

ADO .NET

Maja Štula

Ak. God. 2011/2012

ADO .NET

- ADO.NET je dio .NET Frameworka koji omogućava spajanje aplikacije na različite izvore podataka i upravljanje podacima koji se nalaze u izvoru podataka.
- To je objektno-orijentirana skupina biblioteka sa gotovim klasama koje u sebi uključuju funkcionalnosti potrebne za rad sa izvorima podataka.

ADO

- Preteča ADO.NET-a je ADO (ActiveX Data Objects). ADO je skup COM objekata koji se koriste za pristup izvorima podataka. ADO.NET je prošireni (*extended*) ADO, te se naziva i ADO+.
- ADO se temelji na spojnom (*connected*) pristupu podacima, što znači da kada se uspostavi veza sa bazom podataka, ta veza ostaje otvorena sve dok se aplikacija ne zatvori.

ADO

- Spojni pristup podacima je relativno dobar za kreiranje Windows desktop aplikacija, ali se ne preporuča za Web aplikacije jer narušava sigurnost sustava, smanjuje performanse i može biti neupotrebljiv ako aplikaciju koristi veliki broj korisnika.

ADO .NET

- ADO.NET uvodi takozvani bespojni (*disconnected*) pristup podacima.
- Kod ovog pristupa podacima veza između aplikacije i baze podataka ostaje otvorena samo onoliko dugo koliko je potrebno da bi se dohvatili podaci iz baze ili izvršile određene promjene u bazi podataka (npr. ažuriranje tablica), i onda se automatski zatvara.

