**національний університет БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ україни**

**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**ЛАБОРАТОРНА РОБОТА №8**

Виконав:

студент групи ІПЗ-2006б

Купчик В. Г.

Київ – 2023

**Завдання до лабораторної роботи**

**Мета роботи:** 1. Створення бази даних в середовищі VS 2015

2. Організація доступу до таблиць SQLEXPRESS 2015

3. Створення програмного застосунку для роботи з базою даних,

використовуючи приєднані об'єкти.

**Завдання до лабораторної роботи:**

1. Створити таблицю

2. Заповнити її даними (5 записів)

3. Створити форму із списком і текстовими полями

4. Реалізувати виведення даних у список і текстові поля

5. Реалізувати додавання, видалення та оновлення записів в таблиці.

Розв’язати завдання, відповідно варіанту.

Варіант - 15



Код SQL:

USE AthletesDB;

CREATE TABLE Athletes (

Code varchar(10) NOT NULL,

LastName varchar(50),

FirstName varchar(50),

MiddleName varchar(50),

BirthYear int,

Gender varchar(10),

SportType varchar(50),

Achievements varchar(100),

MedalType varchar(20),

);

INSERT INTO Athletes (Code, LastName, FirstName, MiddleName, BirthYear, Gender, SportType, Achievements, MedalType)

VALUES

('1', 'Іванов', 'Іван', 'Іванович', 1990, 'Чоловіча', 'Легка атлетика', 'Здобув золоту медаль на Олімпіаді', 'Золота медаль'),

('2', 'Петров', 'Петро', 'Петрович', 1988, 'Чоловіча', 'Плавання', 'Рекорд у вільному стилі', 'Без медалі'),

('3', 'Сидоренко', 'Сидор', 'Сидорович', 1995, 'Чоловіча', 'Бокс', 'Чемпіон світу', 'Дві золоті медалі'),

('4', 'Васильєва', 'Марія', 'Іванівна', 1992, 'Жіноча', 'Гімнастика', 'Багатократна призерка', 'Три золоті медалі'),

('5', 'Кучерявий', 'Сергій', 'Володимирович', 1987, 'Чоловіча', 'Важка атлетика', 'Рекорд у підйомі штанги', 'Золота медаль');

SELECT \*

FROM Athletes;

Код MainWindow.xaml

<Window x:Class="laba8.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="База даних спортсменів" Height="500" Width="1200">

<Grid>

<ListView Name="athletesListView" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="10,10,10,10" Width="1000">

<ListView.View>

<GridView>

<GridViewColumn Header="Код" DisplayMemberBinding="{Binding Code}" Width="60"/>

<GridViewColumn Header="Прізвище" DisplayMemberBinding="{Binding LastName}" Width="100"/>

<GridViewColumn Header="Імя" DisplayMemberBinding="{Binding FirstName}" Width="100"/>

<GridViewColumn Header="По-батькові" DisplayMemberBinding="{Binding MiddleName}" Width="120"/>

<GridViewColumn Header="Рік народження" DisplayMemberBinding="{Binding BirthYear}" Width="100"/>

<GridViewColumn Header="Стать" DisplayMemberBinding="{Binding Gender}" Width="80"/>

<GridViewColumn Header="Вид спорту" DisplayMemberBinding="{Binding SportType}" Width="100"/>

<GridViewColumn Header="Досягнення" DisplayMemberBinding="{Binding Achievements}" Width="210"/>

<GridViewColumn Header="Призи" DisplayMemberBinding="{Binding MedalType}" Width="120"/>

</GridView>

</ListView.View>

</ListView>

<StackPanel HorizontalAlignment="Right" VerticalAlignment="Top" Margin="320,10,10,0">

