



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY



Instrukcja współfinansowana przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego
w projekcie

*„Innowacyjna dydaktyka bez ograniczeń
– zintegrowany rozwój Politechniki Łódzkiej – zarządzanie Uczelnią,
nowoczesna oferta edukacyjna i wzmacniania zdolności
do zatrudniania osób niepełnosprawnych”*

Instrukcja jest dystrybuowana bezpłatnie.

Instrukcja do laboratorium

Piotr Korbel

Projektowanie i programowanie systemów bezprzewodowych

Komunikacja z bazami danych

Zadanie nr 14 – Studia podyplomowe „Bezprzewodowe systemy nadzoru i monitorowania”



Politechnika Łódzka
Instytut Elektroniki

90-924 Łódź, ul. Żeromskiego 116,
tel. 042 631 28 83
www.kapitalludzki.p.lodz.pl

Komunikacja z bazami danych

1. Wprowadzenie

Celem zajęć laboratoryjnych jest zapoznanie słuchaczy z podstawowymi sposobami komunikacji z bazami danych za pomocą aplikacji wykorzystujących kod zarządzany (.NET Compact Framework). Jako serwer bazy danych wykorzystany zostanie Microsoft SQL Compact Server 3.5 SP1.

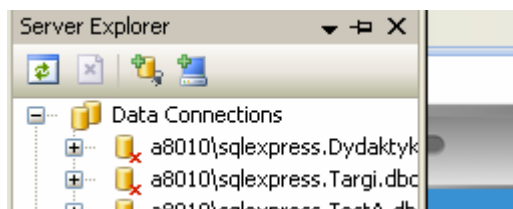
2. Instalacja SQL Compact Server

Podczas instalacji na urządzeniu mobilnym (lub emulatorze) programu napisanego z wykorzystaniem kodu zarządzanego, Visual Studio dokonuje sprawdzenia, czy na urządzeniu są zainstalowane wymagane biblioteki (.NET Framework) oraz SQL Compact Server. Jeżeli serwer nie został uprzednio zainstalowany, zostaje automatycznie dołączony do paczki dystrybucyjnej.

SQL Compact Server jest instalowany na stacjach deweloperskich wraz z pakietem Visual Studio.

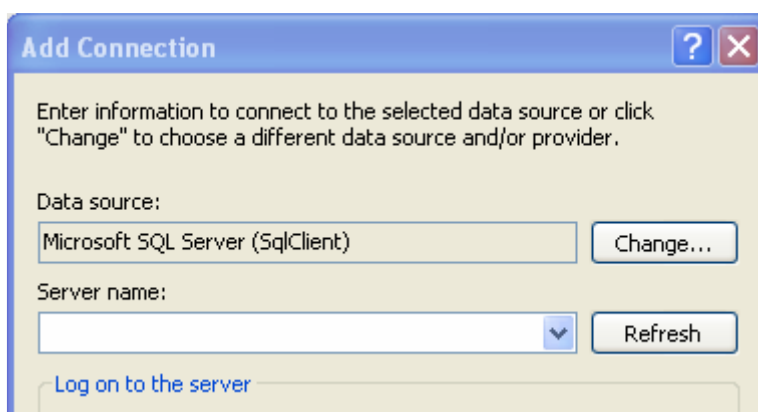
3. Dodawanie bazy danych do projektu

W celu dodania do projektu bazy danych, w pakiecie Visual Studio należy wyświetlić okno Server/Database Explorer (menu *View->Server Explorer*). Następnie należy wskazać kursorem węzeł *Data Connections* (Rys. 1) i po kliknięciu prawym klawiszem wybrać polecenie *Add Connection*.

