Лабораторна робота №4. Організація взаємодії з базою даних за допомогою ADO.NET

Ціль : Розробити програму для перегляду даних, що зберігаються в SQL-базі даних.

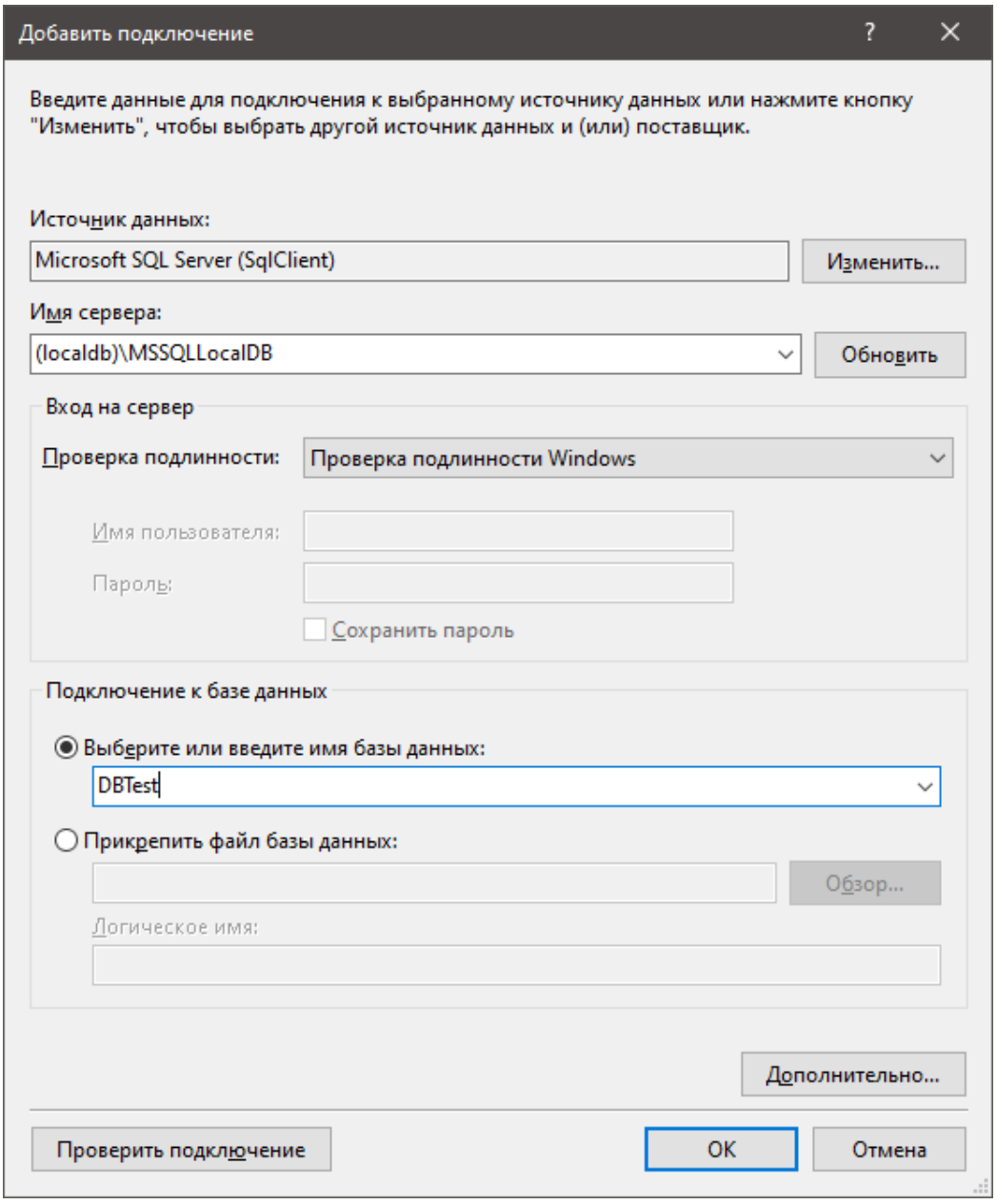
Завдання 1. Створення простої однотабличної бази даних