<TextBlock Text="Код:" />

<TextBox Name="codeTextBox" Width="150" />

<TextBlock Text="Прізвище:" />

<TextBox Name="lastNameTextBox" Width="150" />

<TextBlock Text="Імя:" />

<TextBox Name="firstNameTextBox" Width="150" />

<TextBlock Text="По-батькові:" />

<TextBox Name="middleNameTextBox" Width="150" />

<TextBlock Text="Рік народження:" />

<TextBox Name="birthYearTextBox" Width="150" />

<TextBlock Text="Стать:" />

<TextBox Name="genderTextBox" Width="150" />

<TextBlock Text="Вид спорту:" />

<TextBox Name="sportTypeTextBox" Width="150" />

<TextBlock Text="Досягнення:" />

<TextBox Name="achievementsTextBox" Width="150" />

<TextBlock Text="Призи:" />

<TextBox Name="medalTypeTextBox" Width="150" />

<Button Content="Додати" Click="AddButton\_Click" Width="80" Margin="-0,10,0,0" />

<Button Content="Редагувати" Click="UpdateButton\_Click" Width="80" Margin="0,5,0,0" />

<Button Content="Видалити" Click="DeleteButton\_Click" Width="80" Margin="0,5,0,0" />

<Button Content="Очистити" Click="ClearButton\_Click" Width="80" Margin="0,5,0,0" />

</StackPanel>

</Grid>

</Window>

Код MainWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace laba8

{

public partial class MainWindow : Window

{

private string connectionString = "Data Source=DESKTOP-HJDSDO1;Initial Catalog=AthletesDB;Integrated Security=True";

private ObservableCollection<Athlete> athletes;

public MainWindow()

{

InitializeComponent();

athletes = new ObservableCollection<Athlete>();

athletesListView.ItemsSource = athletes;

LoadAthletes();

// Додаємо подію SelectionChanged для обробки вибору елемента в ListView

athletesListView.SelectionChanged += AthletesListView\_SelectionChanged;

}

private void AthletesListView\_SelectionChanged(object sender, System.Windows.Controls.SelectionChangedEventArgs e)

{

// Перевіряємо, чи є обраний елемент

if (athletesListView.SelectedIndex != -1)

{

// Заповнюємо текстові поля даними обраного спортсмена

Athlete selectedAthlete = athletes[athletesListView.SelectedIndex];

codeTextBox.Text = selectedAthlete.Code;

lastNameTextBox.Text = selectedAthlete.LastName;

firstNameTextBox.Text = selectedAthlete.FirstName;

middleNameTextBox.Text = selectedAthlete.MiddleName;

birthYearTextBox.Text = selectedAthlete.BirthYear;

genderTextBox.Text = selectedAthlete.Gender;

sportTypeTextBox.Text = selectedAthlete.SportType;

achievementsTextBox.Text = selectedAthlete.Achievements;

medalTypeTextBox.Text = selectedAthlete.MedalType;

}

}

private void LoadAthletes()

{

athletes.Clear();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT \* FROM Athletes ORDER BY Code";

SqlCommand command = new SqlCommand(query, connection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

athletes.Add(new Athlete

{

Code = reader["Code"].ToString(),

LastName = reader["LastName"].ToString(),

FirstName = reader["FirstName"].ToString(),

MiddleName = reader["MiddleName"].ToString(),

BirthYear = reader["BirthYear"].ToString(),

Gender = reader["Gender"].ToString(),

SportType = reader["SportType"].ToString(),

Achievements = reader["Achievements"].ToString(),

MedalType = reader["MedalType"].ToString()

});

}

reader.Close();

}

}

private void AddButton\_Click(object sender, RoutedEventArgs e)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "INSERT INTO Athletes (Code, LastName, FirstName, MiddleName, BirthYear, Gender, SportType, Achievements, MedalType) " +

"VALUES (@Code, @LastName, @FirstName, @MiddleName, @BirthYear, @Gender, @SportType, @Achievements, @MedalType)";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@Code", codeTextBox.Text);

command.Parameters.AddWithValue("@LastName", lastNameTextBox.Text);

command.Parameters.AddWithValue("@FirstName", firstNameTextBox.Text);

command.Parameters.AddWithValue("@MiddleName", middleNameTextBox.Text);

command.Parameters.AddWithValue("@BirthYear", birthYearTextBox.Text);

command.Parameters.AddWithValue("@Gender", genderTextBox.Text);

command.Parameters.AddWithValue("@SportType", sportTypeTextBox.Text);

command.Parameters.AddWithValue("@Achievements", achievementsTextBox.Text);

command.Parameters.AddWithValue("@MedalType", medalTypeTextBox.Text);

command.ExecuteNonQuery();

