# МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"**

Інститут **КНІТ**

Кафедра **ПЗ**

ЗВІТ

До лабораторної роботи № 3

**З дисципліни:** *“Конструювання програмного забезпечення”*

**На тему:** *“ADO.Net ORM. Data access through ADO.Net EF”*

**Лектор:** доц. каф. ПЗ Сердюк П. В.

**Виконав:** ст. гр. ПЗ-35 Хруставчук М.Л.

**Прийняв:** асистент каф. ПЗ Яценко Р. О.

« » 2024 р.

∑= .

Львів – 2024

**Тема роботи:** ADO.Net ORM. Data access through ADO.Net EF.

**Мета роботи:** Навчитися реалізовувати CRUD-операції за допомогою ADO.Net EF та реалізувати підходи DB First та Code First.

# ЗАВДАННЯ

1. Реалізувати додавання, оновлення та видалення даних за допомогою фреймворків ADO.Net EF.
2. Реалізувати підходи DB First та Code First.
3. Реалізувати міграцію для Code First. Для цього змінити модель і згенерувати міграцію (<https://www.entityframeworktutorial.net/code-first/code-based-migration-in-code-first.aspx>).
4. Використати SQL profiler (знаходиться у SQL management studio), щоб покзавати SQL запити.

Варіант БД: мережа спортивних комплексів.

# ХІД ВИКОНАННЯ

**1. Код розробленої програми**

*DBFirst*

Program.cs

using DBFirst.Models;

using System;

using System.Linq;

namespace DBFirst

{

class Program

{

static void Main(string[] args)

{

using (var context = new sportComplexEntities())

{

while (true)

{

Console.WriteLine("\nChoose an operation:");

Console.WriteLine("1. Add a client");

Console.WriteLine("2. Update a client");

Console.WriteLine("3. Delete a client");

Console.WriteLine("4. Show all clients");

Console.WriteLine("5. Exit");

Console.Write("Your choice: ");

var choice = Console.ReadLine();

switch (choice)

{

case "1":

AddClient(context);

break;

case "2":

UpdateClient(context);

break;

case "3":

DeleteClient(context);

break;

case "4":

ShowAllClients(context);

break;

case "5":

Console.WriteLine("Program terminated.");

return;

default:

Console.WriteLine("Invalid choice. Try again.");

break;

}

}

}

}

static void AddClient(sportComplexEntities context)

{

Console.Write("Enter the full name of the client: ");

var fullName = Console.ReadLine();

Console.Write("Enter the gender of the client (1 for male, 0 for female, leave empty if not specified): ");

var genderInput = Console.ReadLine();

bool? gender = string.IsNullOrEmpty(genderInput) ? (bool?)null : genderInput == "1";

Console.Write("Enter the phone number of the client: ");

var phoneNumber = Console.ReadLine();

var newClient = new Clients

{

client\_full\_name = fullName,

client\_gender = gender,

client\_phone\_number = phoneNumber

};

context.Clients.Add(newClient);

context.SaveChanges();

Console.WriteLine("Client successfully added.");

}

static void UpdateClient(sportComplexEntities context)

{

Console.Write("Enter the ID of the client to update: ");

if (int.TryParse(Console.ReadLine(), out int clientId))

{

var client = context.Clients.FirstOrDefault(c => c.client\_id == clientId);

if (client != null)

{

Console.Write("Enter the new full name of the client (leave empty to keep current): ");

var newName = Console.ReadLine();

if (!string.IsNullOrEmpty(newName))

client.client\_full\_name = newName;

Console.Write("Enter the new gender of the client (1 for male, 0 for female, leave empty to keep current): ");

var genderInput = Console.ReadLine();

if (!string.IsNullOrEmpty(genderInput))

client.client\_gender = genderInput == "1";

Console.Write("Enter the new phone number of the client (leave empty to keep current): ");

var newPhoneNumber = Console.ReadLine();

if (!string.IsNullOrEmpty(newPhoneNumber))

client.client\_phone\_number = newPhoneNumber;

context.SaveChanges();

Console.WriteLine("Client data successfully updated.");

}

else

{

Console.WriteLine("Client with such ID was not found.");

}

}

else

{

Console.WriteLine("Invalid ID.");

}

}

static void DeleteClient(sportComplexEntities context)

{

Console.Write("Enter the ID of the client to delete: ");

if (int.TryParse(Console.ReadLine(), out int clientId))

{

var client = context.Clients.FirstOrDefault(c => c.client\_id == clientId);

if (client != null)

{

context.Clients.Remove(client);

context.SaveChanges();

Console.WriteLine("Client successfully deleted.");

}

else

{

Console.WriteLine("Client with such ID was not found.");