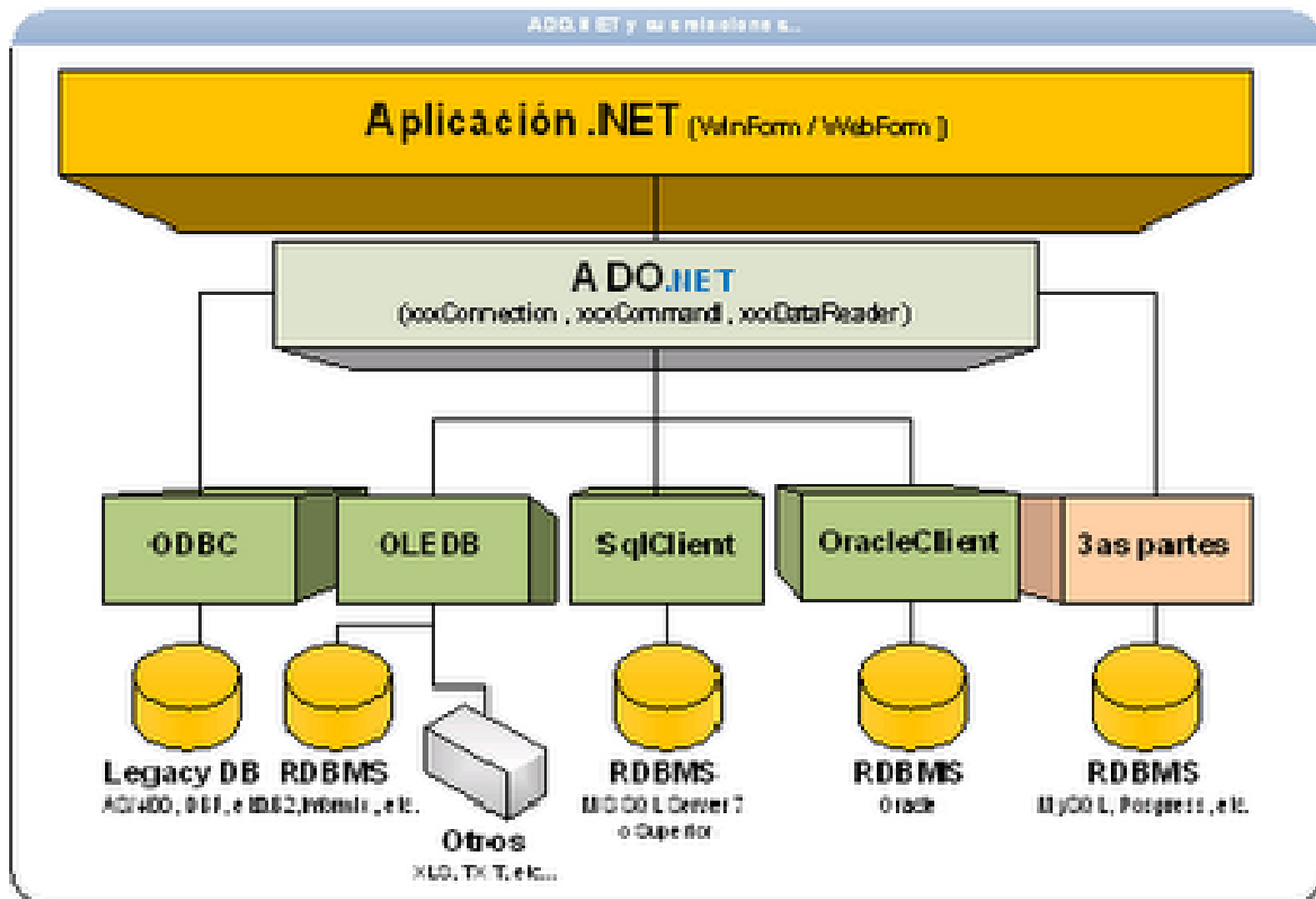
ADO .NET

- Izvor podataka u ADO.NET-u je najčešće relacijska baza podataka, ali može biti i tekstualna, Excel ili XML datoteka.
- Baze podataka (Microsoft SQL Server, Microsoft Access, Oracle, MySQL, IBM DB2, itd.) se razlikuju, između ostalog i po protokolu putem kojeg komuniciraju sa aplikacijom.

ADO. NET

- Sučelje između baze podataka i aplikacije koja pristupa bazi olakšava razvoj aplikacije tako što pruža API koji skriva implementacijske detalje baze od programera.
- ODBC (*Open Database Connectivity*) je API za pristup bazi podataka koji je razvio Microsoft.
- ADO.NET u sebi uključuje podršku preko niza API mehanizama za pristup bazama, pored ODBC-a, podržava, OleDb, Oracle,

ADO.NET



ADO. NET

- Arhitektura ADO.NET-a je složena, te uključuje veliki skup klasa.
- Prva stvar u aplikaciji je uspostavljanje konekcije sa bazom podataka. Nakon uspostavljene komunikacije mogu se dohvaćati podaci iz baze. Dvije osnovne skupine klasa za rad sa bazama podataka nazivaju se *Data Provider* i *DataSet*.

