Bazy Danych 2
20.06.2021

Projekt nr 4 - Przetwarzanie danych hierarchicznych

Michał Kuś

Wstęp

Celem projektu było stworzenie API, jego implementacji i wykorzystanie go w aplikacji konsolowej w języku C# i przetestowanie go za pomocą testów jednostkowych. API służy do tworzenia struktury hierarchicznej dla drzewa genealogicznego w bazie MS SQL wykorzystując interfejs ADO.NET.

Środowisko

W realizacji projektu wykorzystane zostały:

1. Visual Studio 2019
2. SQL Server Management Studio 18
3. Visual Studio 2008
4. SQL Server Management Studio 8

API zostało napisane wykorzystujące .NET Framework 3.5, testy wykorzystując .NET Framework 4.7.2.

Baza danych oraz odpowiednie tabele zostały stworzone w SSMS 8 i utrzymywane w SSMS 18.

Baza danych

Najpierw tworzy się bazę 'ProjectDB' wykorzystując skrypt **'ProjectDB_create.sql'**. Należy w nim **wskazać lokalizację na dysku** na umieszczenie pliku .mdf oraz zawierającego logi .ldf. Tradycyjnie w lokalizacji środowiska MSSQL w folderze DATA.

```
CREATE DATABASE [ProjectDB] ON  PRIMARY
( NAME = N'ProjectDB', FILENAME = N'D:\SQLSERVEREXPRESS\MSSQL15.SQLEXPRESS\MSSQL\DATA\ProjectDB.mdf' , SIZE = 3072KB , FILEGROWTH = 1024KB )
 LOG ON
( NAME = N'ProjectDB_log', FILENAME = N'D:\SQLSERVEREXPRESS\MSSQL15.SQLEXPRESS\MSSQL\DATA\ProjectDB_log.ldf' , SIZE = 1024KB , FILEGROWTH = 10%)
GO
ALTER DATABASE [ProjectDB] SET COMPATIBILITY LEVEL = 100
```

Rysunek 1. Fragment skryptu *ProjectDB_create.sql* zawierający lokalizację plików .mdf i .ldf na dysku

Następnie należy dodać do bazy XML Schema wykorzystując plik '**ProjectDB_addSchema.sql**'.

```
USE ProjectDB
GO
CREATE XML SCHEMA COLLECTION FamilyTreeSchemaCollection AS
N'<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Names" type="xs:string"/>
  <xs:element name="Surname" type="xs:string"/>
  <xs:element name="Born" type="xs:date"/>
  <xs:element name="Died" type="xs:date"/>
  <xs:element name="Mother">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute type="xs:int" name="id" use="required"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
```

Rysunek 2. Fragment skryptu *ProjectDB_addSchema.sql*

Ostatecznie za pomocą pliku '**ProjectDB_createTreesTable.sql**' tworzy się tabela FamilyTrees oraz zostaje wypełniona przykładowymi danymi. Ta tabela zawiera kolumny:

- ID - INT IDENTITY NOT NULL PRIMARY KEY
- FamilyName - NVARCHAR(50) NOT NULL
- FamilyTree - XML(dbo.FamilyTreeSchemaCollection)

```
USE ProjectDB
GO
CREATE TABLE FamilyTrees
(
    ID INT IDENTITY(1,1) NOT NULL PRIMARY KEY,
    FamilyName NVARCHAR(50) NOT NULL,
    FamilyTree XML(dbo.FamilyTreeSchemaCollection)
);
```

Rysunek 3. Fragment skryptu *ProjectDB_createTreesTable.sql* zawierający deklarację tabeli *FamilyTrees*

Modele

Do poprawnego działania API konieczne są dwa modele danych: *FamilyTreeModel* i *PersonModel*, reprezentujące dane pobierane z bazy w postaci rekordu lub elementu XML.

