

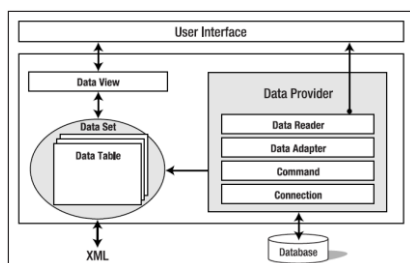
Technologia .Net

Bazy danych

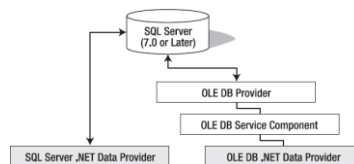
Technologia ADO.Net

- Służy do dostarczania danych z różnych źródeł (baz danych) do aplikacji
- Jest produktem Microsoft
- Umożliwia manipulowanie danymi XML
- Pozwala na logiczne modelowanie danych

Architektura ADO.NET



SQL Server i OLE DB



Przegląd obiektów dostawczych

- **SqlClient** jest zoptymalizowany pod kątem współpracy z MS SQL
- **OLE DB** (*Object Linking and Embedding, Database – łączenie i zagnieżdżanie obiektów*) jest rodzajem **Application programming interface (API)** opracowanym przez Microsoft i pozwala łączyć się ze źródłami, dla których opracowano dostawcę
- **Odbc Open Database Connectivity** to zestaw uniwersalnych narzędzi dostępu do baz danych
- **Java Database Connectivity** narzędzia dostępu do źródeł danych dla Javy

Obiekty dostawców danych w .Net

- **Connection** – nawiązuje połączenie ze źródłem danych
- **Command** – wykonuje polecenia SQL w źródle danych
- **DataReader** – zwraca jednokierunkowy, przeznaczony do odczytu strumień danych
- **DataAdapter** – tworzy pomost pomiędzy obiektem DataSet a źródłem danych

Obiekt Connection

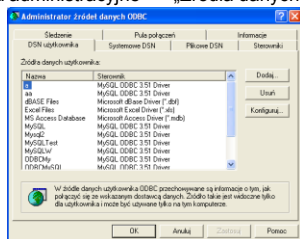
- Służy do nawiązywania połączenia ze źródłem danych z użyciem nazwy użytkownika i hasła
- OdbcConnection
- OleDbConnection
- SqlConnection

OdbcConnection

- Wymaga odpowiedniego dla danego SZBD sterownika odbc (Driver)
- Microsoft dostarcza w pakiecie Net typowe sterowniki
- Użycie połączenia wymaga zdefiniowania źródła danych

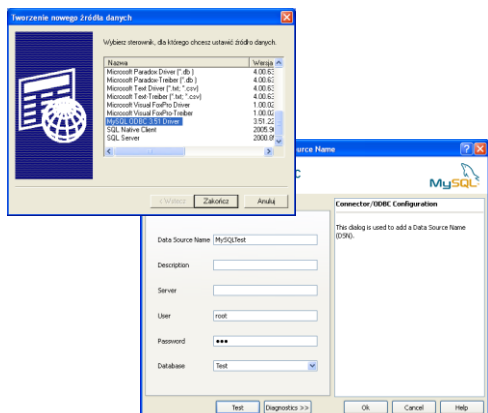
Tworzenie źródeł danych odbc

- W Windows 2000 i wyżej „Panel Sterowania” -> „Narzędzia administracyjne” -> „Źródła danych (ODBC)”

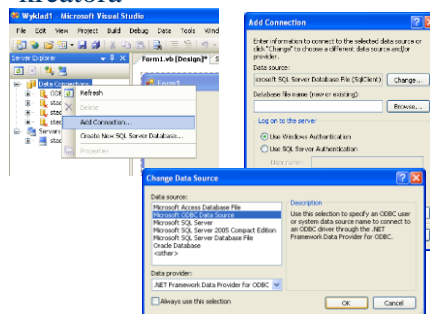


Tworzenie źródeł danych odbc

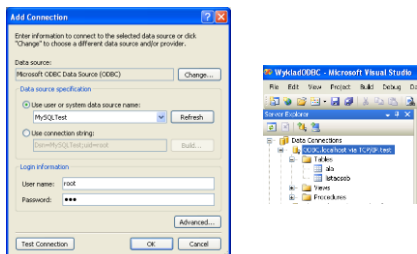
- DSN użytkownika są dostępne tylko dla zalogowanego użytkownika
- DSN systemowe są widoczne dla wszystkich użytkowników komputera
- DSN plikowe może być użytkowany przez wszystkich, którzy mają odpowiedni sterownik



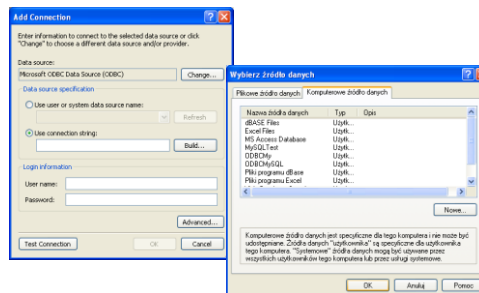
Tworzenie połączenia przy użyciu kreatora



Tworzenie połączenia przy użyciu kreatora



Tworzenie DS (inny sposób)



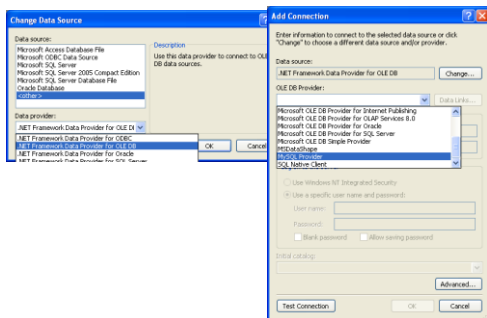
Programowe ustawianie połączenia ODBC

```
OdbcConnection PolaczenieOdbc = new OdbcConnection();
PolaczenieOdbc.ConnectionString =
"Dsn=MySQL Test;database=Test;option=0;port=0;uid=root";
try
{
    PolaczenieOdbc.Open();
    MessageBox.Show("Połączenie otwarte");
}
catch
{
    MessageBox.Show("Problemy z otwarciem połączenia");
}
```

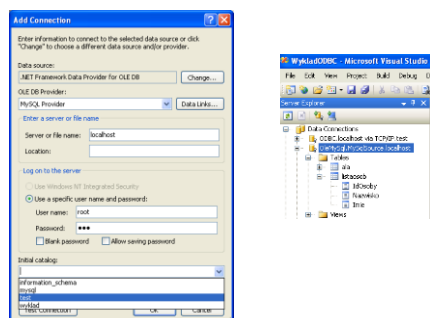
OleDbConnection

- Dla tego połączenia nie tworzy się źródła danych
- Konieczny jest dostawca (Provider) zainstalowany na komputerze

Wybór dostawców

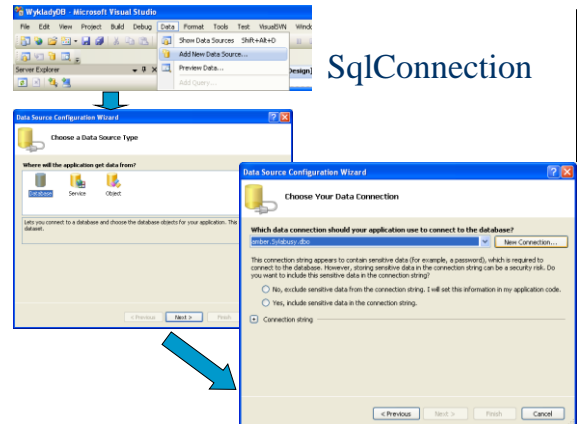


Wybór źródła

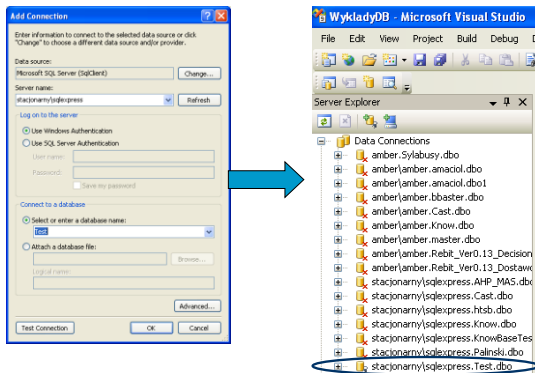


Programowe ustawianie połączenia OLE

```
OleDbConnection PolaczenieOleDb = new OleDbConnection();
PolaczenieOleDb.ConnectionString =
"Provider=OleMySQL.MySqlSource.1;Data Source=localhost;Persist
Security Info=True;Password=abc;User ID=root;Initial Catalog=test";
try
{
    PolaczenieOleDb.Open();
    MessageBox.Show("Połączenie otwarte");
}
catch
{
    MessageBox.Show("Problemy z otwarciem połączenia");
}
```



SqlConnection



Programowe ustawianie połączenia SQL – windows authentication

nazwa serwera na komputerze

```
SqlConnection PolaczenieSql = new SqlConnection();
PolaczenieSql.ConnectionString = "Data Source=stacjonarny\\sqlserver;Initial Catalog=Test;Integrated
Security=True";
try
{
    PolaczenieSql.Open();
    MessageBox.Show("Połączenie otwarte");
}
catch
{
    MessageBox.Show("Problemy z otwarciem połączenia");
}
```

Programowe ustawianie połączenia SQL – windows authentication

nazwa serwera na komputerze

```
SqlConnection PolaczenieSql = new SqlConnection();
PolaczenieSql.ConnectionString = "Data Source=L3N10;Persist
Security Info=True;User Id=sa;User Instance=False; Password=mssql;
Initial Catalog=Test ";
try
{
    PolaczenieSql.Open();
    MessageBox.Show("Połączenie otwarte");
}
catch
{
    MessageBox.Show("Problemy z otwarciem połączenia");
}
```