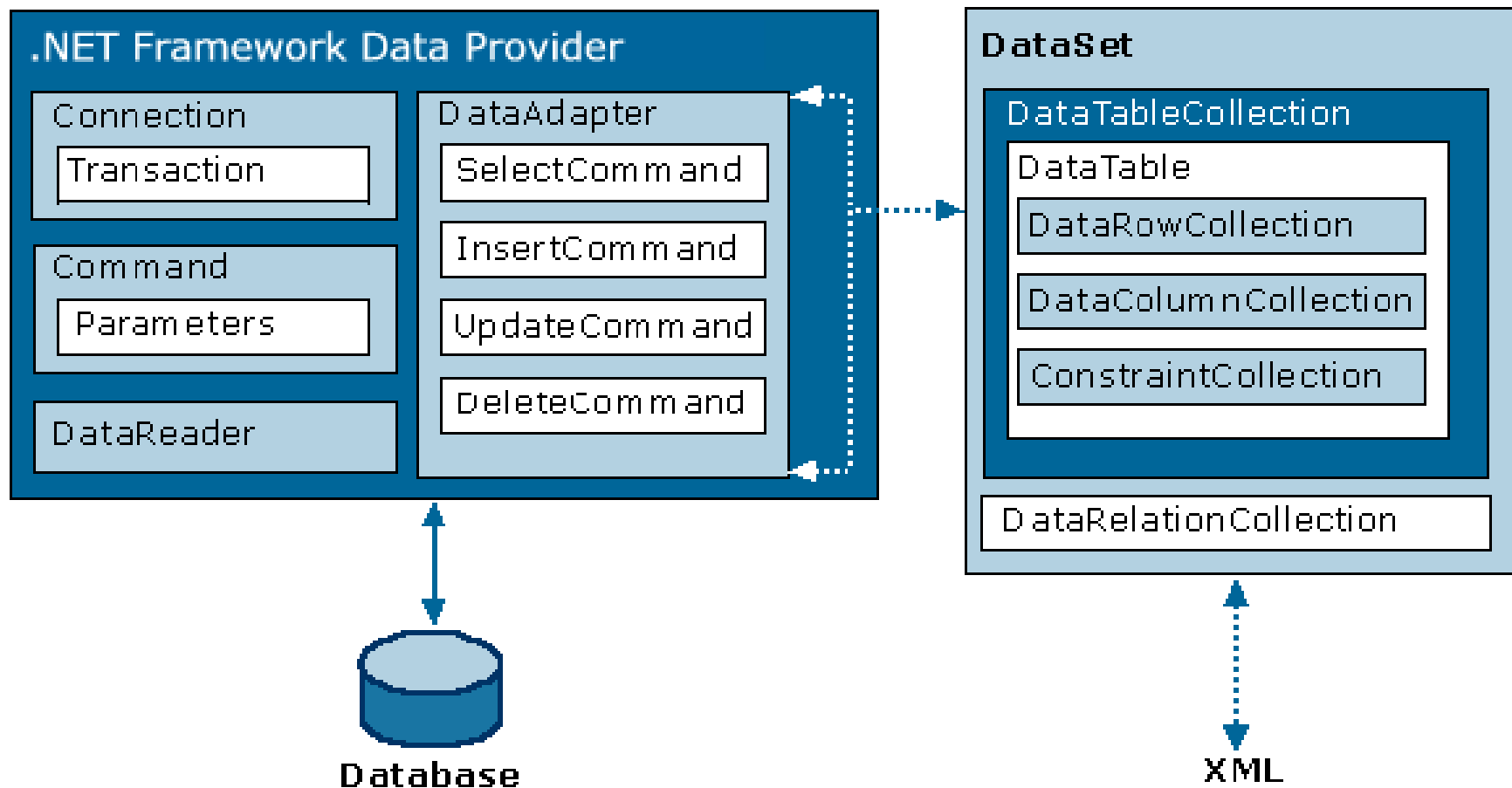
ADO .NET

- Instance *Data Provider* klasa omogućavaju komunikaciju sa bazom podataka. Niz *Data Provider* klasa je definirano s obzirom na izvor podataka tj. za različite baze podataka postoje različiti *Data Provideri*
(System.Data.SqlClient.SqlConnection – za sql konekcije,
System.Data.OleDb.OleDbConnection – za oledb konekcije,)

ADO .NET

- *DataSet* objekti su pojednostavljeno objekti koji predstavljaju kopiju podataka iz baze u memoriji i na koje se oslanja bespojni pristup.
- Omogućavaju prikaz podataka koji se nalaze u bazi podataka s kojom se komunicira preko *Data Provider* objekta.

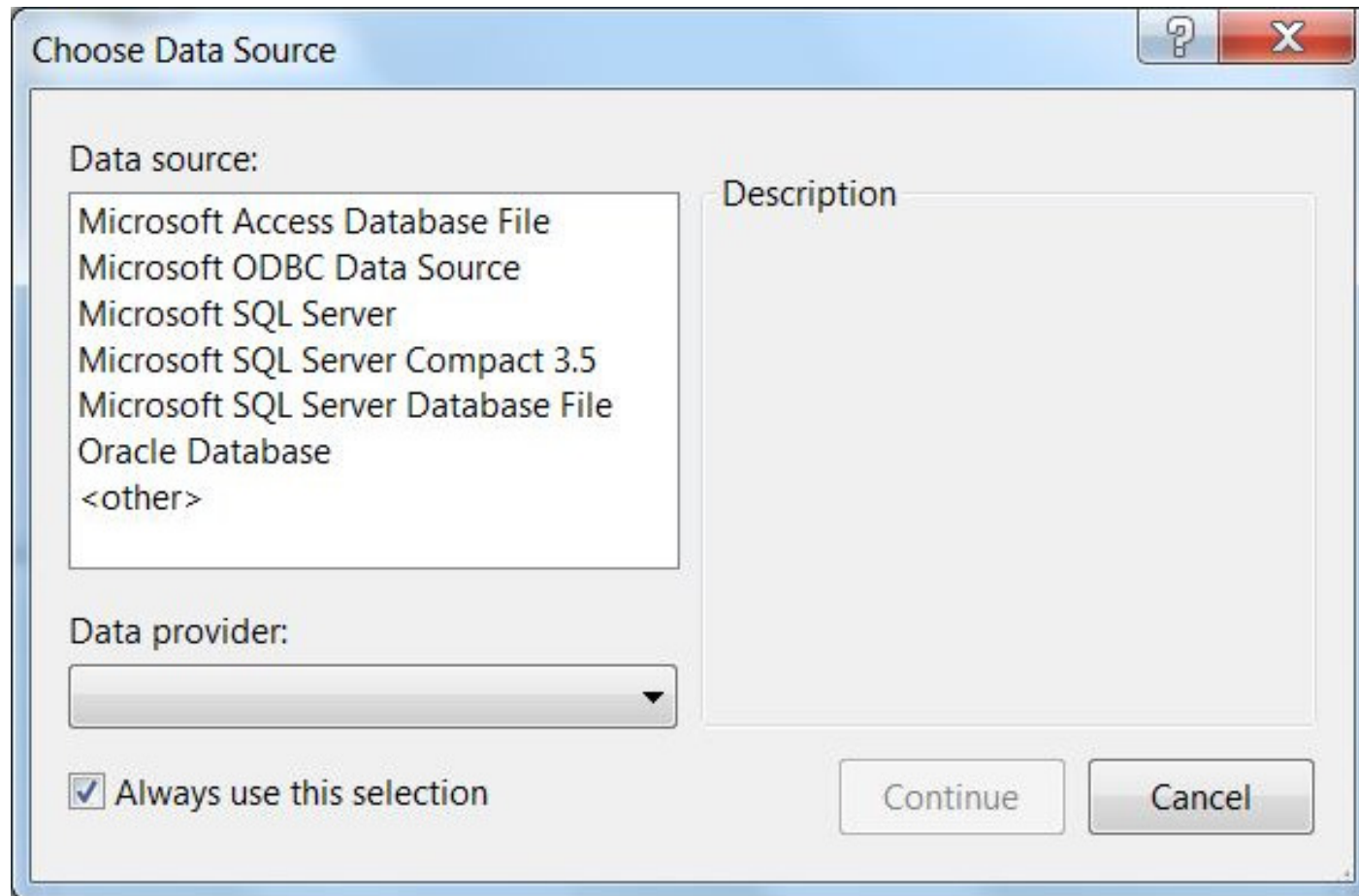
ADO .NET architektura



Microsoft Visual Studio

- Uključivanje gotove baze podataka u određeni projekt u Microsoft Visual Studio je jednostavno.
- Klikom na *Data* → *Add New Data Source* pojavi se *Data Source Configuration Wizard* koji nam omogućava odabir baze podataka koju želimo uključiti u naš projekt.