PersonModel

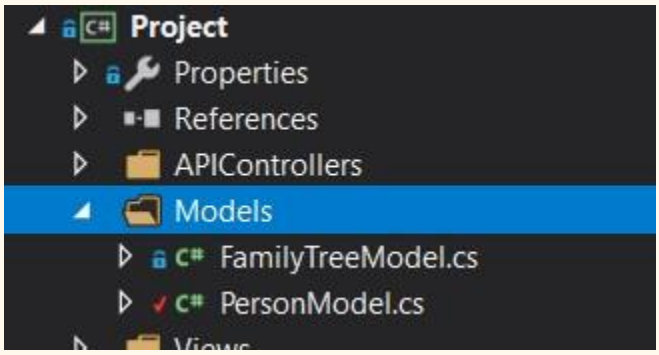
- int id - id osoby
- string names - imię/imiona
- string surname - nazwisko
- string born - data urodzenia
- string died - data śmierci
- int mother - id matki
- int father - id ojca
- XElement node - węzeł XML zawierający dane osoby

FamilyTreeModel

- int id - id rodziny(rekordu w bazie)
- string name - nazwa rodziny
- XElement tree - drzewo XML zawierające członków rodziny

Modele zawierają odpowiednie konstruktory pozwalające na stworzenie ich obiektów.

Znajdują się w folderze **Models**.



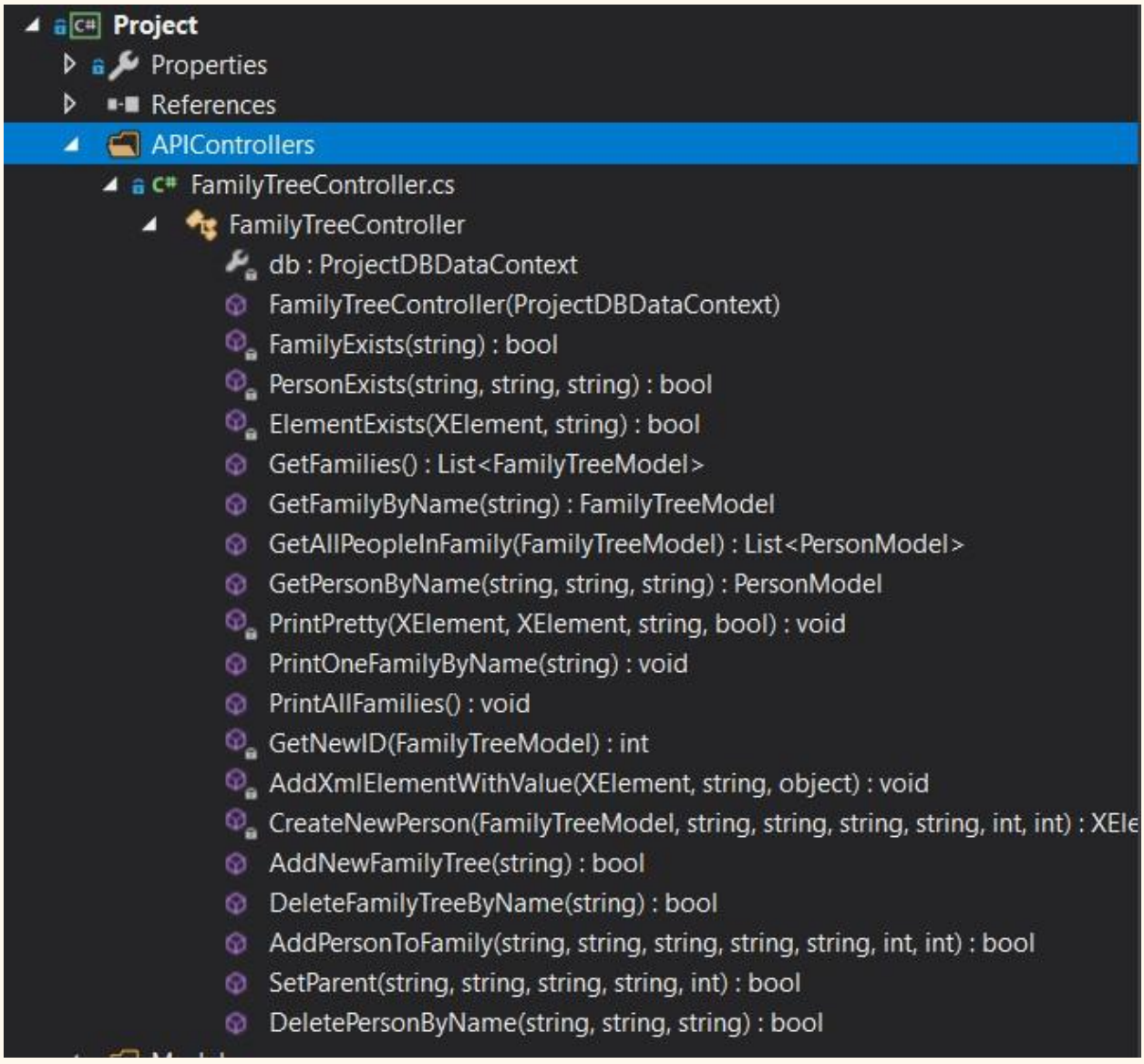
Rysunek 4. Lokalizacja modeli w hierarchii rozwiązania