Programowe ustawianie połączenia SQL – server authentication

nazwa serwera na komputerze

```
Dim SQLConn As New SqlConnection
SQLConn.ConnectionString =
"Data Source=L3N10;Persist Security Info=True;User Id=sa;User
Instance=False;Password=mssql;Initial Catalog=Test"
Try
    SQLConn.Open()
    MsgBox("Połączenie otwarte", MsgBoxStyle.OkOnly)
Catch ex As Exception
    MsgBox("Problemy z otwarciem połączenia",
    MsgBoxStyle.OkOnly)
End Try
```

Obiekt Command

- Wykonuje polecenia SQL
- Właściwość Connection wskazuje połączenie do wykonania polecenia
- Obiekt jest realizowany w trzech wersjach ODBC, OLE i SQL

Sposoby wykonywania poleceń

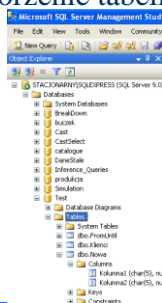
- ExecuteNonQuery – wykonuje polecenie, które nie zwraca żadnych rekordów
- ExecuteScalar – zwraca pierwszą kolumnę pierwszego wiersza
- ExecuteReader – zwraca wynikowy zestaw rekordów za pośrednictwem obiektu DataReader
- ExecuteXMLReader - zwraca wynikowy zestaw rekordów w formacie XML za pośrednictwem obiektu XMLReader

ExecuteNonQuery – tworzenie tabeli w katalogu 'Test'

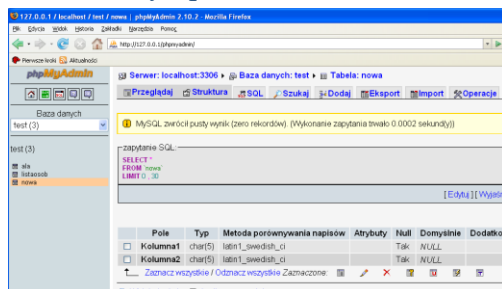
- Uwaga! źródło danych musi być otwarte (patrz poprzednie listingi)
- W przypadku wykonywania poleceń w wersji OleDb konieczne jest aktywowanie tzw. providera (np. <http://cherrycitysoftware.com/CCS/Providers>)

```
int rezultat;
string sql;
SqlCommand Polecenie = new SqlCommand();
Polecenie.Connection = PolaczenieSql;
Polecenie.CommandType = CommandType.Text;
sql = "CREATE TABLE Nowa (Kolumna1 char(5), Kolumna2
char(5))";
Polecenie.CommandText = sql;
try
{
    rezultat = Polecenie.ExecuteNonQuery();
    MessageBox.Show("Wykonano polecenie" + sql);
}
catch
{
    MessageBox.Show("Problemy z wykonaniem polecenia" +
sql);
}
```

ExecuteNonQuery – tworzenie tabeli w katalogu 'Test'



ExecuteNonQuery – tworzenie tabeli w katalogu 'Test' – wersja dla OLE DB i MySql



ExecuteNonQuery z parametrami

- Do tabeli **Prowadzacy** dopisujemy rekord przy użyciu procedury składowanej

STACJONARNY		Prowadzacy		Summary	
	ProwadzacyId	Nazwisko	Imie		Tytul
▶	1	Maciej	... Andrzej	...	dr inż.
	2	Kot	... Jan	...	prof.
	3	Słoń	... Antoni	...	mgr
★	ALL	ALL	ALL	...	ALL

Budujemy programowo nową procedurę 1/2

```
int rezultat;
string sql;
SqlCommand Polecenie = new SqlCommand();
Polecenie.Connection = PolaczenieSql;
Polecenie.CommandType = CommandType.Text;
sql = "CREATE PROCEDURE DopisaniePracownika
@Nazwisko char(50), @Imie char(50), @Tytul char(10)";
sql += " As DECLARE @ind int";
sql += " INSERT INTO Prowadzacy (Nazwisko, Imie, Tytul)";
sql += " VALUES (@Nazwisko, @Imie, @Tytul)";
sql += " SET @ind=SCOPE_IDENTITY() RETURN(@ind)";
Polecenie.CommandText = sql;
```

Budujemy programowo nową procedurę 2/2

```
try
{
    rezultat = Polecenie.ExecuteNonQuery();
    MessageBox.Show("Wykonano polecenie" + sql);
}
catch
{
    MessageBox.Show("Problemy z wykonaniem polecenia" +
sql);
}
```

Skrypt opisujący nową procedurę

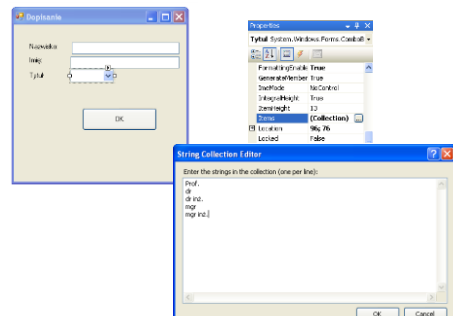
```
CREATE PROCEDURE [dbo].[DopisaniePracownika] @Nazwisko
char(50), @Imie char(50), @Tytul char(10) As DECLARE @ind int
INSERT INTO Prowadzacy (Nazwisko, Imie, Tytul) VALUES
(@Nazwisko, @Imie, @Tytul) SET @ind=SCOPE_IDENTITY()
RETURN(@ind)
```

Formularz do wprowadzania danych

```
public string _Nazwisko;
public string _Imie;
public string _Tytul;
```

```
private void OK_Click(object sender, EventArgs e)
{
    _Nazwisko = Nazwisko.Text;
    _Imie = Imie.Text;
    _Tytul = Tytul.Text;
    this.Close();
}
```

Tworzenie listy typu ComboBox



```

int rezultat;
SqlCommand polecenie = new SqlCommand();
Polecenie.Connection = PolaczenieSql;
Polecenie.CommandType = CommandType.StoredProcedure;
Polecenie.CommandText = "DopisaniePracownika";
Polecenie.Parameters.Add("@Nazwisko", SqlDbType.Char);
Polecenie.Parameters["@Nazwisko"].Direction =
ParameterDirection.Input;
Polecenie.Parameters.Add("@Imie", SqlDbType.Char);
Polecenie.Parameters["@Imie"].Direction =
ParameterDirection.Input;
Polecenie.Parameters.Add("@Tytul", SqlDbType.Char);
Polecenie.Parameters["@Tytul"].Direction =
ParameterDirection.Input;
Polecenie.Parameters.Add("@ind", SqlDbType.Char);
Polecenie.Parameters["@ind"].Direction =
ParameterDirection.ReturnValue;

```

Definiowanie parametrów

```

Form2 Formularz = new Form2();
Formularz.ShowDialog();
Polecenie.Parameters["@Nazwisko"].Value =
Formularz._Nazwisko;
Polecenie.Parameters["@Imie"].Value = Formularz._Imie;
Polecenie.Parameters["@Tytul"].Value = Formularz._Tytul;
try
{
    rezultat = Polecenie.ExecuteNonQuery();
    int wynik =
Convert.ToInt16(Polecenie.Parameters["@ind"].Value);
    MessageBox.Show("Indeks dodanego rekordu: " +
wynik.ToString());
}
catch
{
    MessageBox.Show("Problemy z wykonaniem polecenia");
}

```

Wykonanie procedury

Efekty działania programu

	ProwadzącyId	Nazwisko	Imie	Tytul
1	1	Maciąg	Andrzej	dr inż.
2	2	Kot	Jan	prof.
3	3	Słoniowski	Antoni	mgr
4	16	Nowak	Jan	prof.

ExecuteScalar – dla MS SQL

- Budujemy procedurę, która liczy ile jest studentów lub studentek
- Parametrem procedury jest zmienna Płeć równa 1 dla kobiet i 0 dla mężczyzn
- Płeć jest rozpoznawana po ostatniej literze imienia ('a' to kobiety)

```

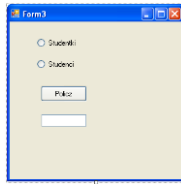
ALTER PROCEDURE [dbo].[Policz]
    @Plec bit
AS
BEGIN
    If @Plec=1
    BEGIN
        SELECT COUNT(*)
        FROM Studenci
        WHERE (SUBSTRING(Imie, LEN(Imie), 1) = 'a')
    END
    ELSE
    BEGIN
        SELECT COUNT(*)
        FROM Studenci
        WHERE (SUBSTRING(Imie, LEN(Imie), 1) <> 'a')
    END
END

```

Wyniki działania procedury

StudentId	Nazwisko	Imie	PESEL	Kod
1	Kowalski	Jan	870101096518	32-098
2	Abacka	Anna	88101004517	32-088
3	Osa	Ewa	86020405345	30-150
4	Maciąg	Andrzej	56031606618	25-118
5	Stawowy	Adam	12345678906	54-321
6	Nowy	Jan	12345678906	32-082

Formularz do uruchamiania procedury



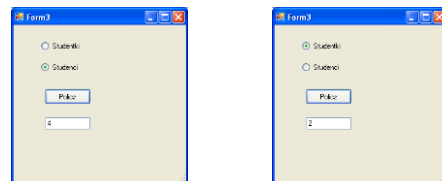
Ustalanie parametrów procedury

```
{
    Polecenie.Connection = Form1.PolaczenieSql;
    Polecenie.CommandType = CommandType.StoredProcedure;
    Polecenie.CommandText = "Policz";
    Polecenie.Parameters.Add("@Plec", SqlDbType.Int);
    Polecenie.Parameters["@Plec"].Direction =
        ParameterDirection.Input;
}
```

Uruchamianie procedury

```
if (Student1.Checked)
{
    Polecenie.Parameters["@Plec"].Value = 0;
}
else
{
    Polecenie.Parameters["@Plec"].Value = 1;
}
Wynik.Text = Convert.ToString(Polecenie.ExecuteScalar());
```