Microsoft Visual Studio



Izbor Data Providera u Visual Studiju

Data Provider

- Svaki *Data Provider* sastoji se od četiri glavna objekta: *Connection*, *Command*, *DataAdapter* i *DataReader*.
 1. *Connection*: Ostvaruje vezu između aplikacije i baze podataka. Ovaj objekt prima samo jedan argument tipa *string* koji se naziva *ConnectionString* i koji identificira ime baze podataka na koju se spajamo, ime servera, korisničko ime i lozinku.

ConnectionString

- Visual Studio sam generira *ConnectionString*. *ConnectionString* je argument tipa *string* kojega prima objekt *Connection* i koji identificira ime servera i baze podataka te prava pristupa.
- [Primjer](#)
- `connectionString="Data Source=RACUNALO\SQLEXPRESS;Initial Catalog=mala;Integrated Security=True"`

ConnectionString

Svojstvo ConnectionStringa	Funkcija
DataSource	Ime ili mrežna adresa instance SQL servera na kojeg se spajamo. .\SQLEXPRESS je <i>default</i> instanca SQL servera.
Initial Catalog -or- Database	Ime baze
AttachDbFilename	Predstavlja put do primarne baze podataka koja se koristi u aplikaciji.
Integrated Security (Windows Authentication)	Predstavlja prava pristupa bazi podataka. Ako je <i>Integrated Security</i> postavljen na <i>True</i> , to znači da se unutar <i>ConnectionStringa</i> ne mora upisivati korisničko ime i lozinka da bi se pristupilo bazi podataka. Umjesto toga, koristi se trenutni Windows identitet (korisničko ime i lozinka korisnika koji radi na tom računalu). Zato se ovo svojstvo <i>ConnectionStringa</i> još naziva i <i>Windows Authentication</i> .
User Instance	Omogućava korisnicima računala koji nisu administratori da se spoje na SQL Server bazu podataka bez potrebe za administratorskim pravima.
...	postoji još cijeli niz svojstava

Data Provider

2. *Command*: Predstavlja SQL izraz ili pohranjenu proceduru koja će se izvršiti nad podacima u bazi podataka. Korištenjem *Command* objekata podaci se najčešće čitaju, pišu ili ažuriraju. Da bi ovaj objekt znao kojoj bazi podataka mora poslati SQL upit, mora koristiti *Connection* objekt pomoću kojega se ostvaruje veza na bazu podataka. *Command* objekt koriste *DataAdapter* objekti za upravljanje vezom između baze podataka i *DataSeta*.

Data Provider

3. *DataReader*: Omogućava dohvaćanje rezultata SQL upita koji se nad bazom podataka izvršavaju korištenjem *Command* objekta. *DataReader* objekt podržava samo sekvencijalno dohvaćanje podataka iz baze podataka. Ovaj objekt koristi direktnu vezu sa bazom podataka, što znači da se podaci dohvaćaju jako brzo jer se ne spremaju u privremenu (*cache*) memoriju. Korištenje *DataReader* objekta nije uvijek dobro rješenje, te se onda koristi *DataSet* objekt umjesto njega.

Data Provider

4. *DataAdapter*: Postoje aplikacije koje koriste podatke koji se jako rijetko mijenjaju, pa je takve podatke najbolje spremi u međumemoriju (*cache*) kako bi se smanjio broj pristupa bazi podataka i time ubrzao rad same aplikacije. U ovome nam najbolje pomaže *DataAdapter* objekt koji čita podatke iz baze podataka i istovremeno te podatke prepisuje u *DataSet* objekt.

```

.....using System.Data.SqlClient;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            SqlConnection conn = new SqlConnection("Data Source=(local);Initial Catalog=prva;Integrated Security=True");
            SqlDataReader rdr = null;
            try {
                // 2. Open the connection
                conn.Open();
                // 3. Pass the connection to a command object
                SqlCommand cmd = new SqlCommand("select * from Table_1", conn);
                // 4. Use the connection
                // get query results
                rdr = cmd.ExecuteReader();
                while (rdr.Read())
                {
                    Console.WriteLine(rdr[0]); Console.WriteLine(rdr[1]); }
            }
            finally
            { // close the reader
                if (rdr != null) { rdr.Close(); }
                // 5. Close the connection
                if (conn != null) { conn.Close(); } }
        } } }

```

Microsoft SQL

- SqlConnection je objekt Connection za Microsoft SQL konekciju.
- Nakon instanciranja objekta, pozivom metode open() se otvara konekcija prema serveru.
- SqlCommand je objekt Command za Microsoft SQL konekciju. Ovim objektom definiramo akciju koja će se izvršiti na bazi (select, insert, ...).