}

ClearTextBoxes();

LoadAthletes();

}

private void UpdateButton\_Click(object sender, RoutedEventArgs e)

{

if (athletesListView.SelectedIndex != -1)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "UPDATE Athletes SET LastName = @LastName, FirstName = @FirstName, " +

"MiddleName = @MiddleName, BirthYear = @BirthYear, Gender = @Gender, " +

"SportType = @SportType, Achievements = @Achievements, MedalType = @MedalType " +

"WHERE Code = @Code";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@Code", athletes[athletesListView.SelectedIndex].Code);

command.Parameters.AddWithValue("@LastName", lastNameTextBox.Text);

command.Parameters.AddWithValue("@FirstName", firstNameTextBox.Text);

command.Parameters.AddWithValue("@MiddleName", middleNameTextBox.Text);

command.Parameters.AddWithValue("@BirthYear", birthYearTextBox.Text);

command.Parameters.AddWithValue("@Gender", genderTextBox.Text);

command.Parameters.AddWithValue("@SportType", sportTypeTextBox.Text);

command.Parameters.AddWithValue("@Achievements", achievementsTextBox.Text);

command.Parameters.AddWithValue("@MedalType", medalTypeTextBox.Text);

command.ExecuteNonQuery();

}

ClearTextBoxes();

LoadAthletes();

}

}

private void DeleteButton\_Click(object sender, RoutedEventArgs e)

{

if (athletesListView.SelectedIndex != -1)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "DELETE FROM Athletes WHERE Code = @Code";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@Code", athletes[athletesListView.SelectedIndex].Code);

command.ExecuteNonQuery();

}

ClearTextBoxes();

LoadAthletes();

}

}

private void ClearButton\_Click(object sender, RoutedEventArgs e)

{

ClearTextBoxes();

}

private void ClearTextBoxes()

{

codeTextBox.Text = "";

lastNameTextBox.Text = "";

firstNameTextBox.Text = "";

middleNameTextBox.Text = "";

birthYearTextBox.Text = "";

genderTextBox.Text = "";

sportTypeTextBox.Text = "";

achievementsTextBox.Text = "";

medalTypeTextBox.Text = "";

