

Свързване на данни (Data Binding)

Лекция 9

Съдържание

- ▶ Свързване на данни (Data Binding)
- ▶ Просто свързване
- ▶ Сложно свързване
- ▶ Валидация на контроли

Свързване на данни

▶ **Свързването на данните (Data Binding)**

- ▶ Осигурява **автоматично прехвърляне на данни** между **контроли** и **източници на данни**
- ▶ Например: свързване на масив, съдържащ имена на студенти с **ComboBox** контрола

▶ **Източници на данни**

- ▶ **ICollection** – масиви и колекции
- ▶ **IBindingList** – поддържа се от **DataView**
 - ▶ поддържа нотификация за промяна на даните

▶ **Контроли, поддържащи data binding**

- ▶ **Всички Windows Forms контроли**
 - ▶ **TextBox, ComboBox, ListBox, DataGridView**

Свързване на данни

- ▶ При добавяне на **свързване** указваме:
 - 1) **свойството** на контролата, което ще **свързваме с данните**
 - 2) **източникът** на данните
 - 3) **път до списък** или **свойство на източника**, към което ще се свържем
- ▶ Този път може да е:
 - ▶ **име** на свойство
 - ▶ **йерархия от имена**, разделени с точки
 - ▶ **празен низ**
- ▶ Ако пътят е празен низ ще се извика метода **ToString()** на обекта, използван като източник на данни

Свързването е еднопосочно

- ▶ Промяна на свързано свойство от дадена контрола **променя данните в източника**, към който то е свързано
- ▶ **Обратното не е вярно!**
 - ▶ При промяна на източника на данни свързаните към него контроли не си променят свойствата
- ▶ След промяна на данните в източника на данни за отразяване на промените в свързаните с него контроли се прави следното:
 - 1) Премахване (изтриване) на свързването
 - 2) Добавяне на свързването отново

Видове свързване

- ▶ **Просто свързване (simple binding)**
 - ▶ Свързване на контрола с **единичен обект**
 - ▶ Свързване на контрола с **единичен (текущ) елемент от списък**

Например: `TextBox` и `CheckBox`

- ▶ **Сложно свързване (complex binding)**
 - ▶ Свързване на списъчна контрола **със списък**

Например: `ListBox`, `ComboBox`, `DataGrid`

- ▶ Поддържа се **текущо избран елемент** (активен) от списъка

Просто свързване – Пример 1. SimpleBindingText

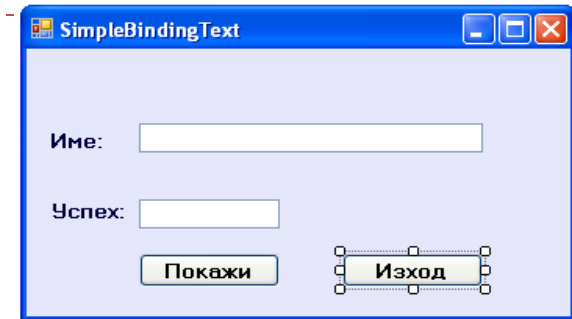
Свързване на контрола към обект

1/4

► При промяна на `TextBox1.Text` и `TextBox2.Text` се променя свързаният обект

1) Създаваме ГПИ с два етикета, две текстови полета и два бутона.

2) Създаваме клас **Student** с две свойства низ за **име (string Name)** и реално число за **успех (double value)**



```
class Student
{
    private string name;
    private double gpa;
    public string Name
    {
        get { return name; }
        set { name = value; }
    }
    public double Gpa
    {
        get { return gpa; }
        set { gpa = value; }
    }
}
```

доц. д-р Т. Терзиева

Свързване на контрола към обект – пример 1

2/4

3) Създаваме инстанция на класа Student

```
Student student = new Student();
```

3) Можем да **свържем свойствата Name и Gpa** със свойствата **Text** на двете текстови кутии:

```
private void FormBind_Load(object sender, EventArgs e)
{
    // create the student
    // Student student = new Student();
    student.Name = "Димитър Георгиев";
    student.Gpa = 6.0;
    // bind the properties
    textBox1.DataBindings.Add(new Binding("Text", student, "Name"));
    textBox2.DataBindings.Add(new Binding("Text", student, "Gpa"));
}
```