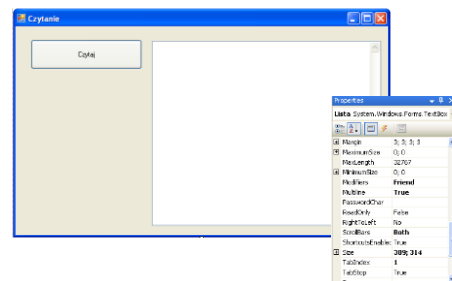
Wyniki działania programu



Obiekt DataReader

- Udostępnia jednokierunkowy strumień wierszy
- Jest najszybszym sposobem dostępu do danych

Odczytywanie listy klientów



Kod

```
SqlConnection PolaczenieSql = new SqlConnection();
PolaczenieSql.ConnectionString = "Data
Source=stacjonarny\\sqlxpress;Initial Catalog=Buczek;Integrated
Security=True";
PolaczenieSql.Open();
string sql = "SELECT Nazwa FROM KlienciAlfabet ORDER BY
Nazwa";
SqlCommand Komenda = new SqlCommand(sql, PolaczenieSql);
SqlDataReader Czytnik = Komenda.ExecuteReader();
while (Czytnik.Read())
{
    Lista.Text += Czytnik.GetString(0) + " \r \n";
}
Czytnik.Close();
```

Wynik działania programu



Metadane dostarczane przez data reader

Metoda albo nazwa zmiennej	Opis
Depth	A property that gets the depth of nesting for the current row
FieldCount	A property that holds the number of columns in the current row
GetDataTypeName	A method that accepts an index and returns a string containing the name of the column data type
GetFieldType	A method that accepts an index and returns the .NET Framework type of the object
GetName	A method that accepts an index and returns the name of the specified column
GetOrdinal	A method that accepts a column name and returns the column index
GetSchemaTable	A method that returns column metadata
HasRows	A property that indicates whether the data reader has any rows
RecordsAffected	A property that gets the number of rows changed, inserted, or deleted

Metadane dostarczane przez data reader

```
SqlDataReader rdr = cmd.ExecuteReader();
// get column names
Console.WriteLine( "Column Name:\t(0) {1}", rdr.GetName(0).PadRight(25), rdr.GetName(1));
// get column data types
Console.WriteLine("Data Type:\t(0) {1}",rdr.GetDataTypeName(0).PadRight(25),
    rdr.GetDataTypeName(1));
Console.WriteLine();
while (rdr.Read())
{
    // get column values for all rows
    Console.WriteLine( "\t(0) {1}", rdr.GetString(0).ToString().PadRight(25), rdr.GetString(1));
}
// get number of columns
Console.WriteLine();
Console.WriteLine( "Number of columns in a row: {0}", rdr.FieldCount);
// get info about each column
Console.WriteLine( "{0} is at index {1} * "and its type is: {2}", rdr.GetName(0),
    rdr.GetOrdinal("contactname"), rdr.GetFieldType(0));
Console.WriteLine( "{0} is at index {1} * "and its type is: {2}", rdr.GetName(1),
    rdr.GetOrdinal("contacttitle"), rdr.GetFieldType(1));
rdr.Close();
```

Metadane dostarczane przez data reader

```
SqlDataReader rdr = cmd.ExecuteReader();
// store Employees schema in a data table
DataTable schema = rdr.GetSchemaTable();
// display info from each row in the data table.
// each row describes a column in the database table.
foreach (DataRow row in schema.Rows)
{
    foreach (DataColumn col in schema.Columns)
    Console.WriteLine(col.ColumnName + " = " + row[col]);
    Console.WriteLine("-----");
}
rdr.Close();
```

Wiele tabel w jednym DR

```
// query 1
string sql1 = @"select companyname, contactname from customers where
    companyname like 'A%' ";
// query 2
string sql2 = @_. select firstname, lastname from employees ";
// combine queries
string sql = sql1 + sql2;
// create connection
SqlConnection conn = new SqlConnection(connString);
try
{
    // open connection
    conn.Open();
    // create command
    SqlCommand cmd = new SqlCommand(sql, conn);
```

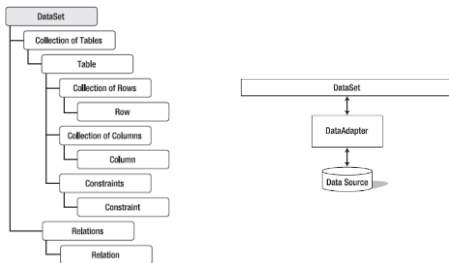
Wiele tabel w jednym DR

```
create data reader
SqlDataReader rdr = cmd.ExecuteReader();
// loop through result sets
do
{
    while (rdr.Read())
    {
        // Print one row at a time
        Console.WriteLine("{0} : {1}", rdr[0], rdr[1]);
    }
    Console.WriteLine("");
} while (rdr.NextResult());
// close data reader
rdr.Close();
}
```

Datasets vs. Data Readers

- Datareader jest przeznaczony do ODCZYTU danych. Jest wydajny i należy go stosować, gdy tylko jest to możliwe. Wydajność jest szczególnie istotna przy dużej ilości danych.
- Dataset jest konieczny w przypadku operacji modyfikujących bazę danych. Data adapter wypełnia dataset korzystając z data readera. Zastosowanie Datasetu w innych warunkach jest marnotrawieniem zasobów
- Jeżeli dane są pobierane z bazy danych, a następnie konwertowane i zapisywane jako XML, należy stosować Datareader

DataSet



DataAdapter

- SqlDataAdapter da = new SqlDataAdapter();
- SqlDataAdapter da = new SqlDataAdapter(cmd);
- SqlDataAdapter da = new SqlDataAdapter(sql, conn);
- SqlDataAdapter da = new SqlDataAdapter(sql, connectionString);

DataSet & DataAdapter

```
// connection string
string connectionString = @"server = .\sqlexpress;integrated security = true;database = northwind";
// query
string sql = @"select productname, unitprice from products where unitprice < 20";
// create connection
SqlConnection conn = new SqlConnection(connectionString);
try(conn.Open());
// create data adapter
SqlDataAdapter da = new SqlDataAdapter(sql, conn);
// create dataset
DataSet ds = new DataSet();
// fill dataset
da.Fill(ds, "products");
// get data table
DataTable dt = ds.Tables["products"];
// display data
foreach (DataRow row in dt.Rows)
{
    foreach (DataColumn col in dt.Columns)
    {
```

DataSet & DataAdapter

```
// connection string
string connectionString = @"server = .\sqlexpress; integrated security = true; database = northwind";
// query 1
string sql1 = @"select * from customers ";
// query 2
string sql2 = @"select * from products where unitprice < 10";
// combine queries
string sql = sql1 + sql2;
// create connection
SqlConnection conn = new SqlConnection(connectionString);
try
{
    // create data adapter
    SqlDataAdapter da = new SqlDataAdapter();
    da.SelectCommand = new SqlCommand(sql, conn);
    // create and fill data set
    DataSet ds = new DataSet();
    da.Fill(ds, "customers");
    // get the data tables collection
```

DataSet & DataAdapter

```
// display data from first data table
// display output header
Console.WriteLine("Results from Customers table:");
Console.WriteLine("CompanyName".PadRight(20) + "ContactName".PadLeft(23) +
    "\n");
// set display filter
string fl = "country = 'Germany'";
// set sort
string srt = "companyname asc";
// display filtered and sorted data
foreach (DataRow row in dtc["customers"].Select(fl, srt))
{
    Console.WriteLine("{0}\t{1}", row["CompanyName"].ToString().PadRight(25), row["Co
        ntactName"]);
}
// display data from second data table
// display output header
Console.WriteLine("\n-----");
Console.WriteLine("Results from Products table:");
Console.WriteLine("ProductID".PadRight(10) + "ProductName".PadLeft(20) + "UnitPrice".PadRight(10) + "\n");
```

DataSet & DataAdapter

```
// display data
foreach (DataRow row in dtc[1].Rows)
{
    Console.WriteLine("{0}\t{1}",
        row["productname"].ToString().PadRight(25),
        row["unitprice"]);
}
catch (Exception e)
{
    Console.WriteLine("Error: " + e);
}
finally
{
    // close connection
    conn.Close();
}
```

DataView

```
// create data view
DataView dv = new DataView(dt, "country = 'Germany'",
    "country", DataViewRowState.CurrentRows);
// display data from data view
foreach (DataRowView drv in dv)
{
    for (int i = 0; i < dv.Table.Columns.Count; i++)
        Console.Write(drv[i] + "\t");
    Console.WriteLine();
}
```

Stany wierszy w Data View

DataViewRowState	Opis
Added	Nowy wiersz
CurrentRows	Aktualny wiersz, zawierający oryginalne, nowe oraz zmienione dane
Deleted	Skasowany wiersz
ModifiedCurrent	Aktualny stan zmodyfikowanego wiersza
ModifiedOriginal	Oryginalny stan zmodyfikowanego wiersza
None	Pusty
OriginalRows	Oryginalne wiersze, w tym skasowane oraz niezmienione
Unchanged	Wiersze, które nie zostały zmienione

Obiekt DataSet

- W odróżnieniu od przedstawionych wcześniej obiektów nie musi mieć związku z zewnętrznym źródłem danych
- Jest pamięciowym relacyjnym zasobnikiem danych
- Może przechowywać dane w postaci XML

