: Zarządzaj pakietami NuGet nuget.org/packages/CommunityToolkit.Mvvm
- **Pakiety Microsoft.EntityFrameworkCore.Sqlite i .Tools 10.x** — wersja zgodna z .NET 10 nuget.org/packages/Microsoft.EntityFrameworkCore.Sqlite
- **DB Browser for SQLite** — podgląd pliku bazy; wersja przenośna nie wymaga instalacji sqlitebrowser.org/dl

Bezpieczeństwo

- Zapisuj projekt w swoim folderze (np. `C:\Users\<login>\source\repos`), a na koniec zajęć skopiuj go na pendrive lub wyślij do repozytorium.
- Nie uruchamiaj programów i pakietów z nieznanych źródeł; pakiety NuGet instaluj tylko te, które wskazuje zadanie.
- Rób przerwy dla oczu (zasada 20-20-20) — praca z projektantem okien wymaga długiego patrzenia w ekran.

Przebieg zadania

1 Model i kontekst

Projekt WPF `Biblioteka`; zainstaluj pakiety EF Core 10 dla SQLite i Tools.

1. Klasa `Ksiazka`: Id, Tytul, Autor, Rok, Dostepna (na razie bez ceny).
2. `BibliotekaContext` z `DbSet<Ksiazka>` i `UseSqlite` (plik obok programu).
3. W `OnModelCreating`: maksymalne długości i wymagalność tekstów, 3 książki startowe `HasData`.

2 Migracje

Schemat bazy z kodu.

```
protected override void OnStartup(StartupEventArgs e)
{
    using (var db = new BibliotekaContext())
        db.Database.Migrate();
    base.OnStartup(e);
}
```

1. Konsola menedżera pakietów: `Add-Migration Poczatkowa`, obejrzyj wygenerowany plik w folderze Migrations.
2. W `App.OnStartup`: `db.Database.Migrate()` — baza tworzy się przy pierwszym uruchomieniu.
3. Dodaj do klasy właściwość `Cena`, potem `Add-Migration DodanaCena` i uruchom program — kolumna dochodzi bez utraty danych.
4. Sprawdź tabelę `__EFMigrationsHistory` w DB Browser for SQLite.

3 CRUD w ViewModelu

`KsiazkiViewModel` z CommunityToolkit.Mvvm.

1. `Wczytaj` — lista posortowana po tytule; `Dodaj` z formularza (po SaveChanges obiekt ma Id z bazy).
2. Edycja w DataGrid i komenda `ZapiszZmiany` (`UpdateRange` + `SaveChanges`, komunikat z liczbą zmian).
3. `Usun` zaznaczonej książki.
4. Każda operacja: `using var db = new BibliotekaContext();`.

4 Sprawdzenie

Dane trwałe.

1. Dodaj, zmień i usuń książki; zamknij i uruchom program ponownie — dane zostają.
2. Porównaj zawartość pliku bazy w DB Browser z widokiem w aplikacji.
3. Zatwierdź pracę w Git: folder Migrations TAK, plik .db NIE.

W sprawozdaniu umieść

☐ kod modelu i kontekstu

- ☐ fragment migracji DodanaCena z opisem
- ☐ kod ViewModelu
- ☐ zrzut aplikacji i tabeli w DB Browser

Pytania kontrolne

1. Co to jest DbContext i DbSet?
2. Po co są migracje i dlaczego trafiają do repozytorium?
3. Co robi SaveChanges?
4. Dlaczego kontekst tworzymy na czas jednej operacji?

Kryteria oceny

Kryterium	Punkty
Model, kontekst, HasData	4
Migracje (dwie) i Migrate przy starcie	5
CRUD w ViewModelu	7
Pytania	4
Razem	**20**

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).