Свързване на свойства

```
// bind the properties
```

```
textBox1.DataBindings.Add(new Binding("Text", student, "Name"));
```

```
textBox2.DataBindings.Add(new Binding("Text", student, "Gpa"));
```

- ▶ Използваме колекцията **DataBindings** на класа **Control**.
- ▶ В нея можем да добавяме **Binding** обекти, които указват **кое свойство на текущата контрола** с кое **свойство на дадена друга контрола** да бъде свързано.
- ▶ В нашият случай:
 - ▶ при промяна на **textBox1.Text** ще се променя и свойството **Name** на свързания обект **student**
 - ▶ при промяна на **textBox2.Text** ще се променя и свойството **Gpa** на свързания обект **student**

Свързване на контрола към обект – пример

3/4

- 3) Ще добавим програмен код на събитието **Click** на бутон “Покажи” за показване на текущата стойност на името и успеха на студент
- 4) Това ни позволява да видим промените, които са направени върху обекта след промяна на текста на **TextBox**.

```
private void buttonShow_Click(object sender, EventArgs e)
{
    MessageBox.Show("Име: " + student.Name +
                    " Успех: " + student.Gpa);
}

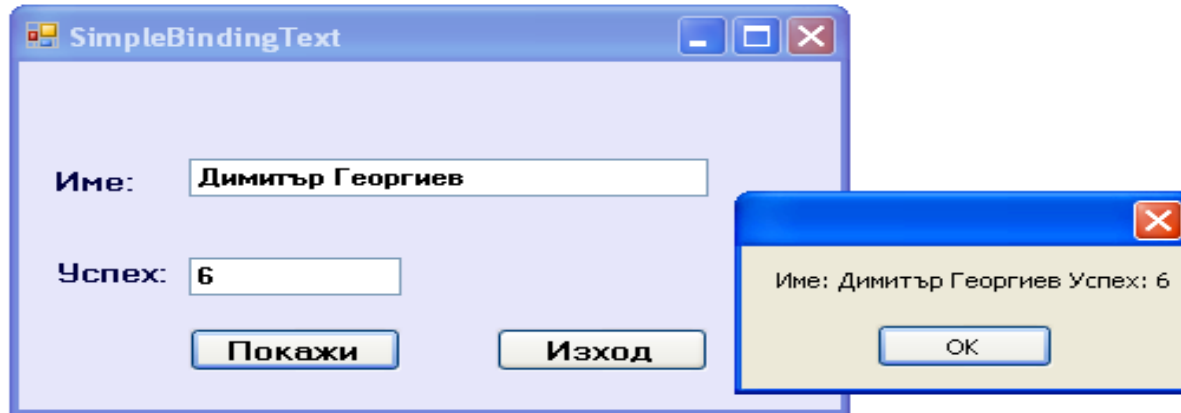
private void buttonExit_Click(object sender, EventArgs e)
{
    FormBind.ActiveForm.Close();
}
```



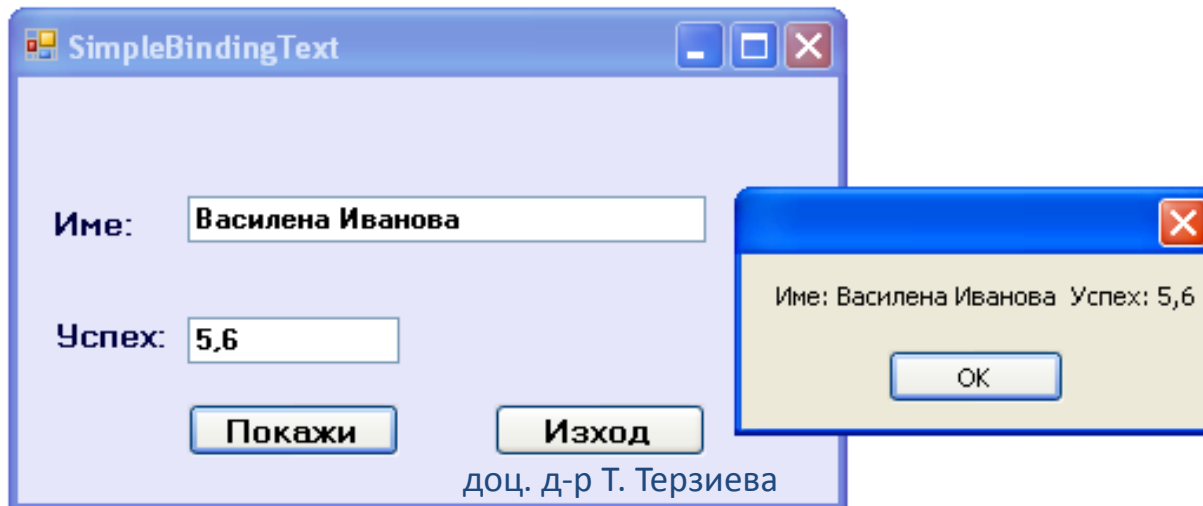
Свързване на контрола към обект – пример