Tworzenie obiektu DataSet

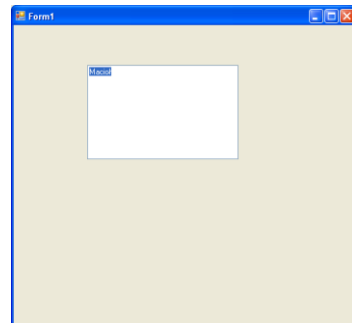
```
DataSet Studia = new DataSet();
DataTable Studenci = new DataTable();
Studia.Tables.Add(Studenci);
DataColumn Nowa = new DataColumn();
Nowa.ColumnName = "IdStudenta";
Nowa.DataType = Type.GetType("System.Int32");
Nowa.Unique = true;
Nowa.AutoIncrement = true;
Studenci.Columns.Add(Nowa);
```

```

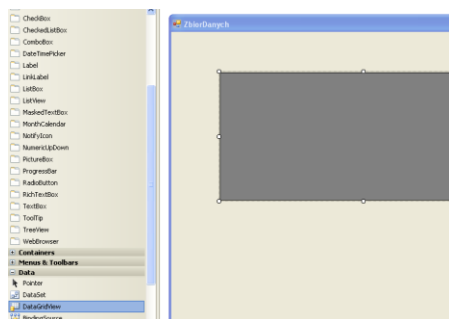
Nowa = new DataColumn();
Nowa.ColumnName = "Nazwisko";
Nowa.DataType = Type.GetType("System.String");
Studenci.Columns.Add(Nowa);
Studenci.Columns.Add("Imie", Type.GetType("System.String"));
DataRow wiersz;
wiersz = Studenci.NewRow();
wiersz["Nazwisko"] = "Maciol";
wiersz["Imie"] = "Andrzej";
Studenci.Rows.Add(wiersz);
Student.Text = Studenci.Rows[0]["Nazwisko"].ToString();

```

Tworzenie obiektu DataSet



Użycie obiektu DataGridView



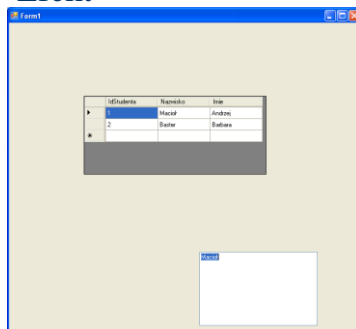
Połączenie obiektów

```

DataRow wiersz;
wiersz = Studenci.NewRow();
wiersz["Nazwisko"] = "Maciol";
wiersz["Imie"] = "Andrzej";
Studenci.Rows.Add(wiersz);
wiersz = Studenci.NewRow();
wiersz["Nazwisko"] = "Baster";
wiersz["Imie"] = "Barbara";
Studenci.Rows.Add(wiersz);
SiatkaDanych1.DataSource = Studenci;

```

Efekt



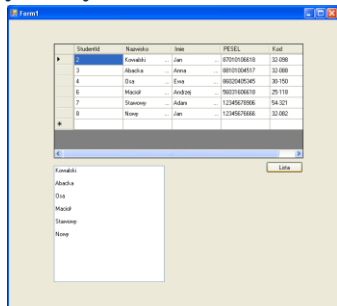
Wykorzystanie obiektu DataSet

```

private void Lista_Click(object sender, EventArgs e)
{
    for (int i = 0; i < Studenci.Rows.Count; i++)
    {
        DataRow row = Studenci.Rows[i];
        Student.Text += row["Nazwisko"].ToString() + "\r" + "\n";
    }
}

```

Wykorzystanie obiektu DataSet

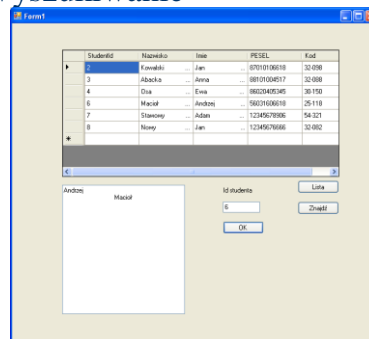


Wyszukiwanie wg klucza

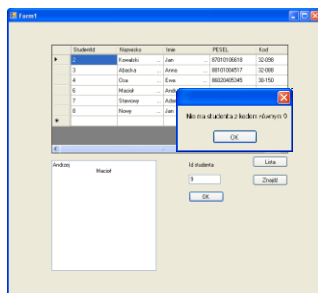
```
private void Znajdz_Click(object sender, EventArgs e)
{
    DataColumn[] pk = new DataColumn[1];
    pk[0] = Studenci.Columns[0];
    Studenci.PrimaryKey = pk;
    label1.Visible=true;
    idStudenta.Visible=true;
    OK.Visible=true;
}
```

```
private void OK_Click(object sender, EventArgs e)
{
    Object Indeks = idStudenta.Text;
    DataRow ZnalezonyWiersz;
    ZnalezonyWiersz = Studenci.Rows.Find(Indeks);
    if (ZnalezonyWiersz == null)
    {
        MessageBox.Show("Nie ma studenta z kodem równym " +
        Indeks.ToString());
    }
    else
    {
        Student.Text = ZnalezonyWiersz["Imie"].ToString() +
        ZnalezonyWiersz["Nazwisko"].ToString();
    }
}
```

Wyszukiwanie



Wynik



Klucz wielokrotny

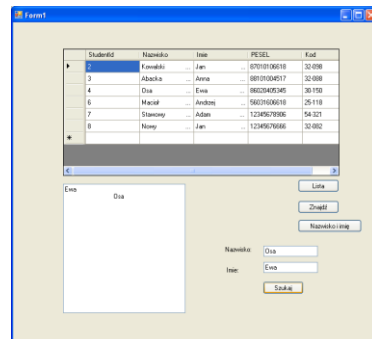
```
private void NazwiskoImie_Click(object sender, EventArgs e)
{
    DataColumn[] pk = new DataColumn[2];
    pk[0] = Studenci.Columns["Nazwisko"];
    pk[1] = Studenci.Columns["Imie"];
    Studenci.PrimaryKey = pk;
    label2.Visible = true;
    label3.Visible = true;
    Nazwisko.Visible = true;
    Imie.Visible = true;
    Szukaj.Visible = true;
}
```

```

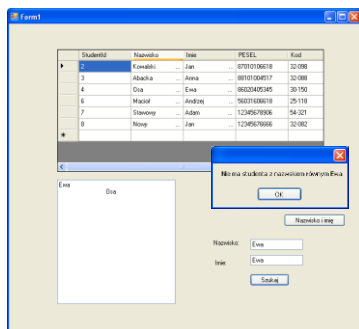
private void Szukaj_Click(object sender, EventArgs e)
{
    Object[] Indeks = new Object[2];
    Indeks[0] = Nazwisko.Text;
    Indeks[1] = Imie.Text;
    DataRow ZnalezonyWiersz;
    ZnalezonyWiersz = Studentci.Rows.Find(Indeks);
    if (ZnalezonyWiersz == null)
    {
        MessageBox.Show("Nie ma studenta z nazwiskiem równym " +
            Nazwisko.Text);
    }
    else
    {
        Student.Text = ZnalezonyWiersz["Imie"].ToString() +
            ZnalezonyWiersz["Nazwisko"].ToString();
    }
}

```

Wynik



Wynik



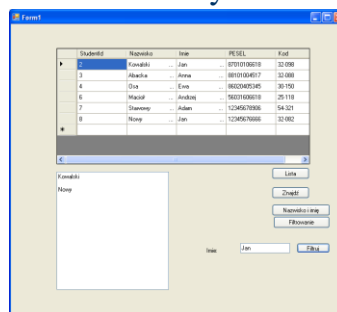
Filtrowanie danych

```

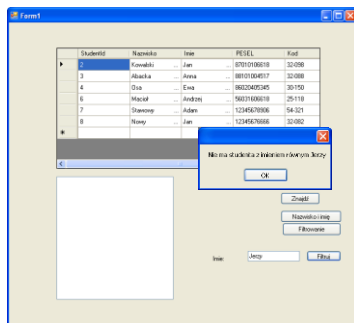
DataRow[] ZnalezonyWiersz;
private void Filtruj_Click(object sender, EventArgs e)
{
    string wyrazenie = "Imie =" + Imie.Text + "";
    ZnalezonyWiersz = Studentci.Select(wyrazenie);
    if (ZnalezonyWiersz.Count() == 0)
    {
        MessageBox.Show("Nie ma studenta z imieniem równym " +
            Imie.Text);
    }
    else
    {
        for (int i = 0; i < ZnalezonyWiersz.Count(); i++)
        {
            DataRow row = ZnalezonyWiersz[i];
            Student.Text += row["Nazwisko"].ToString() + "r" + "n";
        }
    }
}

```

Filtrowanie danych



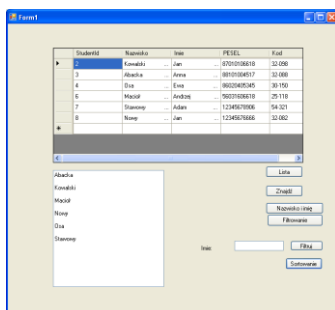
Wynik



Sortowanie

```
private void Sortowanie_Click(object sender, EventArgs e)
{
    ZnalezionyWiersz = Studenci.Select("", "Nazwisko ASC");
    for (int i = 0; i < ZnalezionyWiersz.Count(); i++)
    {
        DataRow row = ZnalezionyWiersz[i];
        Student.Text += row["Nazwisko"].ToString() + "\r" + "\n";
    }
}
```

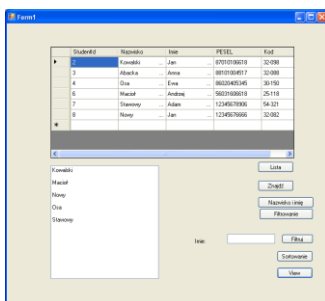
Wynik



Obiekt DataView

```
private void View_Click(object sender, EventArgs e)
{
    DataView widok = new DataView(ds.Tables["Studenci"],
    "Nazwisko > B", "Nazwisko", DataViewRowState.OriginalRows);
    foreach (DataRowView _w in widok)
    {
        Student.Text += _w["Nazwisko"].ToString() + "\r" + "\n";
    }
}
```