Основні теоретичні відомості

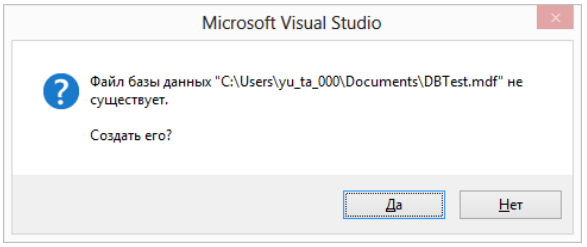
Створення локальної бази даних

Порядок створення локальної SQL бази даних:

1. Запустіть **Visual Studio**.
2. Відкрийте **браузер серверів**, скориставшись однойменною командою з меню **View**.
3. Клацніть правою кнопкою миші на вузлі **Підключення даних** та виберіть у контекстному меню команду **Додати підключення…** .
4. У діалозі вкажіть ім'я створюваної бази даних і натисніть кнопку **ОК**.



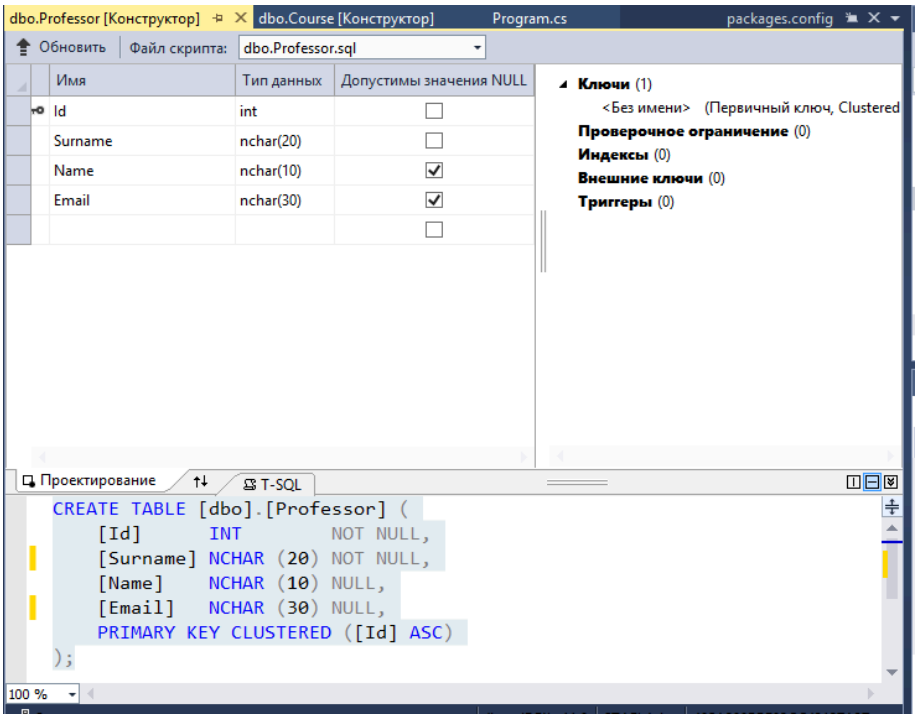
5. Підтвердьте необхідність створення нової бази даних.



Додавання таблиці до бази даних

Щоб додати таблицю до бази даних, виконайте такі дії:

1. У **браузері серверів** розгорніть вузол створеної бази даних.
2. Клацніть правою кнопкою миші по вузлу **Таблиці** та виберіть у контекстному меню команду **Додати нову таблицю** .
3. У вікні редактора таблиць вкажіть для кожного поля назву, тип та інші властивості (за потреби).

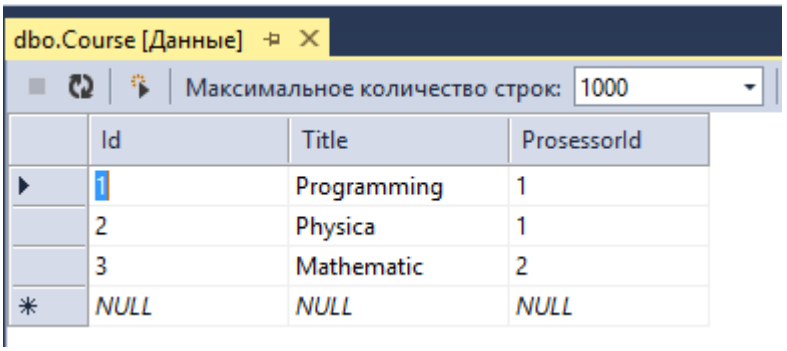


4. Введіть назву таблиці та оновіть базу даних.