Microsoft SQL

- Obično se instancira sa stringom koji sadrži sql upit i referencom na SqlConnection objekt:

```
SqlCommand cmd = new SqlCommand("select *  
from Table_1", conn);
```
- Poziv metode ExecuteReader vraća objekat tipa SqlDataReader. Ovaj objekt je pogodan za čitanje podataka iz baze (ne može se koristiti za pisanje).

Microsoft SQL

- Za ubacivanje podataka u bazu koristi se metoda `ExecuteNonQuery` na `SqlCommand` objektu.

```
string insertString = @"insert into Table_1  
    (id, ime) values (7, 'ivan')";  
SqlCommand cmd = new  
    SqlCommand(insertString, conn);  
// 2. Call ExecuteNonQuery to send command  
cmd.ExecuteNonQuery();
```


Microsoft SQL

- Za updateanje podataka također se koristi metoda ExecuteNonQuery.

```
string updateString = @"  
    update Table_1 set ime = 'mate'  
    where ime= 'ivan';  
SqlCommand cmd = new  
SqlCommand(updateString, conn);  
cmd.ExecuteNonQuery();
```

Microsoft SQL

- Za brisanje podataka također se koristi metoda ExecuteNonQuery.

```
string deleteString = @"  
delete from Table_1  
where ime = 'mate'";  
SqlCommand cmd = new SqlCommand();  
cmd.CommandText = deleteString;  
cmd.Connection = conn;  
cmd.ExecuteNonQuery();
```

Microsoft SQL

- Prenošnje parametara u sql upit pomoću tzv. placeholdera se radi na sljedeći način. Za definiranje parametra se koristi prefiks '@' :

```
SqlCommand cmd = new SqlCommand( "select *  
    from Table_1 where ime = @Ime", conn);
```

- *@Ime* je deklaracija parametra.

Microsoft SQL

- Svaki parametar treba prethodno biti definiran:

```
SqlParameter param = new SqlParameter();  
param.ParameterName = "@Ime";  
param.Value = textBox1.text;
```

- ParameterName mora biti identičan onom korištenom u SqlCommand objektu. Vrijednost parametra će biti zamijenjena sa Value sadržajem.

Microsoft SQL

- Također je potrebno sve parametre dodati u SqlCommand objekt:

```
cmd.Parameters.Add(param);
```

- Zato se koristi metoda Add. Svaki parametar koji se koristi u SqlCommand objektu se treba prethodno dodati.

```

private void button1_Click(object sender, EventArgs e)
{
    SqlConnection conn = new SqlConnection("Data Source=(local);Initial Catalog=prva;Integrated
Security=True");
    SqlDataReader rdr = null;
    try
    {
        // 2. Open the connection
        conn.Open();
        SqlCommand cmd = new SqlCommand("select * from Table_1 where ime = @Ime", conn);
        SqlParameter param = new SqlParameter();
        param.ParameterName = "@Ime";
        param.Value = textBox1.Text;
        cmd.Parameters.Add(param);
        rdr = cmd.ExecuteReader();
        while (rdr.Read())
        {
            richTextBox1.AppendText(rdr[0] + ", " + rdr[1] + "\n");
        }
    }
    finally
    {
        if (rdr != null) { rdr.Close(); }
        if (conn != null) { conn.Close(); }
    }
}

```

DataAdapter

- *DataAdapter* objekt sadrži referencu na *Connection* objekt i on automatski otvara i zatvara bazu podataka kada je to potrebno. Također, *DataAdapter* objekt sadrži reference i na SELECT, INSERT, UPDATE i DELETE operacije nad podacima.
- Svaka tablica koja se nalazi u *DataSetu* ima definiran svoj *DataAdapter* koji se brine za komunikaciju sa bazom podataka.
- Mogli bismo reći da je *DataAdapter* most između *DataSeta* i baze podataka.

DataSet

- *DataSet* je skup međusobno povezanih tablica, odnosno neka vrsta skladišta podataka.
- *DataSet* je jedna od najvažnijih komponenti ADO.NET-a, jer se na njoj zasniva bespojni pristup izvorima podataka.
- Kada se želi pristupiti određenoj tablici unutar neke baze podataka, *DataAdapter* tu tablicu prepíše u *DataSet*.

Punjenje DataSeta preko TableAdaptora

```
DataSet dsmoj = new DataSet();
```

```
SqlDataAdapter damoj = new  
    SqlDataAdapter(  
        "select * from Table_1", conn);
```

```
damoj.Fill(dsmoj, " Table_1");
```

Punjenje DataSeta preko TableAdapttera

- Objekt *conn* (tipa Connection), treba biti instanciran, ali bez poziva metode `open()` jer `SqlDataAdapter` objekt sam otvara i zatvara konekciju u pozivima metoda `Fill` i `Update`. Poziv metode `Fill` na `SqlDataAdapter` objektu sadrži objekt `DataSet` i ime tablice “u memoriji” u koju će se spremiti podaci definirani sql upitom iz `SqlDataAdapter`.

