

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 wyborowi 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.

```
#####
Baza danych kierowców firmy
#####
#
#      Wybierz opcję:
# 1. Wyświetl wszystkie rekordy.
# 2. Wyświetl wszystkie rekordy
#    posortowane według
#    województw.
# 3. Znajdź rekord według email.
#
# 4. Dodaj nowego kierowcę.
# 5. Dodaj procent do kary.
# 6. Oznacz naruszenie jako
#    opłacone.
#
# 0. Wyjście z aplikacji.
#
#####
Wprowadź opcję <0-6>: _
```

Menu główne

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

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

```
#####
Baza danych kierowców firmy
#####
#
#      Wybierz opcję:
# 1. Wyświetl wszystkie rekordy.
# 2. Wyświetl wszystkie rekordy
#    posortowane według
#    województw.
# 3. Znajdź rekord według email.
#
# 4. Dodaj nowego kierowcę.
# 5. Dodaj procent do kary.
# 6. Oznacz naruszenie jako
#    opłacone.
#
# 0. Wyjście z aplikacji.
#
#####
Wprowadź opcję <0-6>: 1
#####
id:1 email:alancarr@gmail.com prawo jazdy:153,198,222 kolor:UM542E opis ruchu:4.3e+2 - 2.85e+2 naruszenie:
id:2 email:ivan.urgent@yandex.ru Kategorie: D Data wydania: 2013-12-03 (249,20,20) KR012FL 17.55e+2 + 3.2e+ 0.06i Prowadzenie pojazdu bez uprawnienia; Czas: 2021-07-05 14:55:07; Do zapłaty: 120; Opłacone: Nie!
id:3 email:kapoliziol@net.pl Kategorie: G1 Data wydania: 2017-03-27 (18,224,19) SL01384 2.3e+2 - 0.66e+ 13.9i Uszkodzenie drogi publicznej; Czas: 2021-06-12 19:33:15; Do zapłaty: 60.5; Opłacone: Nie!
id:4 email:luksim@tut.ru Kategorie: B Data wydania: 1935-10-13 (154,4,23) L015124 23.3e+2 + 1.4e+ 10.42i Tamowanie lub utrudnianie ruchu; Czas: 2021-06-13 15:13:12; Do zapłaty: 230; Opłacone: Tak!
id:5 email:karyac@bqgh.pl Kategorie: NM Data wydania: 2020-06-15 (64,54,133) P04P054114,4e+2 + 0.004e+ -0.6i Prowadzenie pojazdu w stanie po użyciu alkoholu; Czas: 2021-12-03 04:22:14; Do zapłaty: 330; Opłacone: Nie!
id:6 email:piet@bdu.ru Kategorie: B Data wydania: 2007-11-12 (24,176,83) S063391 23.3e+2 + 1.4e+ 10.42i Prowadzenie nieodpłatnego pojazdu; Czas: 2020-03-27 12:00:53; Do zapłaty: 100; Opłacone: Tak!
id:7 email:alonsok@pml.net Kategorie: G Data wydania: 2005-04-01 (12,0,255) F23910K14,33e+2 + -1.2e+ -2.2i Samowolne uszkodzenie znaków; Czas: 2020-10-13 10:19:45; Do zapłaty: 80; Opłacone: Nie!
id:8 email:king@play.pl Kategorie: D Data wydania: 2024-02-27 (255,255,0) M12781 12.4e+2 + 0e+ 2i Prowadzenie pojazdu bez wymaganych dokumentów; Czas: 2021-02-28 23:01:16; Do zapłaty: 140; Opłacone: Nie!
id:9 email:katenene@gmail.com Kategorie: B Data wydania: 1999-04-12 (1,1,1) UM5622L 1e+2 - 2e+ -4.333i Nieudzielenie pomocy ofierze wypadku; Czas: 2021-03-09 13:31:14; Do zapłaty: 950.5; Opłacone: Tak!
id:10 email:pahin.xavi@guo.es Kategorie: D Data wydania: 2004-05-11 (255,12,233) L0280701 0.001e+2 + 12.2e+ -3i Niestosowanie się do znaku w ruchu drogowym; Czas: 2020-05-27 19:17:44; Do zapłaty: 320; Opłacone: Nie!
id:12 email:nalyk@tdlux.org Kategorie: B Data wydania: 2017-11-30 (109,109,0) K298F32 12.2e+2 + 4e+ 3.44i Przekroczenie limit prędkości; Czas: 2021-12-06 14:44:17; Do zapłaty: 110; Opłacone: Nie!
```

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 terminalowa 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
<input type="checkbox"/>	Passed	TestRGBColorR	TestProject
<input type="checkbox"/>	Passed	TestLicenceCategory	TestProject
<input type="checkbox"/>	Passed	TestViolationIsPaid	TestProject
<input type="checkbox"/>	Passed	TestLicenceDate	TestProject
<input type="checkbox"/>	Passed	TestViolationToString	TestProject
<input type="checkbox"/>	Passed	TestQFCountMaxMin	TestProject
<input type="checkbox"/>	Passed	TestQFGetA	TestProject
<input type="checkbox"/>	Passed	TestViolationSurcharge	TestProject
<input type="checkbox"/>	Passed	TestRGBColorG	TestProject
<input type="checkbox"/>	Passed	TestQFToString	TestProject
<input type="checkbox"/>	Passed	TestMyEmailHost	TestProject
<input type="checkbox"/>	Passed	TestViolationSetIsPaid	TestProject
<input type="checkbox"/>	Passed	TestRGBColorToString	TestProject
<input type="checkbox"/>	Passed	TestCarPlateProvince	TestProject
<input type="checkbox"/>	Passed	TestQFGetB	TestProject
<input type="checkbox"/>	Passed	TestMyEmailSameUser	TestProject
<input type="checkbox"/>	Passed	TestCarPlateNumber	TestProject
<input type="checkbox"/>	Passed	TestQFGetC	TestProject
<input type="checkbox"/>	Passed	TestMyEmailUsername	TestProject
<input type="checkbox"/>	Passed	TestMyEmailSameUser	TestProject
<input type="checkbox"/>	Passed	TestLicenceSetDate	TestProject
<input type="checkbox"/>	Passed	TestCarPlateToString	TestProject
<input type="checkbox"/>	Passed	TestMyEmailChangeHos	TestProject
<input type="checkbox"/>	Passed	TestRGBColorIsEquals	TestProject
<input type="checkbox"/>	Passed	TestQFIsEquals	TestProject
<input type="checkbox"/>	Passed	TestViolationDescription	TestProject
<input type="checkbox"/>	Passed	TestMyEmailToString	TestProject
<input type="checkbox"/>	Passed	TestLicenceToString	TestProject
<input type="checkbox"/>	Passed	TestViolationToPay	TestProject
<input type="checkbox"/>	Passed	TestViolationDate	TestProject
<input type="checkbox"/>	Passed	TestRGBColorB	TestProject
<input type="checkbox"/>	Passed	TestRGBColorAdd	TestProject

Screenshot przeprowadzonych testów

5. Kod źródłowy

Cały kod źródłowy 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/>