Редагування даних

Для введення та редагування даних у таблицю виконайте такі дії:

1. У **браузері серверів** розгорніть вузол **таблиці** .
2. Клацніть правою кнопкою миші на вузлі таблиці та виберіть у контекстному меню команду **Показати таблицю даних**.
3. Відредагуйте дані таблиці.



4. Закрийте таблицю.
5. Аналогічно заповніть даними інші таблиці бази даних.

Додавання бази даних до проекту та створення файлу конфігурації

Щоб додати базу даних до проекту, виконайте такі дії:

1. У **браузері рішень** виділіть корінь проекту і командою **Додати/Створити папку** контекстного меню створіть підкаталог для розміщення бази даних, наприклад, ***Data***.
2. У **браузері рішень** за допомогою команди контекстного меню **Додати/Існуючий елемент…** додайте файл бази даних ***.mdf** до проекту (як правило, вона знаходиться в каталозі C:\Users\Admin).
3. Виділіть у **браузері** файл доданої бази даних (***.mdf**) і через панель **Властивості** перевірте, що його властивості мають такі значення: **Дія при складанні** = **Зміст**, **Копіювати у вихідний каталог** = **Завжди копіювати**.

Для створення конфігураційного файлу виконайте такі дії:

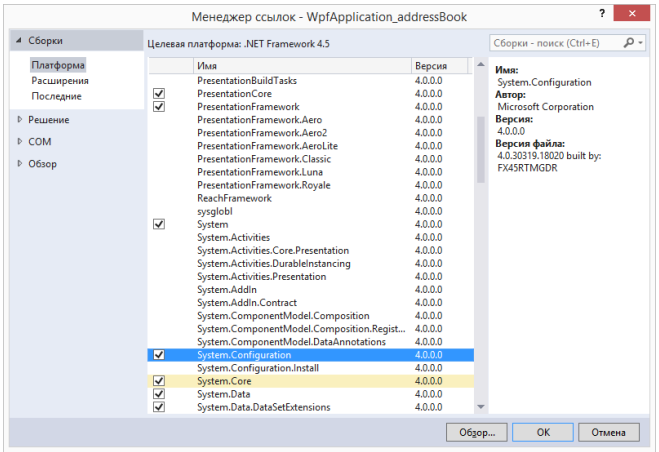
1. Відкрийте файл конфігурації **App.config у проекті** .
2. Вставте в розділ <**configuration**/> розділ <**connectionStrings**/>, що містить відомості про рядок з'єднання з базою даних (ім'я та значення):

```
<connectionStrings>
    <add name="connectionString_ADO"
     connectionString="Data Source=R501N10;Initial Catalog=Clients;Integrated Security=True"/>
 </connectionStrings>
```

Тут *name* - це ім'я з'єднання*, Initial Catalog* - це назва бази даних*, Data Source* - це ім'я сервера бази даних *(*SQL Server name*).*

Значення рядка з'єднання можна скопіювати з властивостей бази даних.

3. У **браузері рішень** викличте для вузла **References** проекту контекстне меню та командою **Додати посилання** додайте посилання на бібліотечну збірку **System.Configuration.dll**, в якій знаходиться клас **ConfigurationManager** для роботи з конфігураційним файлом **App.config** з процедурного коду:



Тепер рядок з'єднання з файлу **App.config** можна отримати таким чином:

String connectionString = System.Configuration.ConfigurationManager.ConnectionStrings["*connectionStringName* "].ConnectionString;

Хід виконання

1. Створіть нову SQL базу даних, що складається з однієї таблиці, відповідно до свого варіанта завдання (Таблиця 1).
2. Збережіть файл сценарію створення таблиці та скопіюйте його у звіт про виконання завдання 1.
3. Додайте дані до бази даних.
4. Скопіюйте створену базу даних у проект.
5. Налаштуйте конфігураційний файл для роботи з локальною базою даних.
6. Файл конфігурації **App.config** скопіюйте у звіт про виконання завдання 1.