Wynik



Tworzenie relacji

```
private void button1_Click(object sender, EventArgs e)
{
    DataColumn rodzic = Studenci.Columns[0];
    DataColumn dziecko = Oceny.Columns[0];
    DataRelation Relacja = new DataRelation("StudenciZaliczenia",
    rodzic, dziecko);
    ds.Relations.Add(Relacja);
    rodzic = Przedmioty.Columns["PrzedmiotId"];
    dziecko = Oceny.Columns["PrzedmiotId"];
    Relacja = new DataRelation("PrzedmiotyZaliczenia", rodzic, dziecko);
    ds.Relations.Add(Relacja);
}
```

```

private void button2_Click(object sender, EventArgs e)
{
    DataRow rodzic;
    string Nazwisko, Imie, Przedmiot;
    foreach (DataRow wiersz in Oceny.Rows)
    {
        rodzic=wiersz.GetParentRow("StudenciZaliczenia");
        Nazwisko= rodzic["Nazwisko"].ToString();
        Imie = rodzic["Imie"].ToString();
        rodzic=wiersz.GetParentRow("PrzedmiotyZaliczenia");
        Przedmiot = rodzic["Przedmiot"].ToString();
        listBox1.Items.Add(Imie.Trim() + " " + Nazwisko.Trim() + " - " +
        Przedmiot.Trim() + " " + wiersz["Data"] + " " + " +
        wiersz["Ocena"].ToString());
    }
}

```

Wynik

Wprowadzenie pól combo

```

dataGridView1.DataSource = Oceny;
DataGridViewComboBoxColumn KolumnaCombo = new
DataGridViewComboBoxColumn();
KolumnaCombo.DataSource = Studenci;
KolumnaCombo.DisplayMember = "Nazwisko";
KolumnaCombo.ValueMember = "StudentId";
KolumnaCombo.HeaderText = "Student";
KolumnaCombo.DataPropertyName = "StudentId";
dataGridView1.Columns.Add(KolumnaCombo);
dataGridView1.Columns[0].Visible = false;

```

Wprowadzenie pól combo

```

DataGridViewComboBoxColumn KolumnaCombo1 = new
DataGridViewComboBoxColumn();
KolumnaCombo1.DataSource = Przedmioty;
KolumnaCombo1.DisplayMember = "Przedmiot";
KolumnaCombo1.ValueMember = "PrzedmiotId";
KolumnaCombo1.HeaderText = "Przedmiot";
KolumnaCombo1.DataPropertyName = "PrzedmiotId";
dataGridView1.Columns.Add(KolumnaCombo1);
dataGridView1.Columns[1].Visible = false;
dataGridView1.Columns[4].DisplayIndex = 0;
dataGridView1.Columns[5].DisplayIndex = 1;

```

Wprowadzenie pól combo

```

Dim KolumnaCombo1 As New
DataGridViewComboBoxColumn
KolumnaCombo1.DataSource = Studia.Tables("Przedmioty")
KolumnaCombo1.DisplayMember = "Przedmiot"
KolumnaCombo1.ValueMember = "IdPrzedmiotu"
KolumnaCombo1.HeaderText = "Przedmiot"
KolumnaCombo1.DataPropertyName = "IdPrzedmiotu"
DataGridView1.Columns.Add(KolumnaCombo1)
DataGridView1.Columns(1).Visible = False
DataGridView1.Columns(4).DisplayIndex = 0
DataGridView1.Columns(5).DisplayIndex = 1
End Sub

```

Wynik

Ograniczenia w DataSet

```
ds.Tables.Add("Ocena");
ds.Tables["Ocena"].Columns.Add("Ocena",
Type.GetType("System.String"));
DataRow wiersz;
wiersz = ds.Tables["Ocena"].NewRow();
wiersz["Ocena"] = "2,0";
ds.Tables["Ocena"].Rows.Add(wiersz);
wiersz = ds.Tables["Ocena"].NewRow();
wiersz["Ocena"] = "3,0";
ds.Tables["Ocena"].Rows.Add(wiersz);
.....
ForeignKeyConstraint OcenaZaliczenieFK = new
ForeignKeyConstraint("OcenaZaliczenieFK",
ds.Tables["Ocena"].Columns["Ocena"],
ds.Tables["Oceny"].Columns["Ocena"]);
OcenaZaliczenieFK.UpdateRule = Rule.Cascade;
ds.Tables["Oceny"].Constraints.Add(OcenaZaliczenieFK);
dataGridView2.DataSource = ds.Tables["Ocena"];
```

Obiekt DataAdapter

- Pośredniczy między obiektem DataSet a rzeczywistym źródłem danych
- Ma dwie metody: Fill i Update
- Oraz cztery właściwości: SelectCommand, InsertCommand, UpdateCommand i DeleteCommand
- Występuje w wersjach SQL, OleDb i Oledbc

Wypełnianie obiektu DataSet ze źródła danych

Źródło danych

StudentId	Nazwisko	Imie	PESEL	Kod
2	Kowalski	Jan	87010106618	32-098
3	Abacka	Anna	88101004517	32-088
4	Osa	Ewa	86020405345	30-150
6	Macioł	Andrzej	56031606618	25-118
7	Stanowcy	Adam	12345678901	54-321

Column Name	Data Type	Allow Nulls
StudentId	bigint	<input type="checkbox"/>
Nazwisko	nchar(100)	<input checked="" type="checkbox"/>
Imie	nchar(100)	<input checked="" type="checkbox"/>
PESEL	nchar(11)	<input checked="" type="checkbox"/>
Kod	nchar(6)	<input checked="" type="checkbox"/>

Połączenie ze źródłem

```
DataSet ds = new DataSet();
DataTable Studenci = new DataTable();
private void Form1_Load(object sender, EventArgs e)
{
    //Tworzenie połączenia (1)
    SqlConnection conn = new SqlConnection(
        "Server=Stacjonarny\\SqlExpress;Database=Test;" +
        "Integrated Security=True");
    // Ciąg znaków połączenia (2)
    string query = "SELECT * from Studenci";
    // Tworzenie adaptera danych (3)
    SqlDataAdapter da = new SqlDataAdapter(query, conn);
    // Tworzenie tabeli obiektu DataSet i załadowanie jej danymi
    da.Fill(ds, "Studenci");
    SiatkaDanvch1.DataSource = ds.Tables["Studenci"];
}
```

Wynik

Pobieranie wielu tabel w oparciu o wiele kwerend – drugie źródło

PrzedmiotId	Przedmiot
1	matematyka
2	fizyka
3	programowanie komputerowe
4	bazy danych
*	Wszystkie

Column Name	Data Type	Allow Nulls
PrzedmiotId	bigint	<input type="checkbox"/>
Przedmiot	nchar(100)	<input checked="" type="checkbox"/>

Nowy grid dla „Przedmiotów”

```
SqlConnection conn = new SqlConnection(
    "Server=Stacjonarny\\SqlExpress;Database=Test;" +
    "Integrated Security=True");
string query = "SELECT * from Studenci";
SqlDataAdapter da = new SqlDataAdapter(query, conn);
da.Fill(ds, "Studenci");
query = "SELECT * from Przedmioty";
SqlDataAdapter dap = new SqlDataAdapter(query, conn);
dap.Fill(ds, "Przedmioty");
query = "SELECT * from Oceny";
SqlDataAdapter dao = new SqlDataAdapter(query, conn);
dao.Fill(ds, "Oceny");
```

```

Private Sub Pobierz_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Pobierz.Click
    Dim Connection As New SqlConnection
    Dim Adapter As SqlDataAdapter
    Dim sql As String
    Connection.ConnectionString = _
        "Data Source=STACJONARNY\SQLEXPRESS;Integrated
Security=True;Initial Catalog=Test"
    Studia.Tables.Add(Studenci)
    Sql = "SELECT * FROM Studenci WHERE
(CEILING(CAST(SUBSTRING(PESEL, LEN(PESEL), 1) AS float) / 2)
- CAST(SUBSTRING(PESEL, LEN(PESEL), 1) AS float) / 2 = 0)"
    Adapter = New SqlDataAdapter(sql, Connection)
    If Connection.State = ConnectionState.Closed Then
        Connection.Open()
    End If
    Studia.Clear()
    Adapter.Fill(Studia, "Studenci")
    DataGridView1.DataSource = Studia.Tables("Studenci")

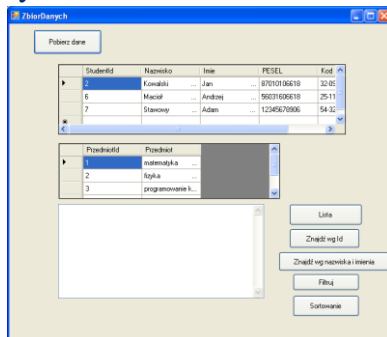
```

```

Studia.Tables.Add(Przedmioty)
Adapter.SelectCommand.CommandText = "SELECT * FROM
Przedmioty"
Adapter.Fill(Studia, "Przedmioty")
PrzedmiotyGrid.DataSource = Studia.Tables("Przedmioty")
End Sub

```

Wynik



Uwaga

- Użycie jednego „adaptera” do wielu tabel uniemożliwia rozsądne ich aktualizowanie

Pobieranie danych przy użyciu procedury składowanej

```

CREATE PROCEDURE [dbo].[WybieranieStudentow]
AS
SELECT StudentId, Nazwisko, Imie, PESEL, Kod
FROM Studenci

```

Aktualizacja danych z dataset przy użyciu procedury składowanej - dopisywanie

```

USE [Test]
GO
/***** Object: StoredProcedure [dbo].[Dopisanie] Script Date:
12/03/2007 14:36:57 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[DopisaniePracownikow] @Nazwisko
char(50), @Imie char(50), @Tytul char(10)
AS INSERT INTO Prowadzacy (Nazwisko, Imie, Tytul)
VALUES (@Nazwisko, @Imie, @Tytul)