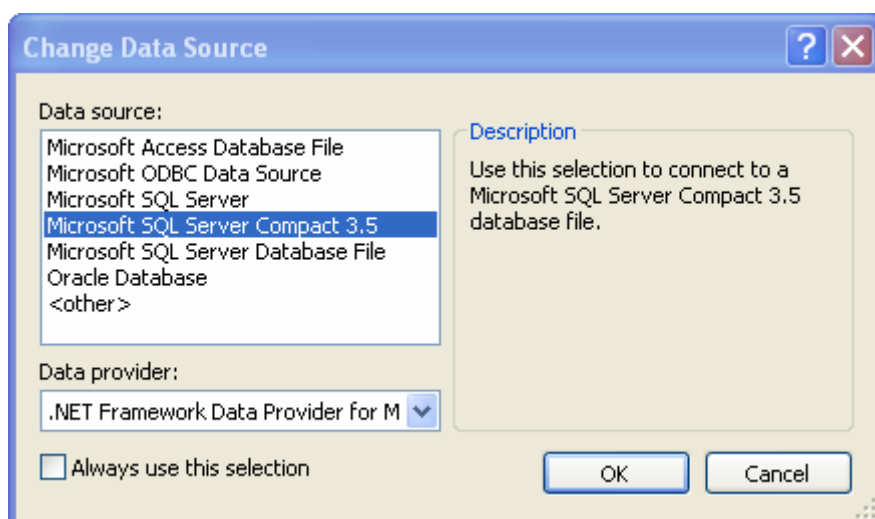


Rys. 1 Okno zarządzania połączeniami ze źródłami danych

W oknie dialogowym nowego połączenia *Add Connection* (Rys. 2) należy zweryfikować, czy wybrany rodzaj źródła jest serwerem SQL Server Compact 3.5. Jeżeli nie, należy zmienić poprzez wywołanie (przycisk *Change...*) okna dialogowego wyboru rodzaju źródła danych (Rys. 3).

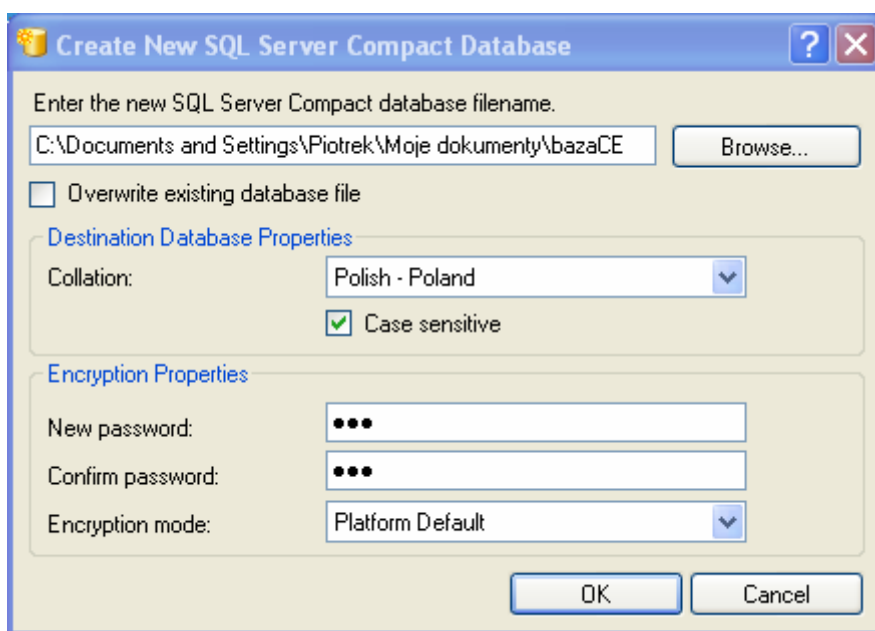


Rys. 2 Okno dialogowe tworzenia nowego połączenia ze źródłem danych



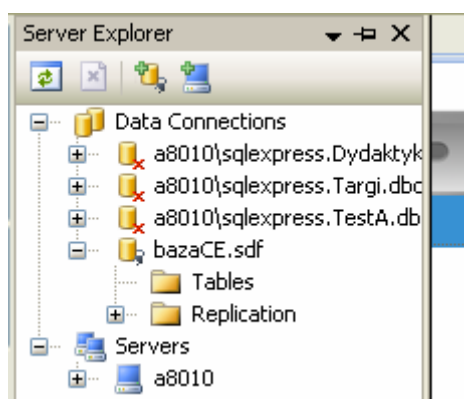
Rys. 3 Okno wyboru typu źródła danych

Po wybraniu rodzaju źródła danych (Microsoft SQL Server Compact 3.5) należy zdefiniować właściwości połączenia (*Connection Properties*). W przypadku dodawania odwołania do istniejącej bazy, w polu *Database* należy podać ścieżkę do pliku bazy (*Browse...*). W przypadku tworzenia nowego pliku bazy danych, należy wywołać okno dialogowe *Create New SQL Server Compact 3.5 Database* (przycisk *Create...*), zdefiniować położenie pliku bazy na stacji deweloperskiej oraz pozostałe właściwości (Rys. 4).