Таблиця 1. Варіанти завдань

Варіанти	**Таблиця**	**Поля**
1-5	Клієнти	ID, Назва, Телефон, Адреса, Надходження, Витрати
6-10	Товари	Артикул, Назва, Одиниця виміру, Кількість, Ціна
11-15	Студенти	Номер залікової книги, ПІБ, Група, Адреса
16-20	Книги	ISBN, Назва, Автори, Видавництво, Рік видання
21-25	Співробітники	ID, ПІБ, Підрозділ, Посада, Телефон
Свій варіант		

Завдання 2. Використання ADO.NET та розробка класу доступу до бази даних

Основні теоретичні відомості

Використання ADO.NET

ADO.NET дозволяє взаємодіяти з базою даних за допомогою об'єктів підключення, читання даних та команд конкретного постачальника даних.

Підключення та читання записів із бази даних вимагає виконання наступних кроків:

1) Створення об'єкта підключення.
2) Створення об'єкта команди.
3) Створення об'єкта читання даних та обробка записів.

Для створення об'єкта підключення необхідно:

- Імпортувати в проект простори імен **System.Data** та **System.Data.SqlClient**.
- Створити об'єкт підключення типу **SqlConnection**, скориставшись оператором **new**.
- Надіслати об'єкту підключення рядок підключення, скориставшись його властивістю **ConnectionString**.
- Відкрити з'єднання за допомогою методу **Open()** об'єкту підключення.

Приклад коду, що реалізує ці операції:

```
SqlConnection cn = new SqlConnection();

cn.ConnectionString = @"Data Source=(local)\SQLEXPRESS;Integrated Security=SSPI;" + "Initial Catalog=AutoLot";

cn.Open();
```

Для створення об'єкта команди та відправки SQL-запитів до бази даних призначено тип **SqlCommand**. SQL- запит може передаватися через параметр конструктора команди, або через властивість команди **CommandText**.

Приклад коду, що реалізує ці операції:

```
SqlConnection cn = new SqlConnection();

string strSQL = "Select * From Inventory";//SQL-запит

//...

//Створення об'єкта команди за допомогою конструктора

SqlCommand myCommand = new SqlCommand(strSQL, cn);


// Створення об'єкта команди за допомогою властивості CommandText

SqlCommand testCommand = new SqlCommand();

testCommand.Connection = cn;

testCommand.CommandText = strSQL;
```

Для читання даних використовується об'єкт читання даних **SqlDataReader**, який можна отримати з об'єкта команди з допомогою методу **ExecuteReader()**. Щоб після зчитування даних об'єкт підключення закрився автоматично, метод слід викликати з **CommandBehavior.CloseConnection**.

Об'єкт читання даних дозволяє почергово обробляти записи за допомогою **Read()**. Після зчитування даних необхідно звільнити об'єкт читання методом **Close()**.

Приклад коду, що реалізує ці операції:

```
// Отримання об'єкта читання даних за допомогою ExecuteReader

SqlDataReader myDataReader;

myDataReader = myCommand.ExecuteReader (CommandBehavior.CloseConnection);

// Цикл за результатами

while (myDataReader.Read())

{

        Console.WriteLine("Field1: {0}, Field2: {1}, Field3: {2}.",

        myDataReader ["Field1"].ToString().Trim(),

        myDataReader ["Field2"].ToString().Trim(),

        myDataReader["Field3"].ToString().Trim());

}

myDataReader.Close();
```

Виконання операцій вставки, видалення та зміни даних зводиться до формування відповідного SQL-запиту та виклику методу **ExecuteNonQuery** () об'єкта команди:

```
SqlCommand cmd = new SqlCommand (strSQL, cn);

cmd.ExecuteNonQuery();
```

Приклади рядків з SQL-запитами:

```
string strSQL = string.Format("Insert Into Inventory" +"(CarID, Make, Color, PetName) Values" +"('{0}', '{1}', '{2}', '{3} ')", id, make, color, petName);

string strSQL = string.Format("Delete from Inventory where CarID = '{0}'", id);

string strSQL =string.Format("Update Inventory Set PetName = '{0}' Where CarID = '{1}'", newPetName, id);
```

Такі запити не повертають набір результатів, тому називаються типу «NonQuery».