```

```

Private Sub ZapiszP_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ZapiszP.Click
    Dim SQLConn As New SqlConnection
    SQLConn.ConnectionString = _
        "Data Source=STACJONARNY\sqlexpress;Initial
Catalog=Test;Integrated Security=True"
    SQLConn.Open()
    Polecenie.Connection = SQLConn
    Polecenie.CommandType = CommandType.StoredProcedure
    Polecenie.CommandText = "DopisaniePracownikow"
    Polecenie.Parameters.Add(New SqlParameter("@Nazwisko",
SqlDbType.Char))
    Polecenie.Parameters("@Nazwisko").Direction =
ParameterDirection.Input
    Polecenie.Parameters.Add(New SqlParameter("@Imie",
SqlDbType.Char))
    Polecenie.Parameters("@Imie").Direction =
ParameterDirection.Input

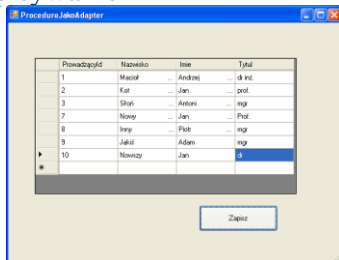
```

```

Polecenie.Parameters.Add(New SqlParameter("@Tytul",
SqlDbType.Char))
    Polecenie.Parameters("@Tytul").Direction =
ParameterDirection.Input
    Dim Widok As New DataView(Me.TestDataSet2.Prowadzacy)
    Widok.RowStateFilter = DataViewRowState.Added
    Dim i, wynik As Integer
    For i = 0 To Widok.Count - 1
        Polecenie.Parameters("@Nazwisko").Value =
Widok(i)("Nazwisko")
        Polecenie.Parameters("@Imie").Value = Widok(i)("Imie")
        Polecenie.Parameters("@Tytul").Value = Widok(i)("Tytul")
        wynik = Polecenie.ExecuteNonQuery
    Next i
End Sub

```

Aktualizacja danych z dataset przy użyciu procedury składowanej - dopisywanie



Wynik

ProwadzacyId	Nazwisko	Imie	Tytul
1	Maciej	Andrzej	d. inż.
2	Kat	Jan	prof.
3	Stasi	Antoni	mgr
7	Nowy	Jan	prof.
8	Imy	Piotr	mgr
9	Jakub	Adam	mgr
10	Nowy	Jan	d.
11	Nowy	Jan	d.

Aktualizacja danych z dataset przy użyciu procedury składowanej - zmiana

```

CREATE PROCEDURE [dbo].[ZmianaPracownikow] @Nazwisko
char(50), @Imie char(50), @Tytul char(10), @ProwadzacyId bigint
AS
UPDATE Prowadzacy
SET Nazwisko = @Nazwisko,
    Imie = @Imie,
    Tytul = @Tytul
WHERE
ProwadzacyId = @ProwadzacyId

```

```

Polecenie.Parameters.Add(New SqlParameter("@ProwadzacyId",
SqlDbType.Char))
Polecenie.Parameters("@ProwadzacyId").Direction =
ParameterDirection.Input
.....
Polecenie.CommandText = "ZmianaPracownikow"
Widok.RowStateFilter = DataViewRowState.ModifiedCurrent
For i = 0 To Widok.Count - 1
    Polecenie.Parameters("@Nazwisko").Value =
Widok(i)("Nazwisko")
    Polecenie.Parameters("@Imie").Value = Widok(i)("Imie")
    Polecenie.Parameters("@Tytul").Value = Widok(i)("Tytul")
    Polecenie.Parameters("@ProwadzacyId").Value =
Widok(i)("ProwadzacyId")
    wynik = Polecenie.ExecuteNonQuery
Next i

```

Stan wyjściowy

ProwadzącyId	Nazwisko	Imię	Tytuł
1	Macioł	Andrzej	dr inż.
2	Kot	Jan	prof.
3	Skoń	Antoni	mgr.
7	Nowy	Jan	Prof.
8	Inny	Piotr	mgr.
9	Zmieniony	Adam	mgr.
10	Nowszy	Jan	dr.

Zmiany

ProwadzącyId	Nazwisko	Imię	Tytuł
1	Macioł	Andrzej	dr inż.
2	Kot	Jan	prof.
3	Skoń	Antoni	mgr.
7	Nowy	Jan	Prof.
8	Inny	Piotr	mgr.
9	Zmieniony	Adam	mgr.
10	Nowszy	Jan	dr.

Wynik

ProwadzącyId	Nazwisko	Imię	Tytuł
1	Macioł	Andrzej	dr inż.
2	Kot	Jan	prof.
3	Skoń	Antoni	mgr.
7	Nowy	Jan	Prof.
8	Inny	Piotr	mgr.
9	Zmieniony	Adam	mgr.
10	Nowszy	Jan	dr.
NULL	NULL	NULL	NULL

Aktualizacja danych z dataset przy użyciu procedury składowanej – usuwanie rekordów

```
CREATE PROCEDURE [dbo].[UsuwaniePracownikow]
@ProwadzącyId bigint
AS
DELETE FROM Prowadzący
WHERE
ProwadzącyId = @ProwadzącyId
```

Usuwanie rekordu Zmieniony

```
Polecenie.CommandText = "UsuwaniePracownikow"
Widok.RowStateFilter = DataViewRowState.Deleted
For i = 0 To Widok.Count - 1
    Polecenie.Parameters.Clear()
    Polecenie.Parameters.Add(New
SqlParameter("@ProwadzącyId", SqlDbType.Char))
    Polecenie.Parameters("@ProwadzącyId").Direction =
ParameterDirection.Input
    Polecenie.Parameters("@ProwadzącyId").Value =
Widok(i)("ProwadzącyId")
    wynik = Polecenie.ExecuteNonQuery
Next i
```

ProwadzącyId	Nazwisko	Imię	Tytuł
1	Macioł	Andrzej	dr inż.
2	Kot	Jan	prof.
3	Skoń	Antoni	mgr.
7	Nowy	Jan	Prof.
8	Inny	Piotr	mgr.
10	Nowszy	Jan	dr.

Wynik

	ProwadzacyId	Nazwisko	Imie	Tytul
▶	1	Maciol ...	Andrzej ...	dr inż.
	2	Kot ...	Jan ...	prof.
	3	Słofi ...	Antoni ...	mgr
	7	Nowy ...	Jan ...	Prof.
	8	Inny ...	Piotr ...	mgr
	10	Nowszy ...	Jan ...	dr
*	NULL	NULL	NULL	NULL

Aktualizacja danych przy użyciu CommandBuildera

Imports System.data.SqlClient

```
Public Class PzAdapterem
    Dim Polecenie As New SqlCommand
    Dim Adapter As SqlDataAdapter
    Dim Builder As SqlCommandBuilder
    Dim Prowadzacy As DataSet
    Dim SqlConnection As New SqlConnection
```

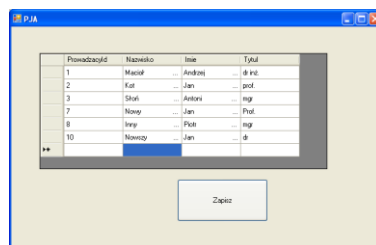
```
Private Sub PzAdapterem_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
```

```
    SqlConnection.ConnectionString = _
        "Data Source=STACJONARNY\sqlexpress;Initial
        Catalog=Test;Integrated Security=True"
    SqlConnection.Open()
    Prowadzacy = New DataSet
    Adapter = New SqlDataAdapter("Select * from Prowadzacy",
    SqlConnection)
    Builder = New SqlCommandBuilder(Adapter)
    Adapter.Fill(Prowadzacy, "a")
    DataGridView1.DataSource = Prowadzacy.Tables("a")
End Sub
```

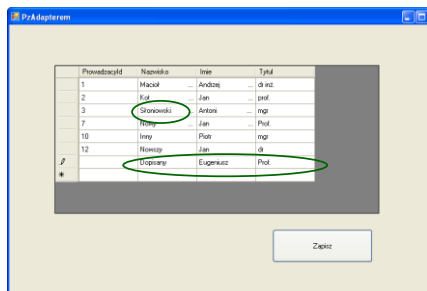
```
Private Sub Zapisz_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Zapisz.Click
```

```
    Adapter.Update(Prowadzacy, "a")
End Sub
End Class
```

Aktualizacja danych przy użyciu CommandBuildera - stan wyjściowy



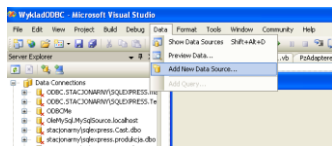
Poprawki



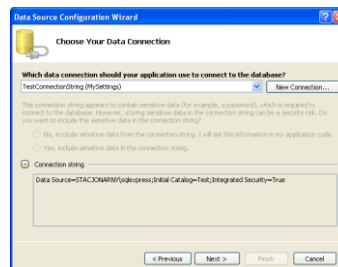
Wynik

STACJONARNY...	6.Prowadzacy	STACJONARNY(S...	Query7.sql*	not con...
ProwadzacyId	Nazwisko	Imie	Tytul	
1	Maciol	Andrzej	dr inż.	
2	Kot	Jan	prof.	
3	Słoniowski	Antoni	mgr	
7	Nowy	Jan	Prof.	
10	Inny	Piotr	mgr	
12	Nowszy	Jan	dr	
13	Dopisany	Eugeniusz	Prof.	
**	NULL	NULL	NULL	

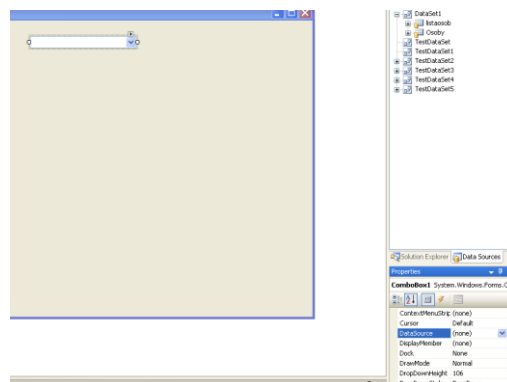
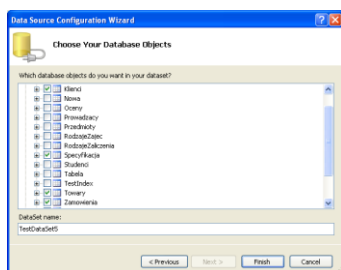
Tworzenie źródła danych przy użyciu kreatora



