Dokumentacja projektu

Projekt numer 6: Przetwarzanie własnych typów danych CLR UDT

Hleb Shypula

BD2 2021

1. Opis problemu

Celem projektu było opracowanie API oraz jego implementacja obsługująca zestaw typów własnych UDT w technologii CLR do obsługi typów złożonych oraz stworzenie aplikacji konsolowej wykorzystującej złożone UDT. Ma ona zawierać niezbędne testy jednostkowe, a także udostępniać opcję dodawania rekordów, wyszukiwania oraz tworzenia raportów.

Jako typy złożone zostały wybrane typy opisujące obiekty, które będą wykorzystywane dla przechowywania danych kierowców firmy – email, prawo jazdy, numer samochodu, kolor samochodu (jako RGB), opis ruchu (jako funkcja kwadratowa) oraz naruszenie.

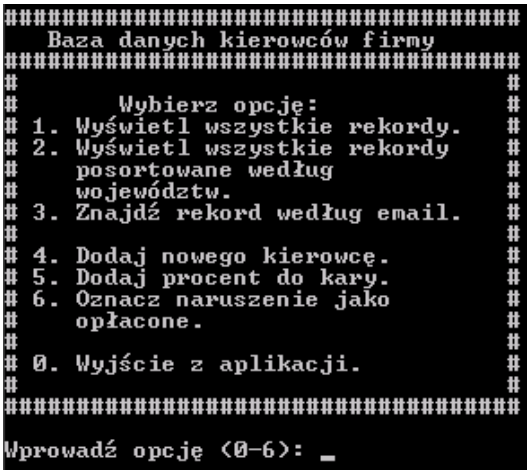
2. Opis funkcjonalności udostępnionej przez API

Stworzona aplikacja jest aplikacją konsolową – użytkownik odpowiednio porusza się po niej dzięki odpowiedniej funkcjonalności, a także instrukcjach widniejących bezpośrednio w konsoli.

Aplikacja działa na jedynej tabeli Driver, która zawiera wszystkie typy złożone. Łatwo się z tym zapoznać w pliku **creation_script.sql**.

Poruszanie się po aplikacji jest intuicyjne dzięki wyboru odpowiednich cyfr oznaczających przejście do danego okna.

Użytkownik ma możliwość dodawania oraz przeszukiwania odpowiednich rekordów bazy kierowców firmy oraz generowania odpowiednich raportów z nimi związanych.



Menu główne

Aby wybrać jedną z opcji, należy wpisać odpowiednią cyfrę, a następniezatwierdzić instrukcję klawiszem Enter.

Baza została wypełniona przykładowymi danymi, które są zawarte w pliku *creation_script.sql*.

3. Opis typów danych oraz metod (funkcji) udostępnionych w ramach API, szczegóły implementacji.

Własne typy UDT (MyEmail, QuadraticFunction, Licence, Violation, RGBColor, CarPlate) oraz odpowiednie testy jednostkowe są zaimplementowane w projekcie **Project**. Natomiast aplikacja terminałowa jest zaimplementowana w projekcie **TerminalProject**.

W projekcie TerminalProject jest zaimplementowana klasa Program, która zawiera statyczną metodę main. W tej metodzie za pomocą ADO.NET oraz operatora switch jest zrealizowana cała funkcjonalność aplikacji.

Opis klas reprezentujących UDT:

Klasy te implementują interfejsy: Inullable oraz IbinarySerialize (W przypadku klas zawierających typy złożone – String, DateTime).

Każda klasa zawiera pola na przechowywanie danych struktury oraz pole *bool m_Null* (flaga, czy dana referencja jest niezainicjalizowana).