}

}

else

{

Console.WriteLine("Invalid ID.");

}

}

static void ShowAllClients(sportComplexEntities context)

{

var clients = context.Clients.ToList();

if (clients.Any())

{

Console.WriteLine("List of clients:");

foreach (var client in clients)

{

string gender = client.client\_gender.HasValue

? (client.client\_gender.Value ? "Male" : "Female")

: "Not specified";

Console.WriteLine($"ID: {client.client\_id}, Name: {client.client\_full\_name}, Gender: {gender}, Phone: {client.client\_phone\_number}");

}

}

else

{

Console.WriteLine("The list of clients is empty.");

}

}

}

}

Purchases.cs

namespace DBFirst.Models

{

using System;

using System.Collections.Generic;

public partial class Purchases

{

public int purchase\_id { get; set; }

public int client\_id { get; set; }

public int subscription\_id { get; set; }

public int payment\_method\_id { get; set; }

public Nullable<int> purchase\_number { get; set; }

public System.DateTime purchase\_date { get; set; }

public virtual Clients Clients { get; set; }

public virtual PaymentMethods PaymentMethods { get; set; }

public virtual Subscriptions Subscriptions { get; set; }

}

}

*CodeFirst*

Program.cs

using CodeFirst.Data;

using CodeFirst.Models;

using System;

using System.Linq;

namespace CodeFirst

{

class Program

{

static void Main(string[] args)

{

using (var context = new SportComplexContext())

{

while (true)

{

Console.WriteLine("\nChoose an operation:");

Console.WriteLine("1. Add a client");

Console.WriteLine("2. Update a client");

Console.WriteLine("3. Delete a client");

Console.WriteLine("4. Show all clients");

Console.WriteLine("5. Exit");

Console.Write("Your choice: ");

var choice = Console.ReadLine();

switch (choice)

{

case "1":

AddClient(context);

break;

case "2":

UpdateClient(context);

break;

case "3":

DeleteClient(context);

break;

case "4":

ShowAllClients(context);

break;

case "5":

Console.WriteLine("Program terminated.");

return;

default:

Console.WriteLine("Invalid choice. Try again.");

break;

}

}

}

}

static void AddClient(SportComplexContext context)

{

Console.Write("Enter the full name of the client: ");

var fullName = Console.ReadLine();

Console.Write("Enter the gender of the client (1 for male, 0 for female, leave empty if not specified): ");

var genderInput = Console.ReadLine();

bool? gender = string.IsNullOrEmpty(genderInput) ? (bool?)null : genderInput == "1";

Console.Write("Enter the phone number of the client: ");

var phoneNumber = Console.ReadLine();

var newClient = new Client

{

client\_full\_name = fullName,

client\_gender = gender,

client\_phone\_number = phoneNumber

};

context.Clients.Add(newClient);

context.SaveChanges();

Console.WriteLine("Client successfully added.");

}

static void UpdateClient(SportComplexContext context)

{

Console.Write("Enter the ID of the client to update: ");

if (int.TryParse(Console.ReadLine(), out int clientId))

{

var client = context.Clients.FirstOrDefault(c => c.client\_id == clientId);

if (client != null)

{

Console.Write("Enter the new full name of the client (leave empty to keep current): ");

var newName = Console.ReadLine();

if (!string.IsNullOrEmpty(newName))

client.client\_full\_name = newName;

Console.Write("Enter the new gender of the client (1 for male, 0 for female, leave empty to keep current): ");

var genderInput = Console.ReadLine();

if (!string.IsNullOrEmpty(genderInput))

client.client\_gender = genderInput == "1";

Console.Write("Enter the new phone number of the client (leave empty to keep current): ");

var newPhoneNumber = Console.ReadLine();

if (!string.IsNullOrEmpty(newPhoneNumber))

client.client\_phone\_number = newPhoneNumber;

context.SaveChanges();

Console.WriteLine("Client data successfully updated.");

}

else

{

Console.WriteLine("Client with the given ID was not found.");

}

}

else

{

Console.WriteLine("Invalid ID entered.");

}

}

static void DeleteClient(SportComplexContext context)

{

Console.Write("Enter the ID of the client to delete: ");

if (int.TryParse(Console.ReadLine(), out int clientId))

{

var client = context.Clients.FirstOrDefault(c => c.client\_id == clientId);

if (client != null)

{

context.Clients.Remove(client);

context.SaveChanges();

Console.WriteLine("Client successfully deleted.");

}

else

{

Console.WriteLine("Client with the given ID was not found.");

}

}

else

{

Console.WriteLine("Invalid ID entered.");

}

}

static void ShowAllClients(SportComplexContext context)