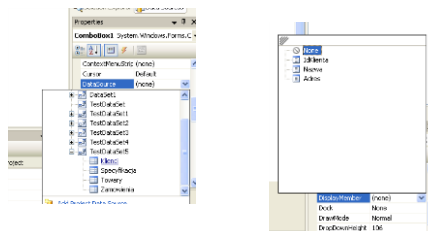
Wybór połączenia



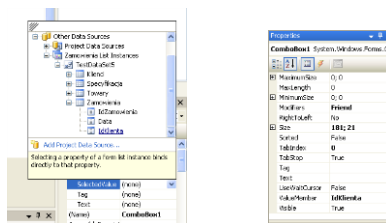
Wybór obiektów dla dataset



Powiązanie pola combo



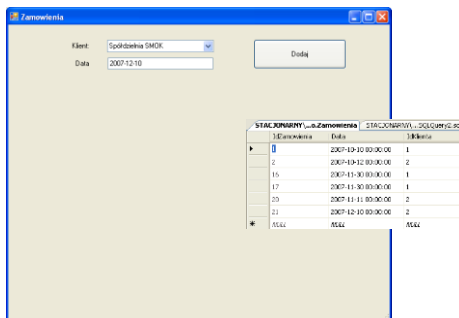
Powiązanie pola combo



```
Imports System.Data.SqlClient
Public Class Zamowienia
Private Sub Zamowienia_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    'TODO: This line of code loads data into the
    'TestDataSet5.Zamowienia' table. You can move, or remove it, as needed.
    Me.ZamowieniaTableAdapter.Fill(Me.TestDataSet5.Zamowienia)
    'TODO: This line of code loads data into the 'TestDataSet5.Klienci'
    table. You can move, or remove it, as needed.
    Me.KlienciTableAdapter.Fill(Me.TestDataSet5.Klienci)
    Me.TestDataSet5.Zamowienia.Rows.Add()
    Me.BindingContext(Me.ZamowieniaBindingSource1).Position =
    Me.TestDataSet5.Zamowienia.Rows.Count - 1
End Sub
```

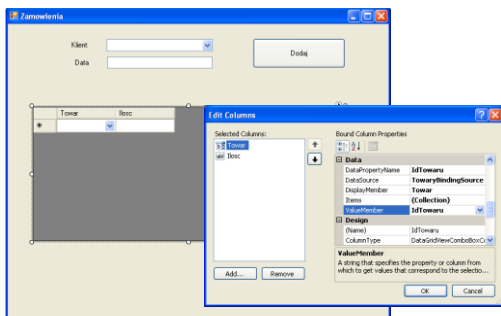
```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    Me.BindingContext(Me.ZamowieniaBindingSource1).Position -= 1
    Me.BindingContext(Me.ZamowieniaBindingSource1).Position += 1
    Me.ZamowieniaTableAdapter.Update(Me.TestDataSet5.Zamowienia)
End Sub
End Class
```

Wynik



Dodanie oryginalnego adaptera

Dodajemy Grid



Przygotowanie procedury

```
USE [Test]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[InsertZamowienie]
    @IdKlienta char(255),
    @Data datetime,
    @Identity bigint out
AS
INSERT INTO Zamowienia (Data, IdKlienta) VALUES(@Data,
@IdKlienta)
SET @Identity = SCOPE_IDENTITY()
```



```

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
    Me.BindingContext(Me.ZamowieniaBindingSource1).Position -= 1
    Me.BindingContext(Me.ZamowieniaBindingSource1).Position += 1
    Dim Adapter As New SqlDataAdapter
    Dim Conn As New SqlConnection
    Dim Builder As SqlCommandBuilder
    Conn.ConnectionString = "Data
Source=STACJONARNY\squlexpress;Initial Catalog=Test;Integrated
Security=True"
    Adapter = New SqlDataAdapter("Select * from Zamowienia", Conn)
    Adapter.InsertCommand = New SqlCommand("InsertZamowienie",
Conn)
    Adapter.InsertCommand.CommandType =
CommandType.StoredProcedure
    Adapter.InsertCommand.Parameters.Add( _
"@Data", SqlDbType.DateTime, 0, "Data")
    Adapter.InsertCommand.Parameters.Add( _
"@IdKlienta", SqlDbType.BigInt, 0, "IdKlienta")

```

```

Dim parameter As SqlParameter =
Adapter.InsertCommand.Parameters.Add( _
"@Identity", SqlDbType.BigInt, 0, "IdZamowienia")
parameter.Direction = ParameterDirection.Output
Builder = New SqlCommandBuilder(Adapter)
Adapter.Update(Me.TestDataSet5.Zamowienia)
Dim x As Integer
x = Adapter.InsertCommand.Parameters(2).Value

```

```

Me.TestDataSet5.Specyfikacja.Columns("IdZamowienia").DefaultValue
= x
End Sub

```

```

Private Sub Button2_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button2.Click
    Me.SpecyfikacjaTableAdapter.Update(TestDataSet5.Specyfikacja)
    Me.Close()
End Sub

```

Wprowadzanie danych

Wynik

STACJONARNY...Zamowienia - Summary			
IdZamowienia	Data	Ilość	
1	2007-10-10 00:00:00	1	
2	2007-10-12 00:00:00	2	
16	2007-11-30 00:00:00	1	
17	2007-11-30 00:00:00	1	
20	2007-11-11 00:00:00	2	
21	2007-12-10 00:00:00	2	
22	2007-12-10 00:00:00	2	
23	2007-12-20 00:00:00	3	

STACJONARNY...Specyfikacja - Summary			
IdZamowienia	Ilość	Specyfikacja	
2	16		
23	3		
2007	2007	2007	

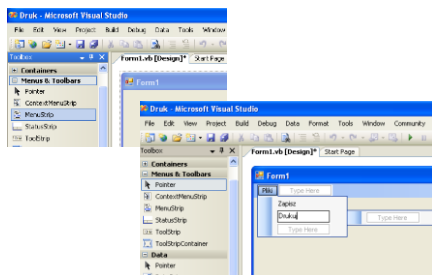
C#

```

{
    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
            'testDataSet.Studenci' table. You can move, or remove it, as needed.
            this.studenciTableAdapter.Fill(this.testDataSet.Studenci);
        }
        private void button1_Click(object sender, EventArgs e)
        {
            this.studenciTableAdapter.Update(this.testDataSet.Studenci);
        }
    }
}

```

Drukowanie - przygotowanie



Tworzenie i pokazywanie instancji formularza drukuj

Public Class Form1

Private Sub DrukujToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles DrukujToolStripMenuItem.Click

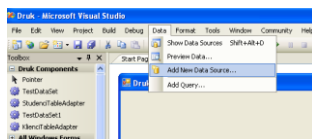
Dim Druk As New Drukuj

Druk.ShowDialog()

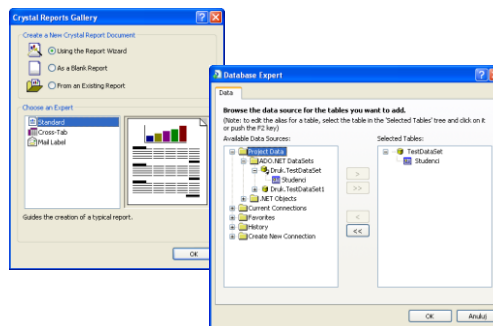
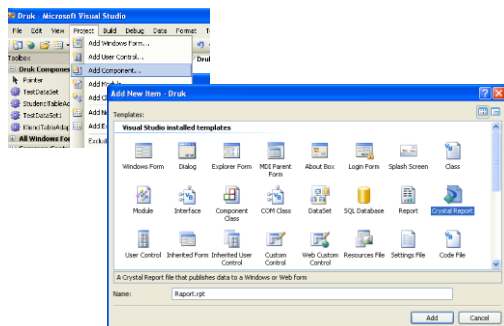
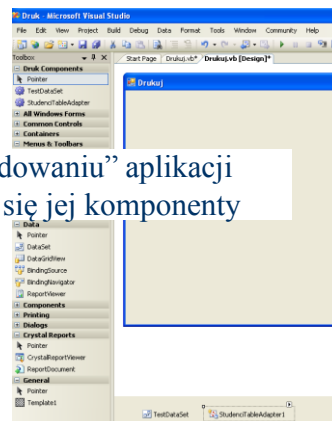
End Sub

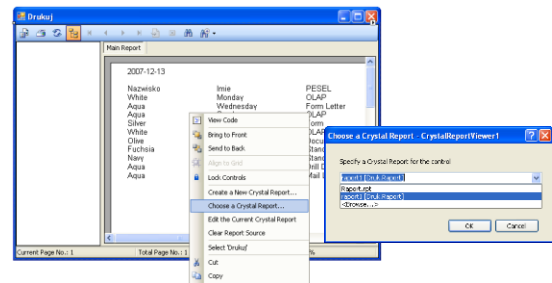
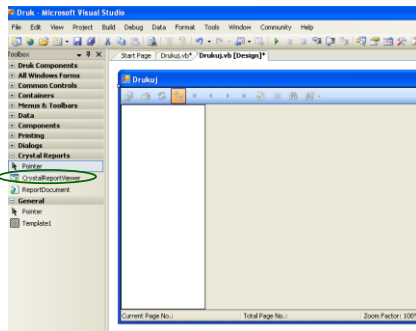
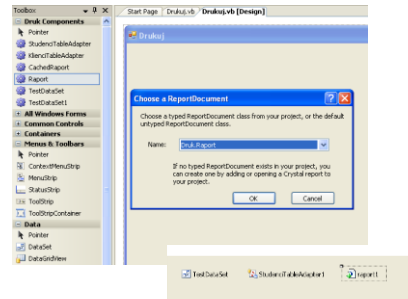
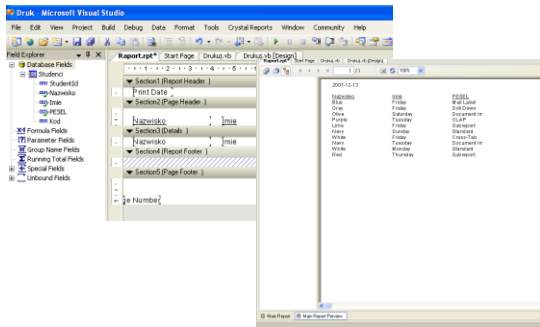
End Class