API

API zostało zbudowane w klasie FamilyTreeController, przyjmuje ono DataContext z interfejsu Linq To Sql i wykorzystuje modele FamilyTreeModel i PersonModel do interfejsowania danych z bazy. Ma ono następujące elementy:

- Konstruktor przyjmujący ProjectDBDataContext i pozwalający na połączenie z bazą
- Metody prywatne sprawdzające czy elementy, rodziny lub osoby istnieją

- Metody prywatne tworzące XML nowej osoby, dodające elementy XML i wyszukujące nowe id osoby
- Metoda prywatna służąca do wypisywania do konsoli osób w postaci drzewa
- Metody służące do zwracania osób lub rodzin w postaci modeli
- Metody służące do wypisywania rodzin do konsoli
- Metoda dodająca nowe drzewo genealogiczne
- Metoda usuwająca drzewo genealogiczne
- Metoda dodająca osobę do rodziny
- Metoda usuwająca osobę z rodziny
- Metoda ustawiająca rodzica osoby

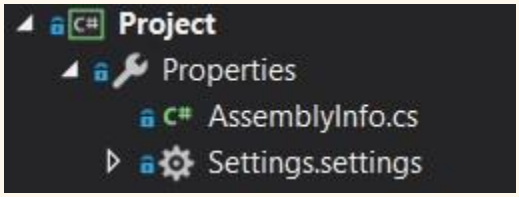


Rysunek 5. Metody udostępnione przez API

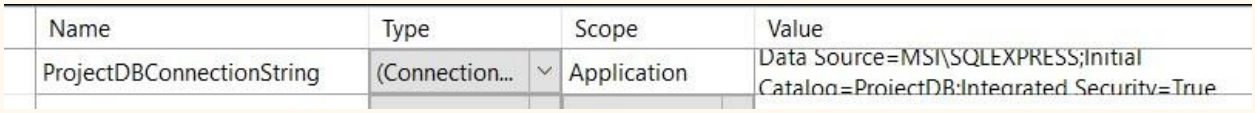
Praca z API

Główny plik zawierający API to 'FamilyTreeController.cs' znajdujący się w folderze APIControllers.

Zanim rozpocznie się pracę z API **należy wykonać skrypty budujące bazę danych**, a następnie w ustawieniach projektu (Solution Explorer => Project => Properties => Settings.settings) **ustawić wartość DataSource** w ProjectDBConnectionString na dostępne **środowisko MS SQL**. Działanie można przetestować w dostarczonej aplikacji konsolowej.



Rysunek 6. Lokalizacja pliku Settings.settings w hierarchii rozwiązania



Rysunek 7. Zmienna *ProjectDBConnectionString* i jej wartość *DataSource*

Do rozpoczęcia pracy należy najpierw utworzyć obiekt klasy ProjectDBDataContext udostępniony przez zawarty plik '**ProjectDB.dbml**'.

Następnie trzeba stworzyć obiekt klasy FamilyTreeController za pomocą konstruktora dostarczając obiekt klasy ProjectDBDataContext.