DataSet

- Nakon punjenja *DataSeta* se više ne radi direktno nad podacima unutar baze podataka, već nad njihovim kopijama koji se nalaze u *DataSetu*.
- Originalnoj bazi podataka se pristupa jedino u slučaju kada se žele sačuvati promjene izvršene nad podacima u *DataSetu* ili kada se želi popuniti *DataSet* sa podacima iz baze podataka.

DataSet

- *DataSet* se sastoji od *DataTable* i *DataRelation* objekata.
- *DataTable* je tablica podataka koja se nalazi u *DataSetu*. Može se proizvoljno definirati ili pak može nastati kao rezultat nekog upita nad bazom podataka.

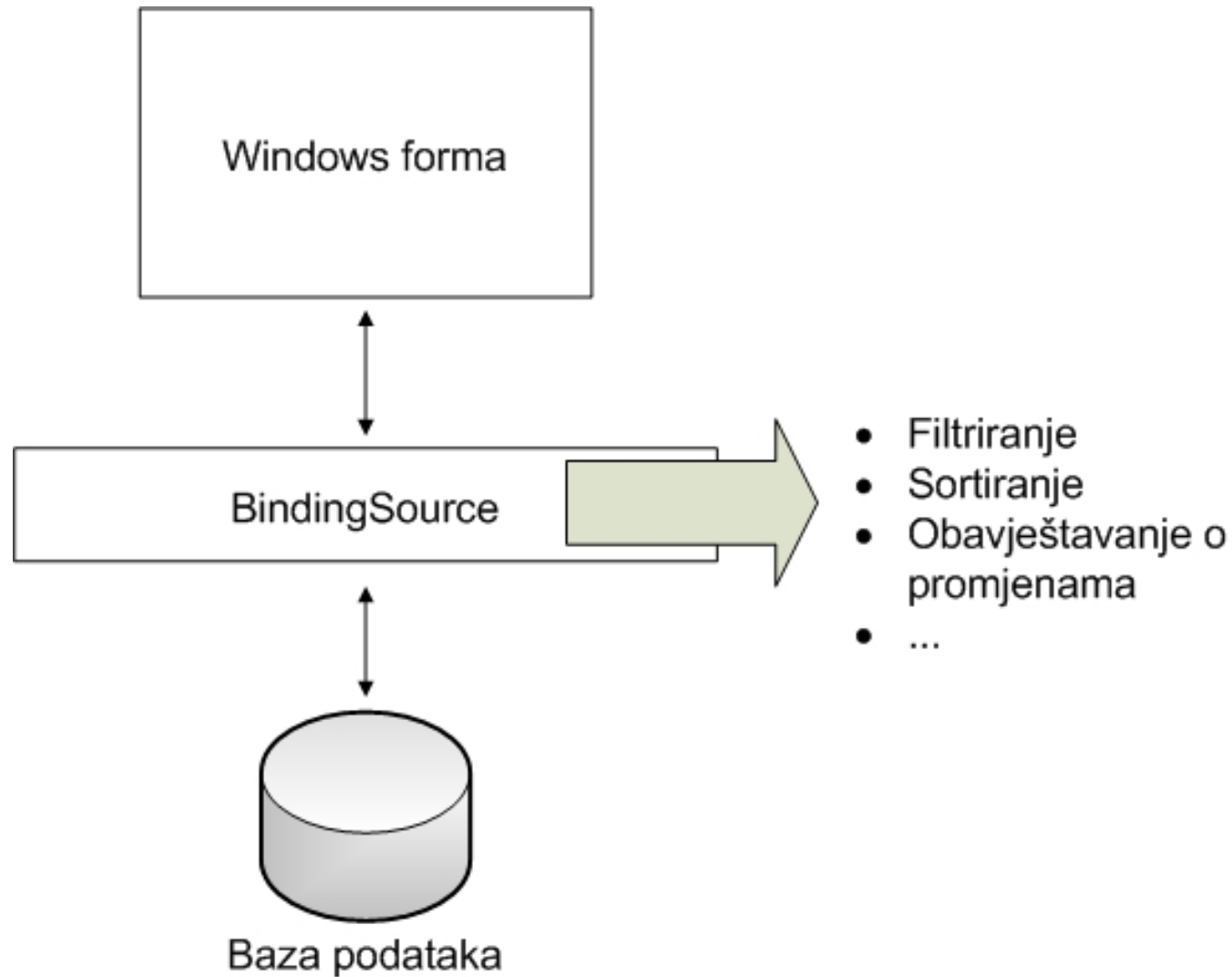
DataSet

- *DataTable* se sastoji od:
 - *DataRow* – Predstavlja jedan redak unutar *DataTable* objekta,
 - *DataColumn* – Predstavlja jedan stupac unutar *DataTable* objekta,
 - *Constraint* – Predstavlja ograničenja nad određenim *DataTable* objektom.
- *DataRelation* predstavlja *roditelj-dijete* (*parent-child*) relaciju između dva *DataTable* objekta.