4/4

5) След стартиране получаваме:



6) Промяната на стойностите на текстовите полета променя стойностите в съобщението:

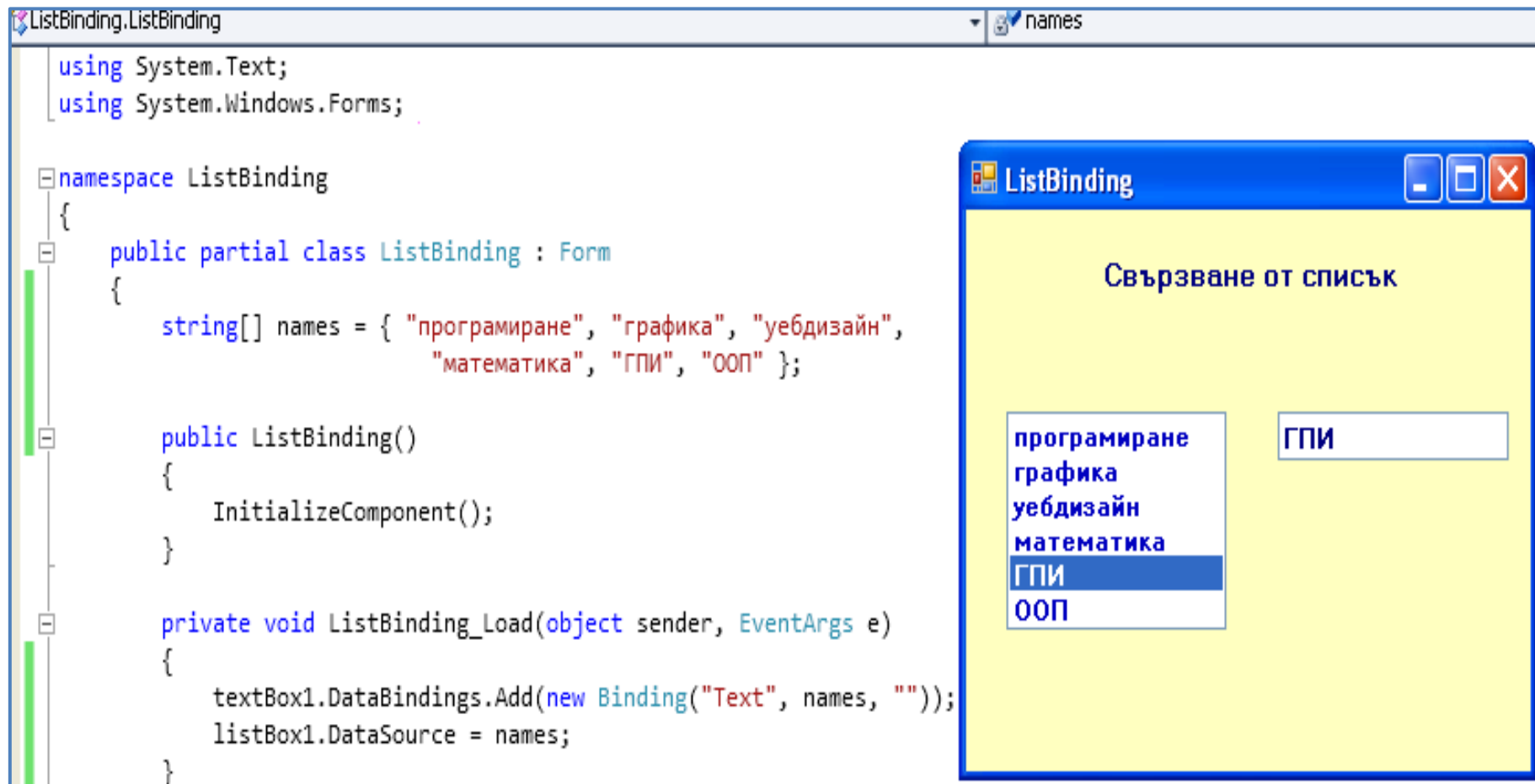


Свързване на контрола към списък

Сложно свързване

- ▶ Може да се реализира **свързване** на контрола **ListBox** с **масив** или **колекция от данни**
- ▶ Свързването с **колекция от данни** се нарича **сложно свързване на данни**
- ▶ За да свържем **ListBox** към колекция използваме свойството **DataSource** на **ListBox**
- ▶ **ListBox** ще покаже **първото достъпно** свойство на всяка единица от списъка

Свързване на масив от низове с ListBox



```
using System.Text;
using System.Windows.Forms;

namespace ListBinding
{
    public partial class ListBinding : Form
    {
        string[] names = { "програмиране", "графика", "уебдизайн",
                           "математика", "ГПИ", "ООП" };

        public ListBinding()
        {
            InitializeComponent();
        }

        private void ListBinding_Load(object sender, EventArgs e)
        {
            textBox1.DataBindings.Add(new Binding("Text", names, ""));
            listBox1.DataSource = names;
        }
    }
}
```

ListBinding