Виконання операції вибірки даних зводиться до формування відповідного SQL-запиту та виклику методу **ExecuteReader()** об'єкта команди. Цей запит повертає набір результатів і відноситься до типу **Query**.

Для повернення відібраних даних зручно користуватися об'єктом **System.Data.DataTable**. Цей клас надає користувачеві табличний блок даних, який може бути заповнений програмним способом за допомогою методу **Load()**.

Приклад коду, що реалізує такі операції:

```
// Тут будуть відібрані записи

DataTable inv = new DataTable();

// Підготовка об'єкта команди
```

```
string strSQL = "Select * From Inventory";

SqlCommand cmd = new SqlCommand (strSQL, cn);

SqlDataReader dr = cmd.ExecuteReader();

// Заповнення DataTable даними з об'єкта читання

inv.Load(dr);

dr.Close();
```

Для читання даних з **DataTable** можна також використовувати **SqlDataAdapter**, який входить до складу **System**.**Data**.**SqlClient**. Після отримання даних через **SqlDataAdapter** можна локально працювати з цими даними незалежно від наявності підключення.

Для використання **SqlDataAdapter** треба налаштувати рядок підключення, сформулювати SQL-запит SELECT, на основі відповідної SQL-команди побудувати екземпляр SqlDataAdapter і за допомогою метода Fill() здійснити завантаження даних та заповнення DataTable.

Приклад коду, що реалізує такі операції:

```
// Тут будуть відібрані записи

DataTable inv = new DataTable();

// Підготовка об'єкта команди

string strSQL = "Select * From Inventory";

SqlCommand cmd = new SqlCommand (strSQL, cn);

SqlDataAdapter adr = new SqlDataAdapter (cmd, cn);

// Заповнення адаптера даними

adr.Fill(inv);

adr.Close();
```

Розробка класу доступу до бази даних

Описані вище операції взаємодії з базою даних за допомогою ADO.NET зручно організувати у вигляді окремого класу, який отримує рядок з'єднання їх файлу конфігурації та повертає об'єкт DataTable:

```
// Клас доступу до БД

public class AdoAssistant

{

// Отримуємо рядок з'єднання з файлу App.config

String connectionString = System.Configuration.

ConfigurationManager.ConnectionStrings[" connectionStringName "].ConnectionString;

//********************************************** *********
```

```
// Метод читання даних з DataTable
//********************************************** *********
DataTable dt = null; // Посилання на об'єкт DataTable
public DataTable TableLoad()
{
if (dt != null) return dt; // Завантажимо таблицю лише один раз
// Заповнюємо об'єкт таблиці даними з БД
dt = new DataTable();
// Створюємо об'єкт підключення
using (SqlConnection connection = new SqlConnection(connectionString))
{
SqlCommand command = connection.CreateCommand(); // Створюємо об'єкт команди
SqlDataAdapter adapter = new SqlDataAdapter(command); // Створюємо об'єкт читання
//Завантажує дані
command.CommandText = "Select ID," + "(Surname + Name) AS FullName, " +
"Address, Place, PostalCode From Table_Address";
try
{
        // Метод сам відкриває БД і сам її закриває
        adapter.Fill(dt);
}
        catch (Exception)
{
        MessageBox.Show("Помилка підключення до БД");
}
}
        return dt;
}
}
```