BindingSource

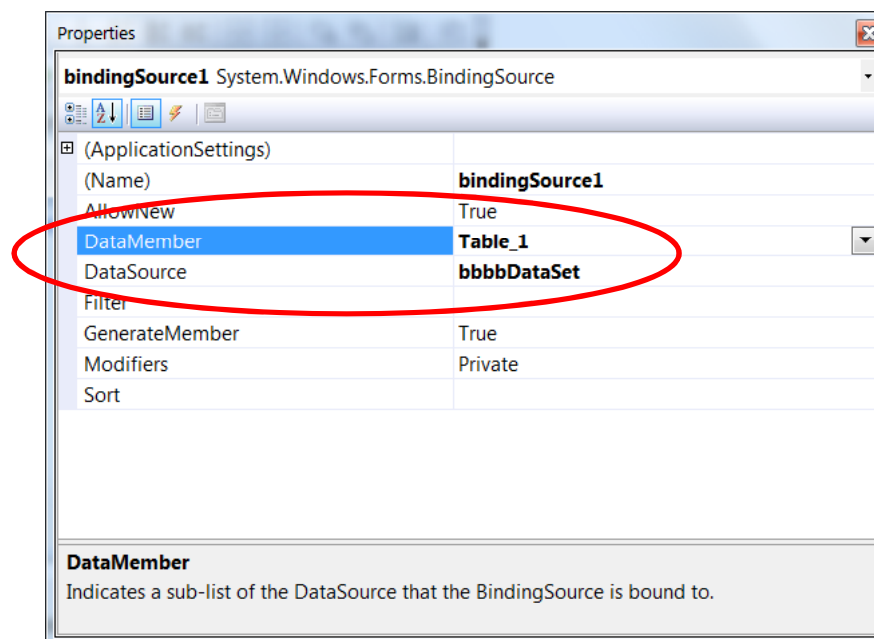
- *BindingSource* je gotova kontrola koja pruža mehanizam povezivanja elemenata grafičkog sučelja i izvora podataka preko prethodno spomenutih objekata.
- Olakšava developeru aplikacije povezivanje GUI sučelja sa izvorom podataka.

BindingSource



BindingSource

- *BindingSource* je vezan na izvor podataka pomoću *DataSource* i *DataMember* svojstva. *DataSource* može biti različiti izvor podataka, ali se najčešće koristi *DataSet* objekt.



BindingSource

- Postupak korištenje *BindingSource* je sljedeći:
 1. Na formu je potrebno dodati novi *BindingSource*.
 2. *DataSource* za *BindingSource* je *DataSet* koji se kreira prilikom povezivanja na bazu.
 3. Na *BindingSource* je potrebno još povezati tablicu iz *DataSource*ea. Time se automatski kreira *TableAdapter* na *DataSetu*.
 4. Povezati željenu kontrolu sa *BindingSourceom* preko *DataBindings* svojstva.

Povezivanje kontrola

Properties

textBox1 System.Windows.Forms.TextBox

(ApplicationSettings)

(DataBindings)

(Advanced)

Tag	bindingSource1 - id_prvi
Text	bindingSource1 - id_prvi
(Name)	textBox1
AcceptsReturn	False
AcceptsTab	False
AccessibleDescription	
AccessibleName	
AccessibleRole	Default
AllowDrop	False
Anchor	Top, Left
AutoCompleteCustomSource	(Collection)
AutoCompleteMode	None
AutoCompleteSource	None
BackColor	Window
BorderStyle	Fixed3D
CausesValidation	True
CharacterCasing	Normal
ContextMenuStrip	(none)
Cursor	IBeam
Dock	None
Enabled	True
Font	Microsoft Sans Serif; 8,25pt
ForeColor	WindowText
GenerateMember	True
HideSelection	True
ImeMode	NoControl
Lines	String[] Array
Location	45; 75
Locked	False
Margin	3; 3; 3; 3
MaximumSize	0; 0
MaxLength	32767
MinimumSize	0; 0
Modifiers	Private
Multiline	False
PasswordChar	
ReadOnly	False

(DataBindings)

The data bindings for the control.