Свързване от списък

програмиране
графика
уебдизайн
математика
ГПИ
ООП

ГПИ

Сложно свързване

► Свързване на контрола към списък

```
string[] names = { "програмиране", "графика", "уебдизайн",  
                  "математика", "ГПИ", "ООП" };  
  
textBox1.DataBindings.Add(new Binding("Text", names, ""));  
listBox1.DataSource = names;
```

- **Свойството на източника**, към което ще се свържем е **празен низ**, защото искаме да свържем свойството **Text** директно с **елементите на масива**, който използваме като източник на данни, а не с тяхно свойство.
- При това **свързване** текстовата контрола ще се свърже първоначално с **първия елемент от масива** (символният низ **“програмиране”**), но след това програмно може да се укаже промяна на текущия елемент и свързването да се промени към някой друг от елементите на масива.

Свързване на **ListBox** с колекция

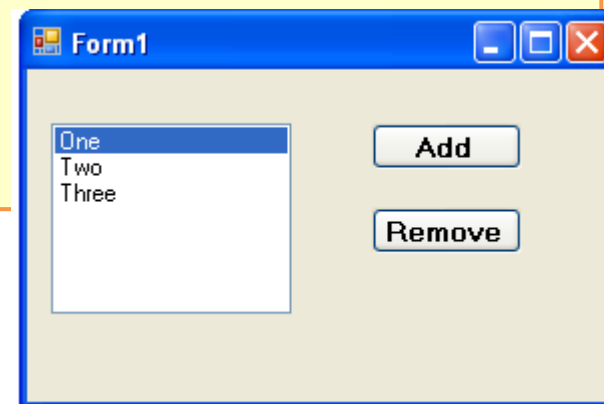
Създаваме форма **new Windows Forms C#**, добавяме контрола **ListBox** и два бутона **button Add** и **button Remove**.

Ще създадем нов списък като колекция от тип **string**.

```
List<String> itemslist = new List<string>();
```

Ще свържем тази колекция с **ListBox**.