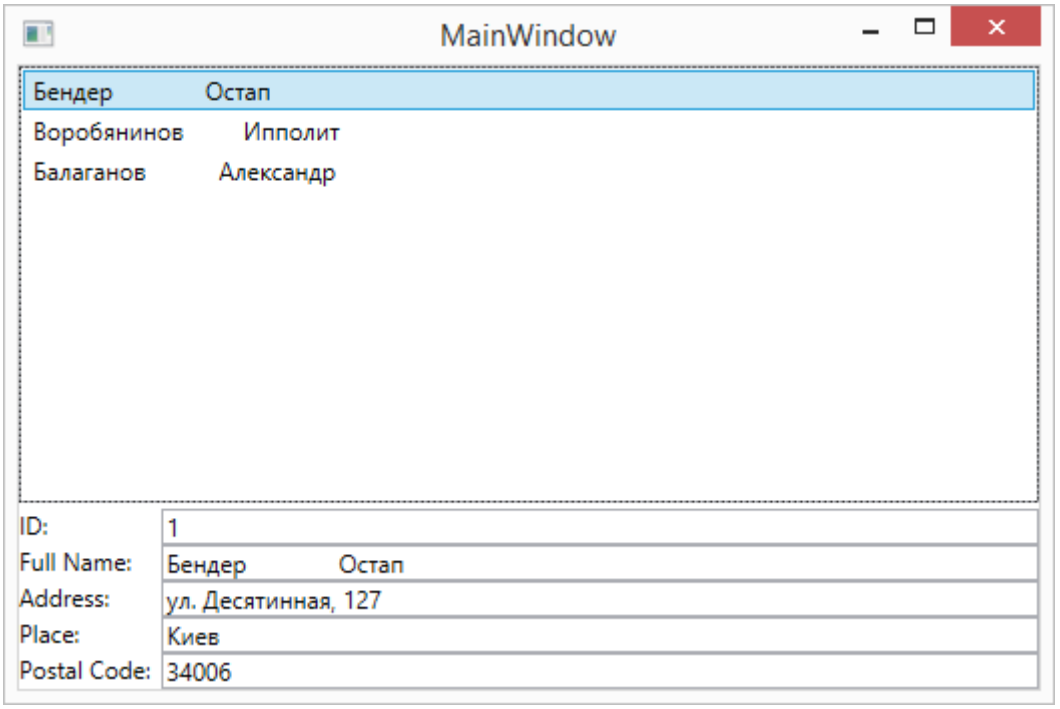
Хід виконання

1. Додайте в проект файл класу доступу до бази даних і опишіть метод для отримання даних з бази даних за допомогою **DataTable**.
2. Збережіть код розробленого класу у звіті про виконання завдання 2.

Завдання 3. Розробка інтерфейсу програми та прив'язування даних

Основні теоретичні відомості

Пропонується організувати інтерфейс, що містить список ListBox, та табличну частину для відображення детальних відомостей про обраний у списку об'єкт :



Такий інтерфейс вимагає:

1) прив'язки таблиці даних до списку **ListBox**;
2) прив'язки полів вибраного у списку **ListBox** запису до відповідних текстових полів у табличній частині.

Створюється інтерфейс в XAML-файлі таким чином:

```
<Grid
        DataContext="{Binding ElementName=listFullName, Path=SelectedItem}">
        <Grid.ColumnDefinitions>
                <ColumnDefinition Width="Auto" />
                <ColumnDefinition />
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
                <RowDefinition Height="*" />
```

```
                <RowDefinition Height="Auto" />
                <RowDefinition Height="Auto" />
                <RowDefinition Height="Auto" />
                <RowDefinition Height="Auto" />
                <RowDefinition Height="Auto" />
                <RowDefinition Height="Auto" />
            </Grid.RowDefinitions>

<ListBox Grid.Row="0"
            Grid.Column="0"
            Grid.ColumnSpan="2"
            Margin = "0,0,0,3"
            ScrollViewer.VerticalScrollBarVisibility="Auto"
            Name="list"
            ItemsSource="{Binding}" DisplayMemberPath="FullName" />

<TextBlock Grid.Row="1" Margin="5">ID:</TextBlock>
<TextBox Grid.Row="1" Grid.Column="1"
Text="{Binding Path=ID, Mode=OneWay}"
Focusable="True"/>

<TextBlock Grid.Row="2" Margin="5">Full Name:</TextBlock>
<TextBox Grid.Row="2" Grid.Column="2"
Text="{Binding Path=FullName, Mode=OneWay}"
Focusable="True" />

<TextBlock Grid.Row="3" Margin="5">Address:</TextBlock>
<TextBox Grid.Row="3" Grid.Column="2"
Text="{Binding Path=Address, Mode=OneWay}"
Focusable="True" />
```

```
<TextBlock Grid.Row="4" Margin="5">Place:</TextBlock>

<TextBox Grid.Row="4" Grid.Column="2"

Text="{Binding Path=Place, Mode=OneWay}"

Focusable="True" />


<TextBlock Grid.Row="5" Margin="5">Postal Code:</TextBlock>

<TextBox Grid.Row="5" Grid.Column="2"

Text="{Binding Path=PostalCode, Mode=OneWay}"

Focusable="True"

/>

</Grid>
```