Także każda klasa zawiera:
1. Gettery i **settery** dla każdego pola;
2. Metodę **ToString()** – zwraca w postaci stringa reprezentację danego obiektu wraz z opisem jego wartości;
3. Metodę **Parse(SqlString s)** – zwraca nowo utworzony obiekt z polecenia podanego jako argument wywołania metody;
4. Metodę **Validate()** – metoda, która waliduje poprawność wprowadzonych danych;
5. Metodę **Null** - metoda zwracająca nowy obiekt, którego referencja jest nullem;
W przypadku klas zawierających typy złożone:
6. Metodę **Write(System.IO.BinaryWriter w)** – metoda serializująca obiekt;
7. Metodę **Read(System.IO.BinaryReader r)** – metoda deserializująca obiekt;

Oprócz powyższych podstawowych pól i metod klasy te posiadają własne metody na pewne obliczenia na podstawie wymaganej logiki biznesowej. Ich opis oraz działanie jest trywialne.

4. Przeprowadzone testy jednostkowe

Zostały przygotowane testy jednostkowe, które testują odpowiednie metody wszystkich klas, a także testy SQL, które widnieją w pliku **Test.sql**.

Łącznie wykonano **32 testy** jednostkowe dla metod klas, zaimplementowanych w języku C#, sprawdzających poprawność wyników, które zwracają metody.

Wszystkie testy klas znajdują się w projekcie **Project** w podprojekcie **TestProject**, pliki te rozpoczynają się od przedrostka *Test*.

Test run completed Results: 32/32 passed; Item(s) checked: 0

Result	Test Name	Project
Passed	TestRGBColorR	TestProject
Passed	TestLicenceCategory	TestProject
Passed	TestViolationIsPaid	TestProject
Passed	TestLicenceDate	TestProject
Passed	TestViolationToString	TestProject
Passed	TestQFCountMaxMin	TestProject
Passed	TestQFGetA	TestProject
Passed	TestViolationSurcharge	TestProject
Passed	TestRGBColorG	TestProject
Passed	TestQFToString	TestProject
Passed	TestMyEmailHost	TestProject
Passed	TestViolationSetIsPaid	TestProject
Passed	TestRGBColorToString	TestProject
Passed	TestCarPlateProvince	TestProject
Passed	TestQFGetB	TestProject
Passed	TestMyEmailSameUser	TestProject
Passed	TestCarPlateNumber	TestProject
Passed	TestQFGetC	TestProject
Passed	TestMyEmailUsername	TestProject
Passed	TestMyEmailSameUser	TestProject
Passed	TestLicenceSetDate	TestProject
Passed	TestCarPlateToString	TestProject
Passed	TestMyEmailChangeHos	TestProject
Passed	TestRGBColorIsEquals	TestProject
Passed	TestQFIsEquals	TestProject
Passed	TestViolationDescription	TestProject
Passed	TestMyEmailToString	TestProject
Passed	TestLicenceToString	TestProject
Passed	TestViolationToPay	TestProject
Passed	TestViolationDate	TestProject
Passed	TestRGBColorB	TestProject
Passed	TestRGBColorAdd	TestProject

Screenshot przeprowadzonych testów

5. Kod źródłowy

Cały kod źrodłowy oraz skrypt tworzenia bazy danych (*creation_script.sql*) znajduje się w archiwum oraz pod linkiem: https://github.com/hlebshypulahub/BD2-Project-WFiIS-2021

6. Podsumowanie i wnioski

Aplikacja została napisana w języku C#, umożliwia użytkownikowi operowanie na złożonych typach UDT poprzez oferowane API, dzięki któremu wchodzi on w interakcję z programem oraz bazą danych.

Opracowane API umożliwia wprowadzanie danych do bazy, wyszukiwanie rekordów a także generowanie raportów.

W projekcie zostały zastosowane testy jednostkowe, które sprawdzają poprawność implementacji wszystkich metod.

Ogólnie podsumowując pracę nad projektem, można powiedzieć, że UDT jest przyjemną funkcjonalnością i nie wymaga wysokiego poziomu wiedzy na opanowanie, ponieważ opiera się na podstawy programowania obiektowego oraz język bliski składniowo do Java.

Mogę powiedzieć, że wiedza otrzymana w czasie robienia projektu jest bardzo przydatna i może być zastosowana przy napotkaniu odpowiednich wymagań biznesowych w przyszłości.

7. Bibliografia

https://newton.fis.agh.edu.pl/~antek/read_pdf.php?file=BD2_L09_CLR.pdf

https://stackoverflow.com/

https://docs.microsoft.com/