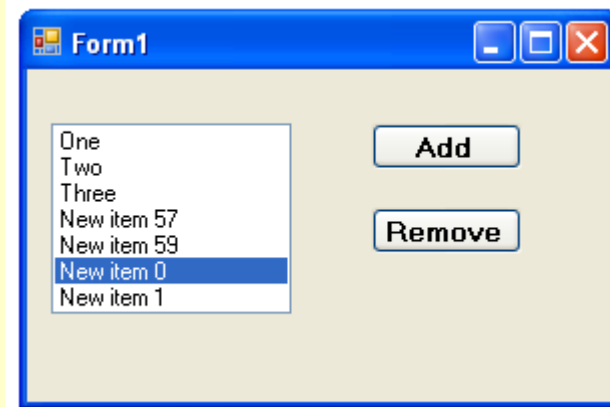
```
public Form1()  
{  
    InitializeComponent();  
    itemslist.Add("One");  
    itemslist.Add("Two");  
    itemslist.Add("Three");  
    listBox1.DataSource = itemslist;  
}
```



Свързване на ListBox с колекция

```
private void buttonAdd_Click(object sender, EventArgs e)
{
    // The Add button was clicked.
    itemslist.Add("New item " + DateTime.Now.Second); // <-- Any string you want
    // Change the DataSource.
    listBox1.DataSource = null;
    listBox1.DataSource = itemslist;
}

private void buttonRemove_Click(object sender, EventArgs e)
{
    // The Remove button was clicked.
    int selectedIndex = listBox1.SelectedIndex;
    try
    {
        // Remove the item in the List.
        itemslist.RemoveAt(selectedIndex);
    }
    catch (Exception b)
    {
        MessageBox.Show("Няма записи", "Грешка!", MessageBoxButtons.OK,
            MessageBoxIcon.Error);
    }
    listBox1.DataSource = null;
    listBox1.DataSource = itemslist;
}
```



Свързване на контрола към таблица

- ▶ Нека имаме **DataSet** обект **ds** с таблица **Towns** с колони **id** и **name** и **TextBox** контрола с име **TextBoxTowns**.
- ▶ Свързването на свойството **Text** на **TextBox** контролата към колоната **name** от таблицата може да се извърши по следния начин:

// Имаме DataSet ds с таблица Towns с колони id и name:

```
DataTable towns = new DataTable("Towns");
towns.Columns.Add(new DataColumn("id", typeof(int)));
towns.Columns.Add(new DataColumn("name", typeof(string)));
DataSet ds = new DataSet();
ds.Tables.Add(towns);
// Add three rows to the table
DataRow row;
row = towns.NewRow();
row["id"] = 1;
row["name"] = "София";
towns.Rows.Add(row);
```