{

var clients = context.Clients.ToList();

if (clients.Any())

{

Console.WriteLine("List of clients:");

foreach (var client in clients)

{

string gender = client.client\_gender.HasValue

? (client.client\_gender.Value ? "Male" : "Female")

: "Not specified";

Console.WriteLine($"ID: {client.client\_id}, Name: {client.client\_full\_name}, Gender: {gender}, Phone: {client.client\_phone\_number}");

}

}

else

{

Console.WriteLine("No clients found.");

}

}

}

}

Purchase.cs

using System;

using System.ComponentModel.DataAnnotations.Schema;

using System.ComponentModel.DataAnnotations;

namespace CodeFirst.Models

{

public class Purchase

{

[Key]

public int PurchaseId { get; set; }

[ForeignKey("Client")]

public int ClientId { get; set; }

public Client Client { get; set; }

[ForeignKey("Subscription")]

public int SubscriptionId { get; set; }

public Subscription Subscription { get; set; }

[ForeignKey("PaymentMethod")]

public int PaymentMethodId { get; set; }

public PaymentMethod PaymentMethod { get; set; }

[Range(1, int.MaxValue)]

public int PurchaseNumber { get; set; }

public DateTime PurchaseDate { get; set; } = DateTime.Now;

}

}

SportComplexContext.cs

using System.Collections.Generic;

using System.Data.Entity;

using System.Data.Entity.Infrastructure;

using System.Data.Entity.SqlServer;

using System.Runtime.Remoting.Contexts;

using CodeFirst.Models;

namespace CodeFirst.Data

{

public class SportComplexContext : DbContext

{

public SportComplexContext() : base("name=sportComplexEntities") {}

public DbSet<PaymentMethod> PaymentMethods { get; set; }

public DbSet<City> Cities { get; set; }

public DbSet<SubscriptionTerm> SubscriptionTerms { get; set; }

public DbSet<SubscriptionVisitTime> SubscriptionVisitTimes { get; set; }

public DbSet<SportComplex> SportComplexes { get; set; }

public DbSet<Gym> Gyms { get; set; }

public DbSet<Trainer> Trainers { get; set; }

public DbSet<Client> Clients { get; set; }

public DbSet<Subscription> Subscriptions { get; set; }

public DbSet<Purchase> Purchases { get; set; }

public DbSet<ActivityType> ActivityTypes { get; set; }

public DbSet<Training> Trainings { get; set; }

public DbSet<TrainerActivity> TrainerActivities { get; set; }

public DbSet<SubscriptionActivityType> SubscriptionActivityTypes { get; set; }

}

}

**2. Результати виконання розробленої програми**

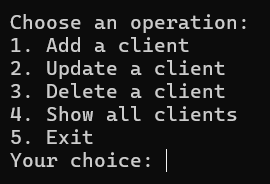
****

Рис. 1. Можливість вибору однієї із CRUD-операцій

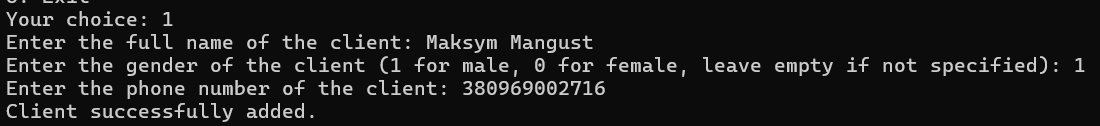
****

Рис. 2. Виконання операції додавання даних в БД

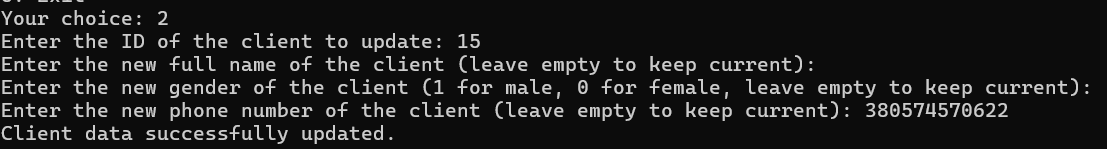


Рис. 3. Виконання операції оновлення даних в БД

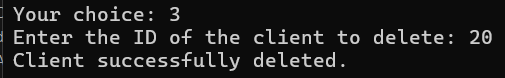


Рис. 4. Виконання операції видалення даних в БД

****

Рис. 5. Виконання операції виведення всіх даних в БД

# ВИСНОВКИ

У цій лабораторній роботі я навчився реалізовувати CRUD-операції за допомогою ADO.Net EF та реалізував підходи DB First та Code First. Для цього написав консольний додаток для реалізації цього завдання. Також навчився створювати міграції.