Rys. 4 Tworzenie nowej bazy danych

Po zakończeniu tworzenia nowej bazy można zweryfikować poprawność wykonanych operacji poprzez wykonanie testowego połączenia z bazą (przycisk *Test Connection*). Po pomyślnym wykonaniu testu oraz zatwierdzeniu operacji przyciskiem *OK*, utworzona baza powinna pojawić się w oknie *Solution Explorer/Data Connections* (Rys. 5).

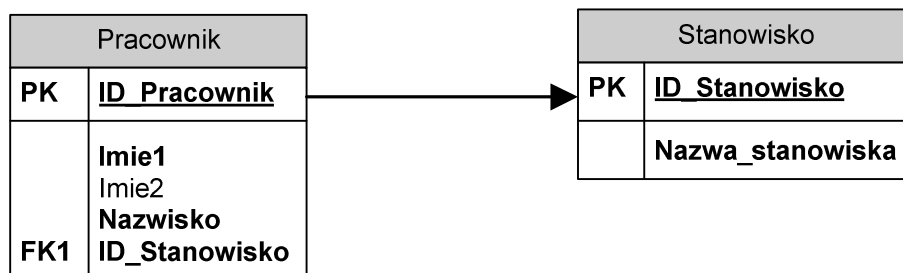


Rys. 5 Utworzona baza pojawia się w oknie *Solution Explorer*

W tym momencie można przystąpić do zdefiniowania struktury bazy. Korzystając z menu kontekstowego dostępnego po naciśnięciu prawego przycisku myszy i wskazaniu pola *Tables* uzyskujemy dostęp do poleceń umożliwiających tworzenie tabel bazy danych (*Create Table*). Strukturę bazy można także definiować za pomocą komend języka SQL (*New Query*). Korzystając z polecenia *New Query* (instrukcja INSERT – opis podstawowych instrukcji języka SQL znajduje się w Dodatku B) można następnie zapęłnić tabele przykładowymi danymi.

Do edycji struktury bazy można także wykorzystać narzędzie SQL Server Management Studio.

W ramach ćwiczeń należy zdefiniować bazę o strukturze przedstawionej na diagramie (Rys. 6) oraz wypełnić ją przykładowymi danymi (INSERT).



Rys. 6 Struktura prostej bazy danych

Po wypełnieniu bazy przykładowymi danymi należy zdefiniować i przetestować przykładowe kwerendy wybierające (SELECT).

4. Komunikacja z lokalną bazą danych

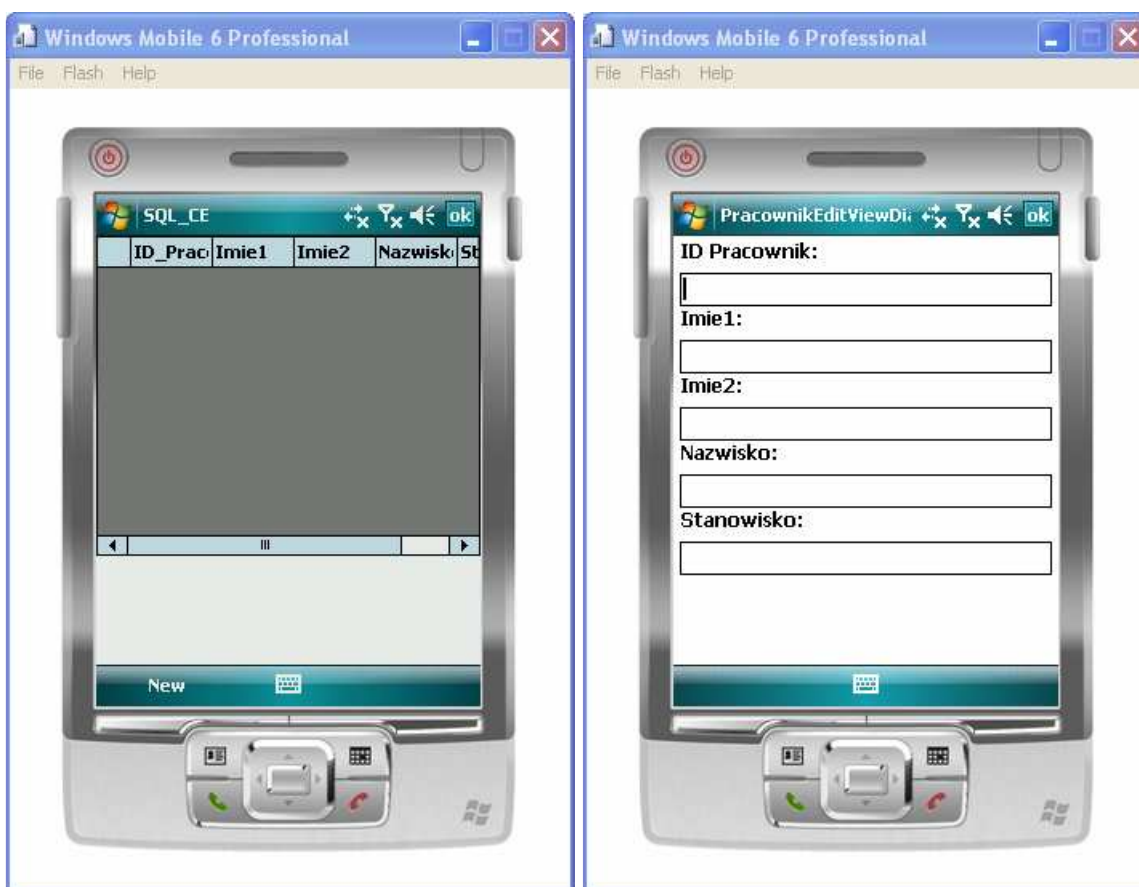
W celu dodania do projektu referencji do istniejącego źródła danych (*Data Source*), należy wybrać menu *Project->Add Existing Item* (Obiekt typu *Microsoft SQL Server Compact 3.5 database File*), następnie wskazać położenie pliku bazy. Definiując źródło danych należy następnie wskazać obiekty bazy (np. tabele), z których chcemy pobierać dane.