Свързване на контрола към таблица

```
row = towns.NewRow();
row["id"] = 2;
row["name"] = "Пловдив";
towns.Rows.Add(row);

row = towns.NewRow();
row["id"] = 3;
row["name"] = "Варна";
towns.Rows.Add(row);

// Create a DataSet and add the table to the DataSet
DataSet ds = new DataSet();
ds.Tables.Add(towns);

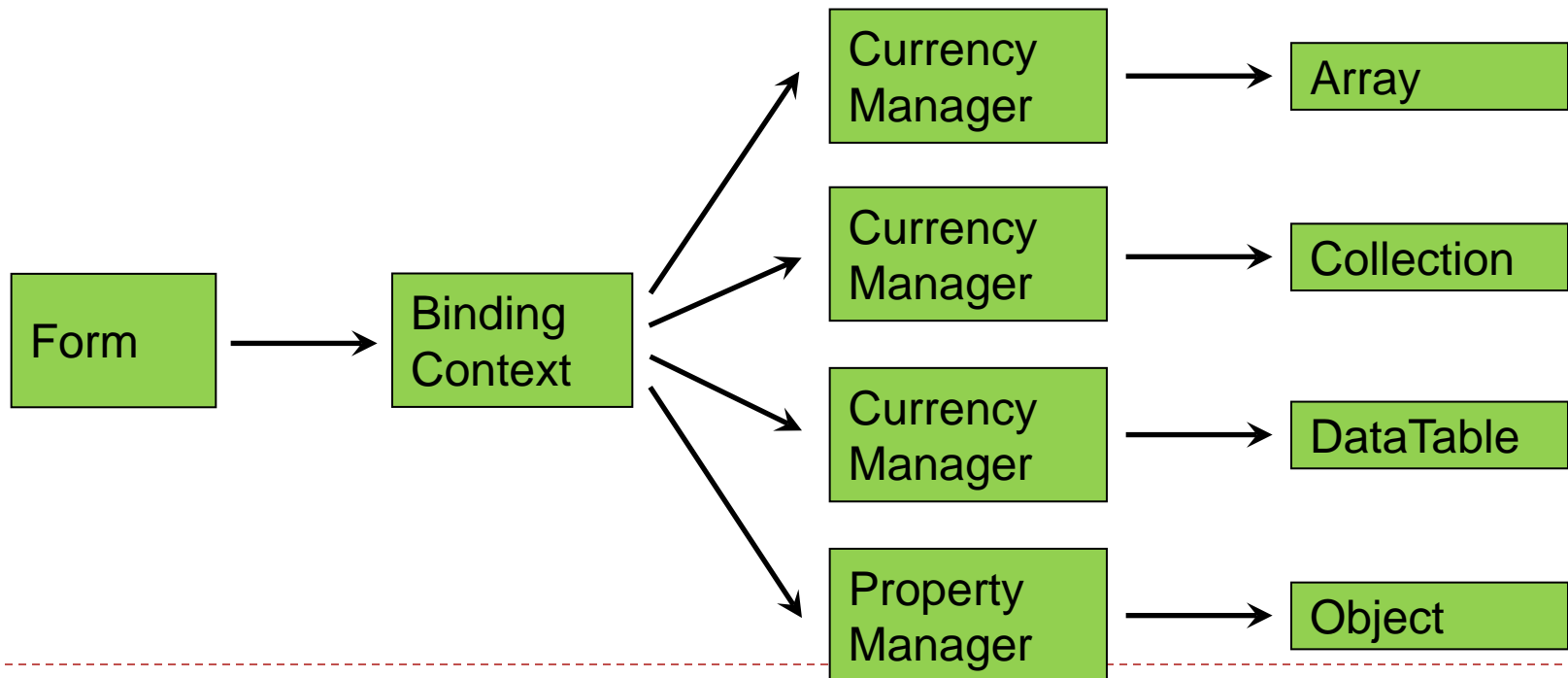
TextBoxTowns.DataBindings.Add( new Binding("Text", ds, "Towns.name"));
```

За да укажем, че искаме да свържем свойството **Text** на **TextBox** контролата с колоната **name** на таблицата **Towns** от източника на данни **ds**, задаваме **"Towns.name"** за път до свойството на източника.

Текстовото поле ще бъде свързано първоначално с първия ред на таблицата т.е. с полето **name** от този ред, но по-късно текущият ред може да бъде променен програмно.

BindingContext

- ▶ Формата пази информация за свързаните контроли в своя **BindingContext** обект
 - ▶ **CurrencyManager** – за контроли **свързани** към **списък**, **масив**, **колекция** или **таблица**. Съдържа **позицията** в списъка
 - ▶ **PropertyManager** – за контроли **свързани** към **обект**.
Ако източникът на данни е **обект**, който **връща само една стойност**



Навигация с CurrencyManager

- ▶ **CurrencyManager** класът пази **текущата позиция в списъка** – източник на данни
 - ▶ **Position** – съдържа позицията
 - ▶ **Count** – съдържа размера на списъка
- ▶ Навигация по източника на данни
 - ▶ Извличане на **CurrencyManager** обекта:

```
CurrencyManager cm = (CurrencyManager)
    textBox1.DataBindings["Text"].BindingManagerBase;
```

// или:

```
CurrencyManager cm = (CurrencyManager)
    form1.BindingContext[dataTableCustomers];
```