Metody modyfikujące bazę

1. Żeby stworzyć nowy rekord w bazie zawierający drzewo genealogiczne trzeba wywołać metodę *AddNewFamilyTree(string):bool* przyjmującą:
   - *string familyName* - nazwę rodziny

   Metoda zwraca *bool* zawierający *true* jeśli rekord dodano poprawnie, w przeciwnym wypadku *false*.

   Metoda nie doda drzewa jeśli w bazie istnieje już rodzina o tej samej nazwie.

2. Aby usunąć rekord drzewa z bazy należy wywołać metodę *DeleteFamilyTree(string):bool* przyjmującą:
   - *string familyName* - nazwę rodziny

Metoda zwraca *bool* zawierający *true* jeśli rekord usunięto poprawnie, w przeciwnym wypadku *false*.

3. Aby dodać osobę do drzewa należy wywołać metodę *AddPersonToFamily(string, string, string, string, string, int, int):bool* przyjmującą:
    - *string targetFamilyName* - nazwę rodziny
    - *string names* - imię/imiona osoby
    - *string surname* - nazwisko osoby
    - *string born* - datę urodzenia
    - *string died* - datę śmierci
    - *int mother* - id innej osoby która jest matką tej osoby
    - *int father* - id innej osoby która jest ojcem tej osoby

    Metoda zwraca *bool* zawierający *true* jeśli rekord dodano poprawnie, w przeciwnym wypadku *false*.

    Konwencja daty przyjęta w aplikacji konsolowej to yyyy-mm-dd, ale ponieważ pola *born* i *died* to są typu *string* jakakolwiek przyjęta konwencja jest akceptowalna.

    Metoda nie doda osoby jeśli w tej rodzinie istnieje już osoba o tym samym imieniu i nazwisku.

4. Aby usunąć osobę z drzewa należy wywołać metodę *DeletePersonByName(string, string, string):bool* przyjmującą:
    - *string targetFamilyName* - nazwę rodziny
    - *string names* - imię/imiona osoby
    - *string surname* - nazwisko osoby

    Metoda zwraca *bool* zawierający *true* jeśli osobę usunięto poprawnie, w przeciwnym wypadku *false*.

5. Aby ustawić rodzica istniejącej osoby należy wywołać metodę *SetParent(string, string, string, string, string, int, int):bool* przyjmującą:
    - *string targetFamilyName* - nazwę rodziny
    - *string names* - imię/imiona osoby
    - *string surname* - nazwisko osoby
    - *string whichParent* - “Mother”/”Father”, pole decydujące którego rodzica ustawia metoda
    - *int parentID* - id innej osoby którą ustawiamy rodzicem tej osoby

Metoda zwraca *bool* zawierający *true* jeśli rekord zmieniono poprawnie, w przeciwnym wypadku *false*.

Metody zwracające rekordy

1. Aby pobrać listę wszystkich rodzin w bazie należy wywołać metodę *GetFamilies():List<FamilyTreeModel>* .

   Metoda zwraca *List<FamilyTreeModel>* zawierający listę rekordów rodzin z bazy zamienionych na model.

2. Aby pobrać listę wszystkich osób w rodzinie z bazy należy wywołać metodę *GetAllPeopleInFamily(FamilyTreeModel):List<PersonModel>* przyjmującą:
   - *FamilyTreeModel family* - model rodziny

   Metoda zwraca *List<PersonModel>* zawierający listę osób z rodziny zamienionych na model.

3. Aby pobrać osobę w rodzinie z bazy należy wywołać metodę *GetPersonByName(string, string, string):PersonModel* przyjmującą:
   - *string targetFamilyName* - nazwa rodziny
   - *string names* - imię/imiona osoby
   - *string surname* - nazwisko osoby

   Metoda zwraca *PersonModel* zawierający dane osoby z rodziny w postaci modelu.

4. Aby pobrać rodzinę z bazy należy wywołać metodę *GetFamilyByName(string):FamilyTreeModel* przyjmującą:
   - *string targetFamilyName* - nazwa rodziny

   Metoda zwraca *FamilyTreeModel* zawierający dane rodziny w postaci modelu.