Kolejnym etapem budowy aplikacji jest dodanie do formularza kontrolek (np. elementów interfejsu graficznego) oraz powiązania ich ze źródłami danych. Dostępne elementy interfejsu graficznego znajdują się w zestawie kontrolek (*Toolbox*) w zakładce *Device Data*. Podstawowym elementem graficznym wykorzystywanym do prezentacji danych jest tabela (*Data Grid*). Po przeciągnięciu komponentu na formularz aplikacji należy w oknie właściwości (*Properties*) zdefiniować źródło danych właściwe dla danego elementu.

5. Przykładowy program

Projekt *SQL_CE_Local.sln* zawiera szablon programu łączącego się z przykładową bazą danych (plik *bazaCE.sdf*). Podstawowym elementem interfejsu graficznego programu jest tabela, która podczas uruchomienia wypełniana jest danymi pobieranymi z bazy. Korzystając z menu kontekstowego komponentu *DataGrid* (menu *Generate Data Forms...*) wygenerowane zostały formularze pomocnicze służące do wprowadzania nowych krotek (formularz dostępny z poziomu menu aplikacji – przycisk *New*) oraz do wyświetlania kompletu danych dla zaznaczonej krotki (formularz wyświetlany po wskazaniu krotki w tabeli danych).

W ramach ćwiczeń należy skompilować oraz uruchomić przykładowy projekt (korzystając z środowiska emulatora lub urządzeń dostępnych w laboratorium). Interfejs graficzny programu przedstawiony został na Rys. 7.



Rys. 7 Interfejs graficzny przykładowej aplikacji: ekran podstawowy oraz ekran wprowadzania danych (środowisko emulatora)

Po uruchomieniu oraz przetestowaniu działania programu należy:

- wzbogacić funkcje interfejsu graficznego o możliwość sortowania danych wyświetlanych w kontrolce *DataGrid* (*pracownikBindingSource.Sort*);
- przećwiczyć definiowanie nowych oraz wprowadzanie modyfikacji do istniejących źródeł danych (kwerendy wybierające).



Literatura – źródła internetowe

[1] <http://msdn.microsoft.com/>



Dodatek A

Kod przykładowego programu do komunikacji z lokalną bazą danych SQL Compact Server

Form1.cs

```
using System;

using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlServerCe;

namespace SQL_CE_Local
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            private void Form1_Load(object sender, EventArgs e)
            {
                // TODO: This line of code loads data into the
                'bazaCEDataSet.Pracownik' table. You can move, or remove it,
                as needed.

                this.pracownikTableAdapter.Fill(this.bazaCEDataSet.Pracownik);

            }

            private void newItemMenuItem_Click(object sender,
            EventArgs e)
            {
                pracownikBindingSource.AddNew();
                SQL_CE_Local.PracownikEditViewDialog
                pracownikEditViewDialog =
                SQL_CE_Local.PracownikEditViewDialog.Instance(this.pracownikBi
                ndingSource);
                pracownikEditViewDialog.ShowDialog();
            }
        }
    }
}
```



```
    }

    private void dataGrid1_Click(object sender, EventArgs
e)
    {
        SQL_CE_Local.PracownikSummaryViewDialog
pracownikSummaryViewDialog =
SQL_CE_Local.PracownikSummaryViewDialog.Instance(this.pracowni
kBindingSource);
        pracownikSummaryViewDialog.ShowDialog();
    }
}
}
```