Залишається у файлі приєднаного коду як властивості **DataContext** списку вказати таблицю **DataTable**. Зробити це можна за допомогою наступного методу:

```
private void Window_Loaded(object sender, RoutedEventArgs e)

{

        AdoAssistant myTable= new AdoAssistant();

        list.SelectedIndex = 0;

        list.Focus();

        list.DataContext = myTable.TableLoad();

}
```

Хід виконання

1. Відкрийте XAML-файл програми та опишіть у ньому інтерфейс програми для перегляду бази даних відповідно до свого варіанта завдання (Таблиця 2).
2. У XAML-коді забезпечте прив'язку даних.
3. У файлі приєднаного коду напишіть метод для отримання контексту даних.
4. Перевірте функціональність програми.
5. Збережіть XAML код розмітки у звіті про виконання завдання 3.
6. Збережіть вміст файлу приєднаного коду у звіті про виконання завдання 3.

Таблиця 2. Дані для розробки інтерфейсу

Варіанти	**Таблиця**	**Поле списку**	**Поля табличної частини**
1-5	Клієнти	Назва	ID, Телефон, Адреса, Сплачено, Одержано
6-10	Товари	Назва	Артикул, Одиниця виміру, Кількість, Ціна
11-15	Студенти	ПІБ	Номер залікової книги, Група, Адреса

16-20	Книги	Назва	ISBN, Автори, Видавництво, Рік видання
21-25	Співробітники	ПІБ	ID, Підрозділ, Посада, Телефон
Свій варіант			

Завдання 4. Реалізація операцій додавання, оновлення і видалення записів бази даних

Основні теоретичні відомості

Елемент керування ToolBar

Панель інструментів представляє собою набір команд, зазвичай розташований безпосередньо під рядком меню в стандартній програмі Windows. Насправді це може бути просто панель з кнопками на ній, проте при використанні елемента керування ToolBar користувач отримує кілька додаткових переваг у вигляді автоматичної обробки переповнення панелі або можливості користувача переставити панелі інструментів.

```
<ToolBar Height="25" VerticalAlignment="Top">
   <Button>Create</Button>
   <Separator />
   <Button>Update></Button>
   <Separator />
   <Button>Delete</Button>
   <TextBox Foreground="LightGray" Width="100">Поиск...</TextBox>
</ToolBar>
```

Елемент WPF ToolBar зазвичай міститься всередині контейнера ToolBarTray .Перевага його використання полягає в можливості встановити як горизонтальне, так і вертикальне розташування елементів ToolBar у вікні програми. Також можна розміщувати кілька елементів керування ToolBar усередині контейнера ToolBarTray:

```
<ToolBarTray>
    <ToolBar>
       // …
   </ToolBar>
   <ToolBar>
      //…
   </ToolBar>
</ToolBarTray>
```

Хід виконання

1. Доповніть клас доступу до бази даних (**AdoAssistant.class** з завдання 2) методами для додавання, оновлення і видалення записів бази даних на основі відповідних SQL-запитів типу «NonQuery».
2. В коді г форми застосунку (XAML-файлі розроблений в завданні 3) весь вміст контейнера **Grid** перемістіть в створений для цього контейнер **DockPanel**.
3. Додайте в **DockPanel** панель інструментів **ToolBar** з трьома кнопками: **Create**, **Update** та **Delete**.
4. До кожної з кнопок панелі інструментів додайте обробник події **Click** для виконання відповідної операції. Сутність операції полягає у виконанні SQL-запиту типу «NonQuery», необхідні параметри якого беруться з форми застосунку, та зчитування отриманої таблиці з бази даних.
5. Перевірте працездатність застосунку.