Metody tworzące raporty

1. Żeby wypisać wszystkie rodziny do konsoli należy skorzystać z metody *PrintAllFamilies():void.*

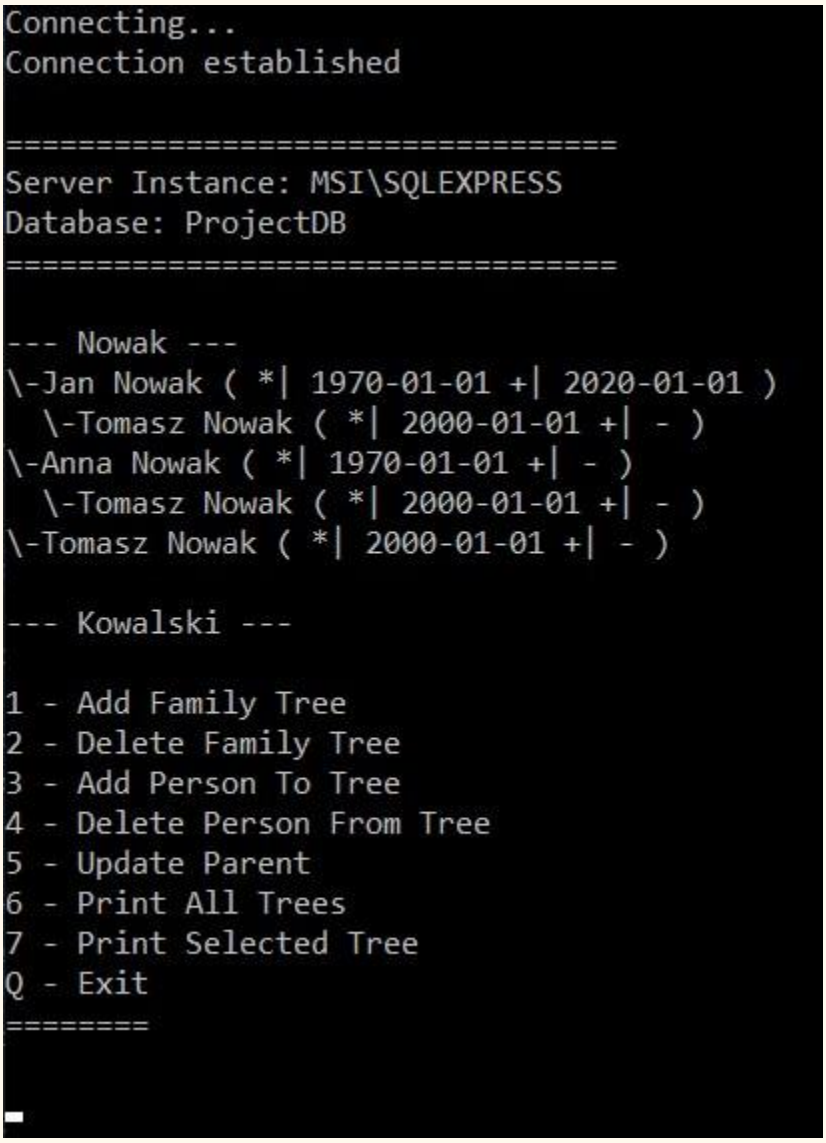
   Wyświetla ona rodziny w postaci drzew hierarchicznych.

2. Żeby wypisać jedną rodzinę do konsoli należy skorzystać z metody *PrintOneFamilyByName(string):void* przyjmującej:
    - *string familyName* - nazwę rodziny

    Wyświetla ona rodzinę w postaci drzewa hierarchicznego.