}

}

public class Athlete

{

public string Code { get; set; }

public string LastName { get; set; }

public string FirstName { get; set; }

public string MiddleName { get; set; }

public string BirthYear { get; set; }

public string Gender { get; set; }

public string SportType { get; set; }

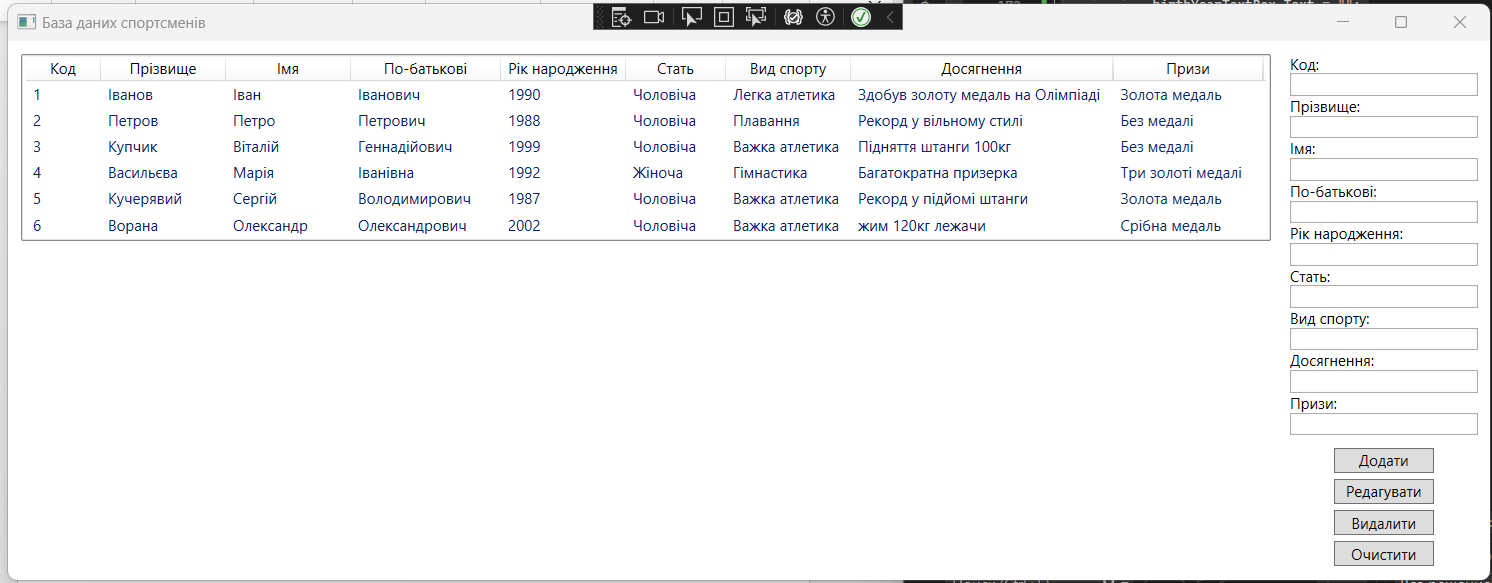
public string Achievements { get; set; }

public string MedalType { get; set; }

}

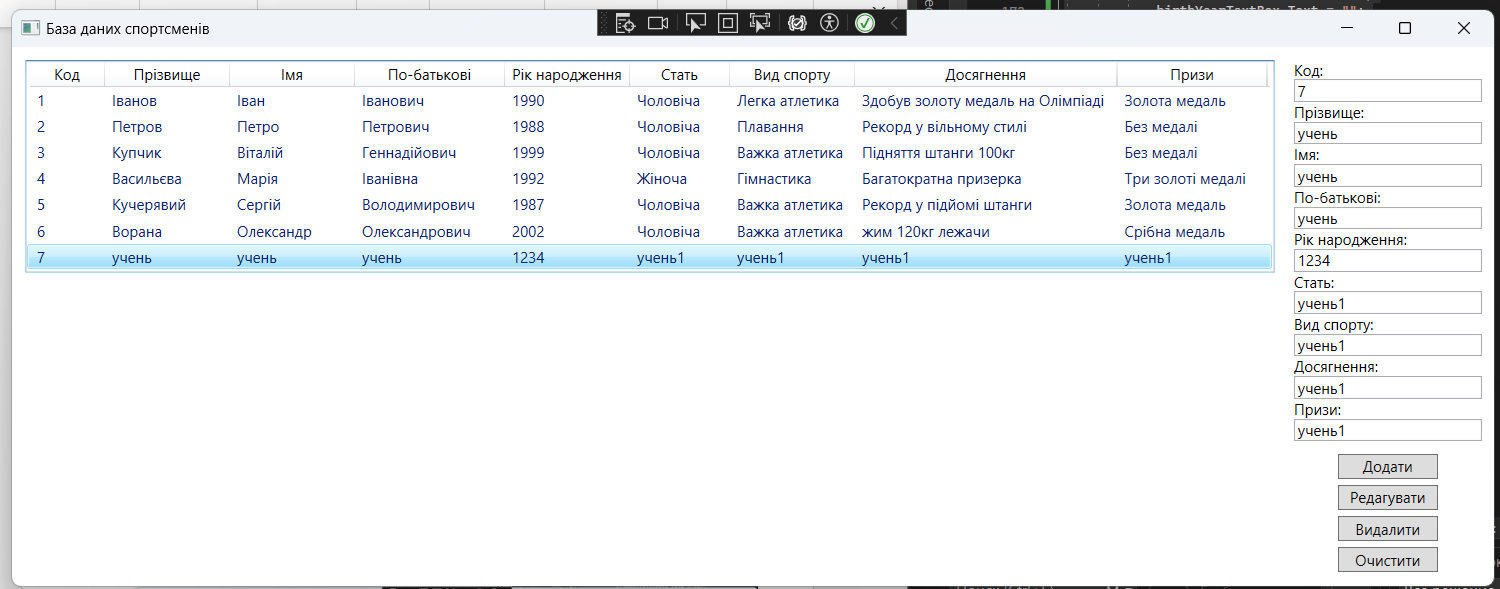
}

Результат виконання:



Додавання даних

При нажиманні на рядок з даними можна редагувати дані



Також можна видаляти дані



Висновок

Лабораторна робота з технології ADO.NET дозволила нам ознайомитися із створенням та управлінням базою даних в середовищі Visual Studio 2022. Ми успішно створили таблицю для зберігання інформації про спортсменів, реалізували з'єднання і доступ до цієї таблиці за допомогою ADO.NET.

У процесі роботи було створено програмний застосунок на платформі WPF (.NET Framework), який дозволяє виводити дані з бази даних у вікно застосунку. Також ми реалізували можливість додавання, редагування та видалення записів в таблиці, використовуючи текстові поля та кнопки в інтерфейсі користувача.

Особлива увага була приділена функціоналу вибору запису в списку для подальшого його редагування. Це дозволяє користувачу легко переглядати та змінювати дані конкретного спортсмена.

Загалом, лабораторна робота надає практичні навички роботи з ADO.NET, розробки WPF-додатку та взаємодії з базою даних в середовищі Visual Studio, що є важливим компонентом для будь-якого розробника програмного забезпечення.

Контрольні питання

1. Яка різниця між приєднаними і від'єднаними об'єктами?

Приєднані об'єкти (які представляють з'єднання з базою даних, наприклад, об'єкти Connection у ADO.NET) підтримують постійне підключення до бази даних і вимагають активного з'єднання протягом виконання операцій. Від'єднані об'єкти (наприклад, DataSet в ADO.NET) не підтримують постійне з'єднання і зберігають дані в пам'яті, поки вони не від'єднані від бази даних.

1. Яке призначення об'єкту класу Connection і які його основні властивості та методи?

Об'єкт класу Connection використовується для з'єднання з базою даних. Його основні властивості включають ConnectionString (рядок підключення), State (стан підключення) та інші. Методи включають Open() (для відкриття з'єднання), Close() (для закриття з'єднання) і багато інших для виконання запитів до бази даних.

1. Яке призначення об'єкту Command?

Об'єкт Command використовується для виконання SQL-запитів в базі даних. Він представляє запит, який може бути виконаний на підключенні до бази даних.

1. Яке призначення об'єкту DataReader?

Об'єкт DataReader використовується для читання результатів запиту, виконаного за допомогою об'єкта Command. Він дозволяє послідовно читати рядки з результатів запиту.

1. Назвіть основні методи об'єкту Command та їх призначення.

методи об'єкту Command включають ExecuteNonQuery (виконання запиту без повернення результатів), ExecuteReader (виконання запиту і повернення DataReader для читання результатів) і ExecuteScalar (повернення одного значення з результатів запиту).

1. Яка різниця між методами ExecuteNonQuery і ExecuteReader?

ExecuteNonQuery виконує запит, який змінює дані в базі даних (наприклад, INSERT, UPDATE або DELETE) і не повертає результатів запиту. ExecuteReader виконує запит і повертає DataReader, який дозволяє читати результати запиту.

1. Що означає запис SqlDataReader reader = yCommand.ExecuteReader()?

Цей код виконує запит, який представлений об'єктом yCommand, і отримує DataReader (reader), який можна використовувати для читання результатів запиту.

1. Як вставити в таблицю бази даних новий рядок? Який метод потрібно використати?

Для вставки нового рядка в таблицю бази даних зазвичай використовують SQL-запит типу INSERT. Ви можете використовувати об'єкт Command для виконання такого запиту.

1. Як видалити з таблиці бази даних рядок?

Для видалення рядка з таблиці бази даних використовують SQL-запит типу DELETE. Ви можете використовувати об'єкт Command для виконання такого запиту.