Form2

0 of {0}

button1

bindingNavigator1 bindingSource1 malaDataSet table_1TableAdapter

BindingNavigator

- *BindingNavigator* je kontrola koja se dodaje određenoj formi i poveže sa određenim *BindingSource*om, a sve sa ciljem da se stvori relacija između forme i baze podataka koja omogućava navigaciju po podacima.

```
private void  
    bindingNavigatorMoveFirstItem_Click(object sender, EventArgs e)  
{  
    try  
    {  
        MessageBox.Show("Pomak na prvi");  
        this.bindingSource1.Position = 0;  
    }  
    catch (ConstraintException ee)  
    {  
        MessageBox.Show(ee.ToString());  
    }  
}
```

BindingNavigator

The screenshot displays a Windows Forms application in design mode. The main window, titled 'Form2', contains a 'BindingNavigator' control at the top. This control is highlighted with a red oval and contains navigation buttons (first, previous, next, last, add, delete) and a text box showing '0 of {0}'. Below the form, the 'Toolbox' shows several components: 'bindingSource1', 'bbbbDataSet', 'bindingNavigator1', and 'table_1TableAdapter'. A green oval highlights 'bindingSource1' in the toolbox, with a yellow arrow pointing from it to the 'BindingSource' property in the 'Properties' window. The 'Properties' window on the right shows the configuration for 'bindingNavigator1' (System.Windows.Forms.BindingNavigator). The 'DataBindings' section is expanded, showing the 'BindingSource' property set to 'bindingSource1'. Other properties like 'AddNewItem', 'DeleteItem', and 'CountItem' are also configured with default values.

Program.cs Form2.cs [Design]* app.config Form1.cs [Design]

Form2

0 of {0}

bindingSource1 bbbbDataSet bindingNavigator1 table_1TableAdapter

Properties

bindingNavigator1 System.Windows.Forms.BindingNavigator

(DataBindings)	
(Name)	bindingNavigator1
AccessibleDescription	
AccessibleName	
AccessibleRole	Default
AddNewItem	bindingNavigatorAddNewItem
AllowDrop	False
AllowItemReorder	False
AllowMerge	True
Anchor	Top, Left
AutoSize	True
BackColor	<input type="checkbox"/> Control
BackgroundImage	<input type="checkbox"/> (none)
BackgroundImageLayout	Tile
BindingSource	bindingSource1
CanOverflow	True
ContextMenuStrip	(none)
CountItem	bindingNavigatorCountItem
CountItemFormat	of {0}
DeleteItem	bindingNavigatorDeleteItem
Dock	Top
Enabled	True
Font	Segoe UI; 7,8pt
GenerateMember	True

BindingSource
The BindingSource that the BindingNavigator navigates.

Povezivanje kontrola

Properties

dataGridView1 System.Windows.Forms.DataGridView

(ApplicationSettings)

(DataBindings)

(Name)	dataGridView1
AccessibleDescription	
AccessibleName	
AccessibleRole	Default
AllowDrop	False
AllowUserToAddRows	True
AllowUserToDeleteRows	True
AllowUserToOrderColumns	False
AllowUserToResizeColumns	True
AllowUserToResizeRows	True
AlternatingRowsDefaultCellStyle	DataGridViewCellStyle { }
Anchor	Top, Left
AutoSizeMode	None
AutoSizeMode	None
BackColor	AppWorkspace
BorderStyle	FixedSingle
CausesValidation	True
CellBorderStyle	Single
ClipboardCopyMode	EnableWithAutoHeaderText
ColumnHeadersBorderStyle	Raised
ColumnHeadersDefaultCellStyle	DataGridViewCellStyle { BackColor=Color [Control],
ColumnHeadersHeight	21
ColumnHeadersHeightSizeMode	AutoSize
ColumnHeadersVisible	True
Columns	(Collection)
ContextMenuStrip	(none)
Cursor	Default
DataMember	
DataSource	bindingSource1
DefaultCellStyle	DataGridViewCellStyle { BackColor=Color [Window],
Dock	None
EditMode	EditOnKeystrokeOrF2
Enabled	True
EnableHeadersVisualStyles	True
GenerateMember	True

[Edit Columns...](#); [Add Column...](#)

(ApplicationSettings)
Maps property settings to an application configuration file.

Form1

	id_prvi	prvi_tekst
*		

	id_drugi	tekst_drugi
*		

bindingSource1 malaDataSet table_1TableAdapter bindingSource2

TableAdapter

- *TableAdapteri* omogućavaju komunikaciju između baze podataka i aplikacije.
- Oni se spoje na bazu podataka, izvrše određeni SQL upit i vrate ili novu tablicu (*DataTable*) ili pak popune staru sa novim podacima.
- *TableAdapteri* se također koriste kod ažuriranja baze podataka.

Spremanje podataka u bazu

- Promjene na *DataSetu* potrebno je na kraju rada sa aplikacijom spremiti u bazu podataka.
- To se radi pozivom metode `Update` na `DataAdapterima`.
- `DataAdapter` ne kreira automatski SQL naredbe za pohranu promjena na bazi podataka pa je to potrebno napraviti ili korištenjem `CommandBuilder` klase ili ručno.

Spremanje podataka u bazu

- Npr. iz DataSeta se dohvati ubačeni podataka i preko TableAdapter u bazu, ali prije treba napraviti SQL upit.

```
DataRow pomocni = majaDataSet.druga.Rows[i];  
this.drugaTableAdapter.InsertQuery((byte)  
    pomocni[0], (string) pomocni[1]);
```