Tworzenie nowego źródła danych



Po „zbudowaniu” aplikacji pojawią się jej komponenty



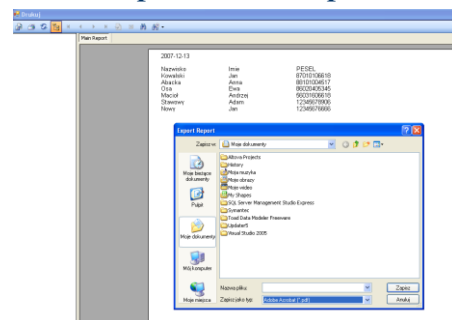


Ładowanie danych i wyświetlanie widoku raportu

Public Class Drukuj

```
Private Sub Drukuj_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    Me.StudenciTableAdapter1.Fill(TestDataSet.Studenci)
    Me.raport1.SetDataSource(TestDataSet)
End Sub
End Class
```

Eksportowanie raportu



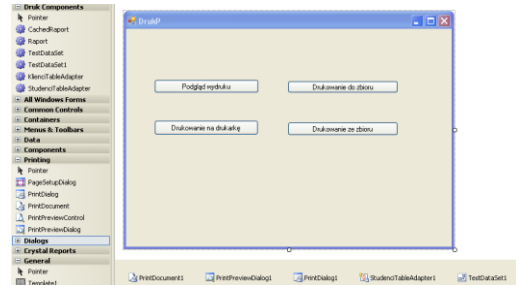
Drukowanie z VB

- Do drukowania wykorzystuje się obiekty z przestrzeni nazw importowanej poleceniem:

Imports System.Drawing.Printing

- Oraz obiekty Printing: PrintDialog, PrintPreviewDialog itd.

Przygotowanie do druku



Przygotowanie dokumentu do druku

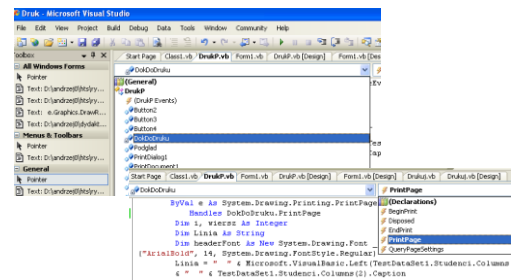
Imports System.Drawing.Printing

```
Private Sub DrukP_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    Me.StudenciTableAdapter1.Fill(TestDataSet1.Studenci)
End Sub
```

```
Private WithEvents DokDoDruku As New Printing.PrintDocument
```

```
Private Sub Podglad_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles Podglad.Click
    PrintPreviewDialog1.Document = DokDoDruku
    PrintPreviewDialog1.ShowDialog()
End Sub
```

Przygotowanie obsługi zdarzenia



```
Private Sub document_PrintPage(ByVal sender As Object, _
ByVal e As System.Drawing.Printing.PrintPageEventArgs) _
Handles DokDoDruku.PrintPage
    Dim i, wiersz As Integer
    Dim Linia As String
    Dim headerFont As New System.Drawing.Font _
("ArialBold", 14, System.Drawing.FontStyle.Regular)
    Linia = " " &
Microsoft.VisualBasic.Left(TestDataSet1.Studenci.Columns(2).Caption,
8) _
& " " & TestDataSet1.Studenci.Columns(1).Caption
```

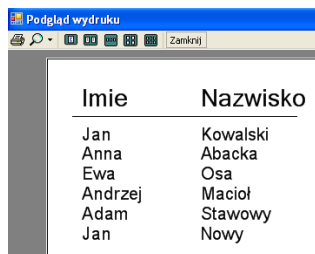
wiersz = 20

```
e.Graphics.DrawString(Linia, headerFont, _
System.Drawing.Brushes.Black, 14, wiersz)
Dim printFont As New System.Drawing.Font _
("Arial", 10, System.Drawing.FontStyle.Regular)
wiersz = wiersz + printFont.GetHeight(e.Graphics)
e.Graphics.DrawLine(Pens.Black, 20, wiersz + 10, 200, wiersz +
10)
```

```
For i = 0 To TestDataSet1.Studenci.Rows.Count - 1
    wiersz = wiersz + printFont.GetHeight(e.Graphics)
    Linia = " " &
Microsoft.VisualBasic.Left(TestDataSet1.Studenci.Rows(i)("Imie"), 10)
& ControlChars.Tab & TestDataSet1.Studenci.Rows(i)("Nazwisko")
    e.Graphics.DrawString(Linia, printFont, _
System.Drawing.Brushes.Black, 10, wiersz)
```

```
Next
End Sub
```

Wynik



Imie	Nazwisko
Jan	Kowalski
Anna	Abacka
Ewa	Osa
Andrzej	Macioł
Adam	Stawowy
Jan	Nowy

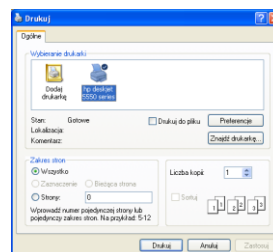
Drukowanie z użyciem dialogu

- Obsługa zdarzenia
DokDoDruku.PrintPage pozostaje bez zmian

Obsługa przycisku

```
Private Sub Drukarka_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Drukarka.Click
    PrintDialog1.AllowSomePages = True
    PrintDialog1.ShowHelp = True
    PrintDialog1.Document = DokDoDruku
    Dim result As DialogResult = PrintDialog1.ShowDialog()
    If (result = DialogResult.OK) Then
        DokDoDruku.Print()
    End If
End Sub
```

Wynik



Drukowanie do pliku w formacie ekranu

- W tym trybie nie ma możliwości formatowania czcionki

```
Imports System.Drawing.Printing
Imports System.IO
```

```
Private Sub DoZboru_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles DoZboru.Click
    SaveFileDialog1.InitialDirectory = "d:\\"
    SaveFileDialog1.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*"
    SaveFileDialog1.FilterIndex = 2
    SaveFileDialog1.RestoreDirectory = True
    SaveFileDialog1.AddExtension = True
    SaveFileDialog1.CheckFileExists = True
    If SaveFileDialog1.ShowDialog() = DialogResult.OK Then
        If Not (SaveFileDialog1.FileName(0) = Nothing) Then
            FileOpen(1, SaveFileDialog1.FileName, OpenMode.Output) '
            Open file for output.
            Print(1, "Lista studentów") ' Print text to file.
            Dim i As Integer
```

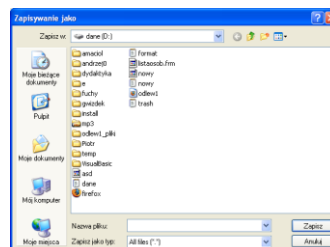
```

PrintLine(1)
PrintLine(1)
PrintLine(1, "Imie", TAB(), "Nazwisko")
PrintLine(1,
"_____")
PrintLine(1)
For i = 0 To TestDataSet1.Studenci.Rows.Count - 1
    PrintLine(1,
Microsoft.VisualBasic.Left(TestDataSet1.Studenci.Rows(i)("Imie"), 10),
TAB(), TestDataSet1.Studenci.Rows(i)("Nazwisko"))
    Next
FileClose(1)
End If
End If
End Sub

```

Efekt działania linii:

If SaveFileDialog1.ShowDialog() = DialogResult.OK Then



Wynik

lista studentów	
Imie	Nazwisko
Jan	Kowalski
Anna	Abacka
Ewa	Ols
Andrzej	Maciąg
Adam	Stawowy
Jan	Nowy

Drukowanie z pliku

- Istnieją problemy z formatowaniem

```

Dim WithEvents DoDruku As New PrintDocument()
Dim printFont As New System.Drawing.Font _
("Arial", 10, System.Drawing.FontStyle.Regular)
Dim StrumienToPrint As StreamReader

Private Sub ZeZbioru_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles ZeZbioru.Click
    OpenFileDialog1.InitialDirectory = "d:\"
    OpenFileDialog1.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*"
    OpenFileDialog1.FilterIndex = 2
    OpenFileDialog1.RestoreDirectory = True

```

```

If OpenFileDialog1.ShowDialog() = DialogResult.OK Then
    Try
        StrumienToPrint = New
StreamReader(OpenFileDialog1.FileName)
        Try
            AddHandler DoDruku.PrintPage, AddressOf
Me.DoDruku_PrintPage
            PrintPreviewDialog1.Document = DoDruku
            PrintPreviewDialog1.ShowDialog()
        Finally
            StrumienToPrint.Close()
        End Try
    Catch ex As Exception
        MessageBox.Show(ex.Message)
    End Try
End If
End Sub

```

```

Private Sub DoDruku_PrintPage(ByVal sender As Object, ByVal e As
System.Drawing.Printing.PrintPageEventArgs) Handles
DoDruku.PrintPage
    Dim linesPerPage As Single = 0
    Dim yPos As Single = 0
    Dim count As Integer = 0
    Dim leftMargin As Single = e.MarginBounds.Left
    Dim topMargin As Single = e.MarginBounds.Top
    Dim line As String = Nothing
    linesPerPage = e.MarginBounds.Height /
printFont.GetHeight(e.Graphics)

```

```

While count < linesPerPage
    line = StrmienToPrint.ReadLine()
    If line Is Nothing Then
        Exit While
    End If
    yPos = topMargin + count * printFont.GetHeight(e.Graphics)
    e.Graphics.DrawString(line, printFont, Brushes.Black,
leftMargin, yPos) ', New StringFormat()
    count += 1
End While
If Not (line Is Nothing) Then
    e.HasMorePages = True
Else
    e.HasMorePages = False
End If
End Sub

```

Wynik

