EntityFramework, LINQ2Entities – laboratorium

I. Code First

a. Stwórz projekt typu ConsoleApplication

b. Dodaj klase Category z property int CategoryID, String Name oraz listą produktow List<Product> Products

c. Dodaj klasę Product z polami int ProductID, string Name, int UnitsInStock, int CategoryID

d. Zainstaluj EF w wersji 6.1.3.0 (View -> Other Windows -> Package Manager Console I następnie install-package entityframework -version 6.1.3.0)

e. Stwórz klasę ProdContext dziedziczącą po DbContext zawierającą zbiory (DbSet) kategorii oraz produktow

f. W Mainie

   i. poproś użytkownika o podanie nazwy kategorii

   ii. zainstancjonuj kategorie o podanej nazwie

   iii. dodaj zainstancjonowany obiekt do kontekstowej kolekcji kategorii

   iv. zapisz zmiany na kontekscie

g. Załóż brake'a na linijce zapisu zmian w kontekście

h. Uruchom Management studio łącząc się z serwerem (localdb)\v11.0. Prześledź jakie bazy są na serwerze dostępne. W szczególności zweryfikuj czy istnieje baza o nazwie takiej jak stworzona przez Ciebie solucja. Jesli tak - jaką ma strukturę?

i. Uruchom SQL Profiler. Rozpocznij tracowanie zapytań dla serwera (localdb)\v11.0

j. Kontynuując krokowo realizację programu prześledź wywołania poleceń sql'owych

k. Przejdź ponownie do management studio i prześledź ponownie jakie bazy są na serwerze dostepne. W szczególności zweryfikuj czy istnieje baza o nazwie takiej jak stworzona przez Ciebie solucja. Jeśli tak - jaką ma strukturę?

l. Korzystając z "Server explorera" lub Management Studio prześledź strukturę bazy danych i sprawdź zawartość

m. Dopisz w mainie fragment kodu pobierający oraz wyświetlający dostępne kategorie. Kategorie posortuj malejaco wg nazwy.

n. Uruchom i przetestuj aplikacje

o. Załóż break'pointa przy definiowaniu pierwszego zapytania. Przejdź do końca programu krokowo i prześledź na profilerze jakie wywołania komend sqlowych występują i w którym momencie.

II. Zarządzanie zmianami

a. Włącz możliwość migrowania modelu przechodząc do Managera pakietu (Tools -> Library Package Manager -> Package Manager Console) i wydając komendę Enable-Migrations

b. Prześledź co zostało dodane do projektu? Co te pliki zawierają?

c. Dokonaj zmian w klasie Category poprzez wprowadzenie do niej pola stringowego Description

d. Wykonaj polecenie Add-Migration AddDescription w konsoli managera pakietu.

e. Prześledź:

   i. co sie zmieniło w projekcie?

   ii. jak zmienił się model?

   iii. jakie pytania zostaly przesłane na serwer?

f. Wykonaj polecenie Update-Database (ew. z opcją -Verbose) w konsoli managera pakietu. Prześledź ponownie:

   i. co sie zmienilo w projekcie

   ii. jak zmienił się model

   iii. jakie pytania zostały przesłane na serwer

III. Data Annotations

a. Dodaj do projektu klasę Customer z polami stringowymi CompanyName oraz Description. Dodaj kolekcję klientow do kontekstu.

b. Dokonaj migracji modelu tak aby uwzględniona została klasa Customers

c. Co się nie powiodlo? Dlaczego?

d. Rozwiąż problem definiując pole CompanyName jako klucz główny tabeli klientow korzystacjąc z adnotacji [Key]

e. Dokonaj ponownej migracji modelu. Jak aktualnie to przebiega?

f. Dokonaj uaktualnienia struktury bazy danych. Prześledź wywołania sqlowe i zmiany jakie zostały wprowadzone

g. Dodaj do klasy Product pole Unitprice typu decimal. Doprowadz do sytuacji w której w bazie danych pole to jest utworzone jako pole typu Money

IV. Bindowanie danych do kontrolek

a. Dodaj do projektu formularz CategoryForm.