Aplikacja konsolowa

Działanie API zostało przedstawione za pomocą aplikacji konsolowej z interfejsem użytkownika, która zostanie **włączona po uruchomieniu projektu**. Po połączeniu z bazą danych aplikacja wyświetla komunikat o połączeniu, a następnie uruchamia główną pętlę aplikacji pozwalającą na skorzystanie z funkcjonalności zapewnionych przez API. Interfejs aplikacji znajduje się w folderze **Views** w pliku **UserInterface.cs.**

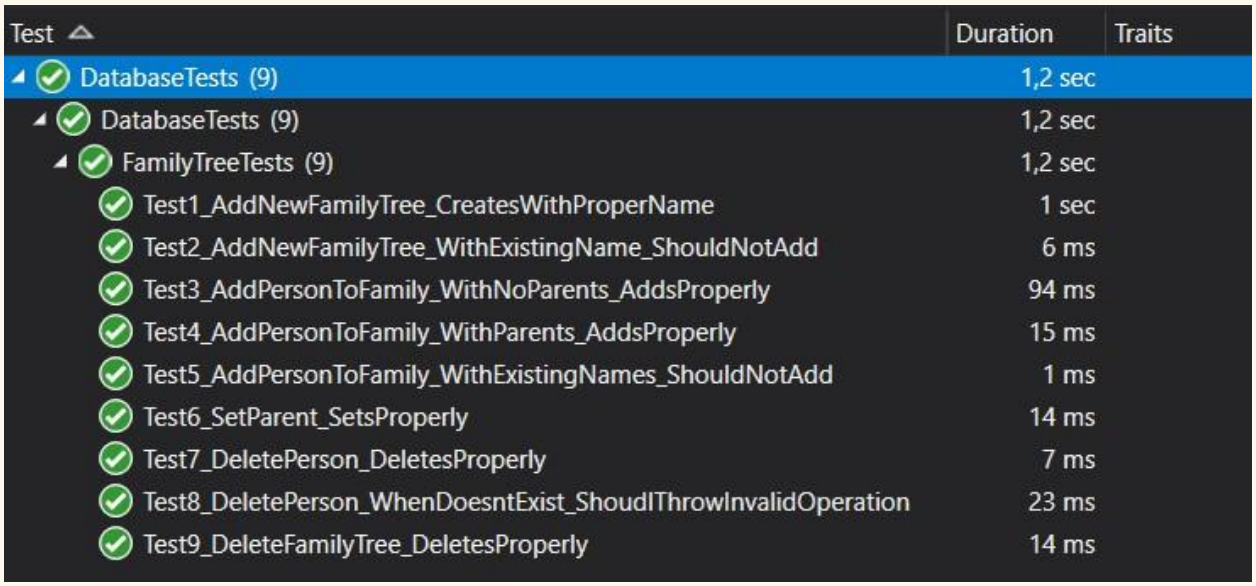


Rysunek 8. Interfejs konsolowy po uruchomieniu programu

Testy jednostkowe

Testy jednostkowe zostały wykonane z pomocą framework'u MSTest i platformy .NET 4.7.2. Sprawdzają funkcjonalności klasy FamilyTreeController:

- Tworzenie rodzin
- Usuwanie rodzin
- Dodawanie osób
- Usuwanie osób
- Ustawianie rodzica



Rysunek 8. Test Explorer po przeprowadzeniu testów jednostkowych

Podsumowanie

API FamilyTreeController zapewnia wymagane założeniami projektu funkcjonalności:

- Tworzenie struktury hierarchicznej dla drzewa genealogicznego
- Wykorzystuje typ XML w SQL Server
- Dodanie drzewa
- Dodawanie węzłów
- Usuwanie węzłów
- Oraz tworzenie raportów

A także:

- Usuwanie drzewa
- Ustawianie wartości rodzica

Wykorzystanie API zostało przedstawione w ramach aplikacji konsolowej odbierającej wejście od użytkownika.

Zrealizowano testy jednostkowe sprawdzające działanie API.

Kod klasy FamilyTreeController jest opisany dokumentacją zgodnie z konwencją języka C# w formacie XML.

Wnioski

Możliwości zapewniane przez platformę .NET pozwalają na sprawną współpracę z bazą MS SQL, a w szczególności z typem XML. Wykorzystanie wyrażeń LINQ umożliwia przeniesienie znacznej części logiki ze strony bazy na stronę platformy .NET.

API można by rozszerzyć o funkcjonalności związane z obsługą małżeństw, a także dodatkowe dane członków rodziny.

Literatura

https://docs.microsoft.com/en-us/

https://stackoverflow.com/