PracownikEditViewDialog.cs

```
using System;

using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace SQL_CE_Local
{
    public partial class PracownikEditViewDialog : Form
    {
        public PracownikEditViewDialog()
        {
            InitializeComponent();
        }

        private void PracownikEditViewDialog_Closing(object
sender, CancelEventArgs e)
        {
            this.pracownikBindingSource.EndEdit();
        }
    }
}
```

PracownikSummaryViewDialog.cs



```
using System;

using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace SQL_CE_Local
{
    public partial class PracownikSummaryViewDialog : Form
    {
        public PracownikSummaryViewDialog()
        {
            InitializeComponent();
            // Attach event handlers to auto-hide controls.
            this.AttachVisibilityBindings(this.Controls);
        }

        private void editMenuItemMenuItem_Click(object sender,
EventArgs e)
        {
            SQL_CE_Local.PracownikEditViewDialog
pracownikEditViewDialog =
SQL_CE_Local.PracownikEditViewDialog.Instance(this.pracownikBi
ndingSource);
            pracownikEditViewDialog.ShowDialog();
            this.Close();
        }

        private void PracownikSummaryViewDialog_KeyDown(object
sender, KeyEventArgs e)
        {
            if ((e.KeyCode == System.Windows.Forms.Keys.Up))
            {
                this.AutoScrollPosition = new
System.Drawing.Point(0, ((0 - this.AutoScrollPosition.Y)
- 16));
                e.Handled = true;
            }
            if ((e.KeyCode == System.Windows.Forms.Keys.Down))
            {
                this.AutoScrollPosition = new
System.Drawing.Point(0, ((0 - this.AutoScrollPosition.Y)
+ 16));
                e.Handled = true;
            }
        }
    }
}
```





```
}  
if ((e.KeyCode == System.Windows.Forms.Keys.Up))  
{  
    // Up  
}  
if ((e.KeyCode == System.Windows.Forms.Keys.Down))  
{  
    // Down  
}  
if ((e.KeyCode == System.Windows.Forms.Keys.Left))  
{  
    // Left  
}  
if ((e.KeyCode ==  
System.Windows.Forms.Keys.Right))  
{  
    // Right  
}  
if ((e.KeyCode ==  
System.Windows.Forms.Keys.Enter))  
{  
    // Enter  
}  
}  
}  
}
```



Dodatek B

Podstawowe instrukcje języka SQL

INSERT – wpisywanie danych do tabel

| | |
|--|--|
| INSERT [INTO] | wstaw w pola tabeli |
| table_name [(column_list)] | nazwa tabeli (lista kolumn) |
| { VALUES | wartości |
| ({ DEFAULT NULL expression } [,...n]) | (domyślne, puste, określone wyrażeniem) |
| derived_table | wartości zwracane przez kwerendę SELECT |
| } | |

SELECT – wybieranie danych z tabel

| | |
|--|---|
| SELECT select_list | lista pól do wybrania z tabeli |
| [FROM table_source] | nazwa tabeli, z której wybierane są dane |
| [WHERE search_condition] | gdzie spełnione są warunki |
| [GROUP BY group_by_expression] | wyniki należy pogrupować wg wyrażenia |
| [HAVING search_condition] | warunki wybierania danych |
| [ORDER BY order_expression [ASC DESC]] | dane należy posortować rosnąco/malejąco wg klucza |

Przykłady:

SELECT * FROM Pracownik; – wybranie wszystkich danych z tabeli Pracownik;

SELECT (ID_Pracownik, Nazwisko) FROM Pracownik; – wybranie pól ID_Pracownik oraz Nazwisko dla wszystkich krotek w tabeli Pracownik;

INSERT INTO Pracownik (ID_Pracownik, Imie1, Nazwisko) VALUES (15, 'Jan', 'Nowak') – wpisywanie przykładowych danych do tabeli Pracownik (pola ID_Pracownik, Imie1, Nazwisko)