b. Doprowadć do jego wywołania (po wszystkich wydrukach na konsoli)

c. Dodaj kolekcję obiektów Category jako nowe źródło danych do projektu. Zwróć uwagę na automatycznie "zaciagniete" produkty

d. Osadź Kategorie na fromularzu CategoryForm

e. Pole CategoryID ustaw jak wyłącznie do odczytu

f. Wykonaj program. Jaka jest zawartość kontrolki DataGridView? Dlaczego?

g. Zdefiniuj metodę Load formularza CategoryForm w taki sposob aby dokonać zbindowania do kontrolki(kontrolek) wszystkich danych aktualnie "trakowanych" przez dbContext. W tym celu:

   i. zainstancjonuj kontekst

   ii. Załaduj dane (metoda Load na kolekcji Categories Kontekstu)

   iii. Przypisz lokalnie zarządzane przez kontekst kolekcje Categories jako źródło danych categoryBindingScource formularza

```
CategoryContext bContext = new CategoryContext();

bContext.Categories.Load();

this.categoryBindingSource.DataSource =
bContext.Categories.Local.ToBindingList();
```

h. Uruchom program, sprawdź jego działanie, usuń ew błędy.

i. Uruchom aplikację ponownie. Prześledź kiedy i jakie zapytania przesyłane są na serwer

j. Dodaj obsluge dodawania/zmiany danych przez formularz. W tym celu - uaktywnij przycisk/ikone zapisu i dodaj jego obsługę zawierajacą w szczególności zapisanie zmian na kontekście oraz odswieżenie kontrolek DataGridView

k. Osadź na formularzu Produkty z tego samego źrodła danych

l. Ponownie ProductID ustaw jak wyłącznie do odczytu

m. Dodaj obsługę formularza w taki sposób aby kliknięcie na kategorie na górnej liście powodowalo pokazanie produktow należących do tej konkretnej kategorii. Do filtrowania wykorzystaj zapytania L2E. Zapytanie przygotuj w obu notacjach.

V. Method syntax vs query syntax

a. Wróć na chwilę do części konsolowej. Dodaj metody które:

   i. Pobiorą i wypiszą nazwy kategorii – zapytanie zdefiniuj w method based synthax.

   ii. Załóż break'pointa przy definiowaniu pierwszego zapytania. Przejdź do końca programu krokowo i prześledź na profilerze jakie wywołania komend sqlowych występują i w którym momencie

b. Powtórz pobieranie i wyświetlanie zdefiniowanych kategporii tym razem wymuszając natychmiastową egzekucję zapytania. Ponownie prześledć na profilerze kiedy i jakie zapytania są przez serwer wykonywane.

c. Dodaj metody które (każde z zapytań przygotuj w dwóch wersjach wykorzystując query syntax i method based syntax):

i. Pobiorą i wyświetlą wszystkie kategorie i produkty wykorzystując:

1. Joiny
2. Navigation properties (jeśli możliwe)
3. lazy i eager loading. Prześledź wywołania zapytań

ii. Dla każdej kategorii pokażą liczbę produktow (jeśli dla kategorii brak produktu - wyświetl 0)

VI. Zadanie domowe

a. Rozszerz/zaimplementuj aplikację Product wprowadzajac możliwosc składania zamowien na produkty (ew inne funkcjonalności wg własnego pomysłu stosownie do umiejętności/czasu/preferencji/chęci poznania).

b. Zwróć szczególną uwagę na wykorzystanie mechanizmów EF i L2E w szczególności (postaraj się wykorzystać każde z poniższych chociaż raz): DataAnnotations, obie składnie L2E, lazy i eager loading, deferred i immediate execution, navigation properties, FluentAPI

c. Zadanie (stosownie do umiejętności/czasu/preferencji/chęci poznania) może zostać zrealizowane jako:

i. rozszerzenie rozpoczętej aplikacji WindowsFormowej

ii. aplikacja WPFowa

iii. aplikacja webowa

iv. aplikacja phonowa / UWPowa (np. nad SQLitem https://blogs.windows.com/buildingapps/2016/05/03/data-access-in-universal-windows-platform-uwp-apps/)

d. Przygotuj sprawozdanie omawiające przygotowaną aplikację, wykorzystane mechanizmy I sposób ich dzialania (wraz ze screenami).