Dokumentacja

Przetwarzanie własnych typów danych
CLR UDT
Projekt z przedmiotu Bazy Danych II

Jakub Urbański

IS
WFiIS
AGH
14.06.2023 r.

1 Opis problemu

W ramach projektu mamy atrapę sklepu internetowego sprzedającego profesjonalne piłki meczowe z najlepszych piłkaskich lig świata. Typy UDT CLR wykorzystane zostały do przechowania różnego rodzaju danych osobowe oraz pełnych informacje o produktach.

2 Funkcjonalność

Aplikacja pozwala na dodanie piłek do koszyka, w koszyku na zmianę ich ilości oraz dokonanie zakupu poprzez dodanie danych karty płatniczej. Możliwa jest rejestracja nowych użytkowników. Zalogowany użytkownik ma możliwość stałego przypisania karty płatniczej do swojego konta, aby nie musieć podawać ich podczas każdych zakupów, każdy użytkownik ma również przypisane dane osobowe.

3 API, GUI

Stworzona aplikacja jest typu Server-Side Rendering - napisana we Flasku, typy oczywiście stworzone zostały w języku C# z użyciem Visual Studio, a tabele używane w aplikacji stworzone zostały w SQL Server. Aplikacja łączy się z Bazą Danych UDT_CLR i w niej wykonuje wszystkie operacje.

4 Funkcjonalności

4.1 Typy UDT CLR

Stworzone w technologii C#

4.1.1 `Adres`

Typ o polach:

- ulica [string]
- numer domu (wiem, że brakuje opcjonalnego numeru mieszkania ale trochę zabrakło mi czasu na przemyślenie działania wtedy tego typu) [Int32]
- miasto [string]
- kod pocztowy [string]
- kraj [string]

4.1.2 `DaneOsobowe`

Typ o polach:

- imię [string]
- nazwisko [string]
- data urodzenia [DateTime]
- numer telefonu [string]
- adres e-mail [string]

.

4.1.3 `Konto`

Typ o polach:

- login [string]
- hasło (wiem, że to nie jest absolutnie bezpieczne rozwiązanie - jest ono stricte stworzone po to aby bardziej pobawić się typami UDT) [string]
- adres e-mail [string]

4.1.4 `Produkt`

Typ o polach:

- producent [string]
- model [string]
- cena [float]
- dostępność [Int32]

4.1.5 `Transakcja`

Typ o polach:

- kwota [float]
- data transakcji (za późno - czyli jak w bazie była ju pokaźna liczba danych - zauważyłem, że stworzyłem ją bez godziny) [DateTime]

.

4.1.6 `Karta`

Typ o polach:

- imię [string]
- nazwisko [string]
- numer karty [string]
- data ważności [DateTime]
- numer CVV [string]

4.2 Tabele

Stworzone SQL SERVER

4.2.1 `Konta`

Tabela o kolumnach:

- Id [int]
- Dane [DaneOsobowe]
- Konto [Konto]

4.2.2 `Uzytkownicy`

Tabela o kolumnach:

- Id [int]
- Dane [DaneOsobowe]
- Adres [Adres]

4.2.3 `Transakcje`

Tabela o kolumnach:

- Id [int]
- Transakcja [Transakcja]
- IdKonta [int]
- Historia [nvarchar(4)]

4.2.4 `Produkty`

Tabela o kolumnach:

- Id [int]
- Produkt [Produkt]

4.2.5 `Karty`

Tabela o kolumnach:

- Karta [Karta]
- Konto [Konto]

4.3 Funkcje pomocnicze

Wszystkie w pliku `methods.py`

4.3.1 `ustaw_koszyk_default`

Funkcja typu void zerująca koszyk.

4.3.2 `selects`

Funkcja wywołująca kilkukrotnie powtarzające się po sobie i powiązane zapytania 'SELECT' i zwracająca odpowiednie listy.

4.3.3 `sprawdz_format_stringa`

Funkcja przyjmująca za paramatr string i sprawdzająca czy pasuje do wzoru na datę ważności karty.

4.4 Endpointy

U góry widoczny jest dynamiczny pasek nawigacyjny. Dla użytkownika niezalogowanego dostępne są:

- 'Produkty' (strona główna)
- 'Koszyk'
- 'Rejestracja'
- 'Logowanie

Dla użytkownika zalogowanego dostępne są:

- 'Produkty'
- 'Moje dane'

- 'Koszyk'
- 'Wylogowanie'
- wyświetlenie loginu zalogowanego użytkownika

4.4.1 *home_page*

Strona główna - wyświetla sprzedawane produkty oraz ich dostępność.

4.4.2 `/koszyk`

Najbardziej rozbudowana strona wyświetlająca różne informacje - w zależności od bycia zalogowanym, przypisanej do użytkownika karty w bazie, a także ilości produktów w trakcie zakupu.

4.4.3 `/koszyk/potwierdzenie`

Strona informująca o potwierdzeniu i zakończeniu zakupu. Przyjmuje dwa parametry pomocnicze - id_konta (id zalogowanego użytkownika - w przeciwnym razie 0) oraz kwota. Oprócz informacji robi 'UPDATE' do tabeli `Produkty` zmniejszając liczbę dostępnych piłek odpowiednich modeli, a także 'INSERT' do tabeli `Transakcje`.

4.4.4 `/dodaj_do_koszyka`

Endpoint 'przepływowy' służący jedynie do dodania modelu piłki do koszyka. Przyjmuje parametr pomocniczy - produkt_id (id dodawanego do koszyka produktu).

4.4.5 `/zmien_liczbe`

Endpoint 'przepływowy' służący jedynie do zamiany liczby zakupowanych piłek każdego modelu. Przyjmuje parametr pomocniczy - produkt_id (id edytowanego w koszyku produktu).

4.4.6 `/rejestracja`

Strona z prostym formularzem do rejestracji nowego użytkownika. Robi 'INSERTy' do tabel `Uzytkownicy` oraz `Konta`.

4.4.7 `/logowanie`

Strona z prostym formularzem do logowania użytkownika.

4.4.8 `/wylogowanie`

Endpoint 'przepływowy' służący jedynie do wylogowania użytkownika. Zeruje koszyk.

4.4.9 `/dane`

Kolejna bardzo rozbudowana strona służąca jednak tylko do wyświetlania danych, opconalnie do przekierowania do dodania do swojego konta karty (jeśli użytkownik nie ma jej w bazie).

4.4.10 `/dodaj_kart`

Strona z prostym formularzem do dodania karty. Przyjmuje parametr pomocniczy - redirect_type (czyli informację skąd została wywołana i w które miejsce powinna później przekierować użytkownika).

5 Podsumowanie

Projekt wykorzystuje możliwości C# do tworzenia i przechowywania w bazie własnych typów CLR. Niestety nie udało mi się obsłużyć wszystkich błędnych wykorzystań własnych typów złożonych UDT, dlatego mogą pojawiać się błedy w przypadku błędnego wprowadzenia niektórych danych.