- ▶ Навигация по списъка:

```
cm.Position++;
```

Свързване на контрола към списък и навигация – пример **SimpleBindCurrency**

- ▶ **Свързване** на контрола към списък и навигация по списъка чрез **CurrencyManager**
- ▶ Поставяме върху главната форма една **TextBox** контрола с име **TextBoxTowns**, която ще свържем с масив от символни низове - имена на градове и два бутона с имена **ButtonPrev** и **ButtonNext**. Тези бутони ще служат съответно за навигация напред и назад по списъка с градовете. На свойството **Text** на двата бутона задаваме съответно стойности "**<< Prev**" и "**Next >>**".
- ▶ Добавяме код, който при зареждане на формата (при събитие **Load** на формата) свързва текстовото поле с масив съдържащ имена на градове:

```
private void MainForm_Load(object sender, EventArgs e)
{
    string[] towns = {"София", "Пловдив", "Варна", "Русе", "Бургас"};
    textBoxTowns.DataBindings.Add( new Binding("Text", towns, ""));
}
```

Свързване на контрола към списък и навигация – пример **SimpleBindCurrency**

- ▶ Добавяме обработчик на събитието **Click** на бутона **ButtonPrev**. В него извличаме от **CurrencyManager** обекта на текстовата контрола, текущата позиция в списъка с градовете и я намаляваме
- ▶ ако сме достигнали началото, позиционираме в края

```
private void buttPrev_Click(object sender, EventArgs e)
{
    CurrencyManager cm = (CurrencyManager)
    textBoxTowns.DataBindings["Text"].BindingManagerBase;
    if (cm.Position > 0)
    {
        cm.Position--;
    }
    else
    {
        cm.Position = cm.Count - 1;
    }
}
```

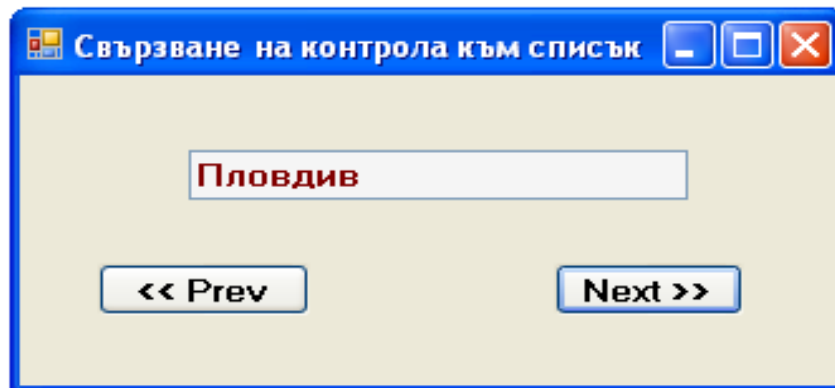
Свързване на контрола към списък и навигация – пример **SimpleBindCurrency**

- ▶ Добавяме обработчик на събитието **Click** на бутона **ButtonNext**. В него извличаме от **CurrencyManager** на текстовата контрола, текущата позиция в списъка с градовете и я увеличаваме
- ▶ ако сме достигнали края, позиционираме в началото:

```
private void buttNext_Click(object sender, EventArgs e)
{
    CurrencyManager cm = (CurrencyManager)
    textBoxTowns.DataBindings["Text"].BindingManagerBase;
    if (cm.Position < cm.Count - 1)
    {
        cm.Position++;
    }
    else
    {
        cm.Position = 0;
    }
}
```

Просто свързване и навигация

- ▶ Стартираме и тестваме приложението:



- ▶ Ако променим името на някой град, промяната се отразява в масива с имената.

Сложно свързване

- ▶ Свързване на контрола към списък, като контролата се свързва с повече от един елемент от списъка.
 - ▶ Използва се при списъчни контроли:
 - ▶ **ListBox**, **ComboBox** и др.
 - ▶ Задават се свойствата:
 - ▶ **DataSource** – списък с данните
 - ▶ **DisplayMember** – път до полето, което да се визуализира
 - ▶ **ValueMember** – път до полето, от което се получава резултата

Стойността по подразбиране в **DisplayMember** и **ValueMember** е празен низ

- ▶ Пример:

```
Dataset dataSetCountries = ...;  
comboBox1.DataSource = dataSetCountries;  
comboBox1.DisplayMember = "Countries.CountryCode";  
comboBox1.ValueMember = "Countries.Name";
```

Сложно свързване на контрола към списък

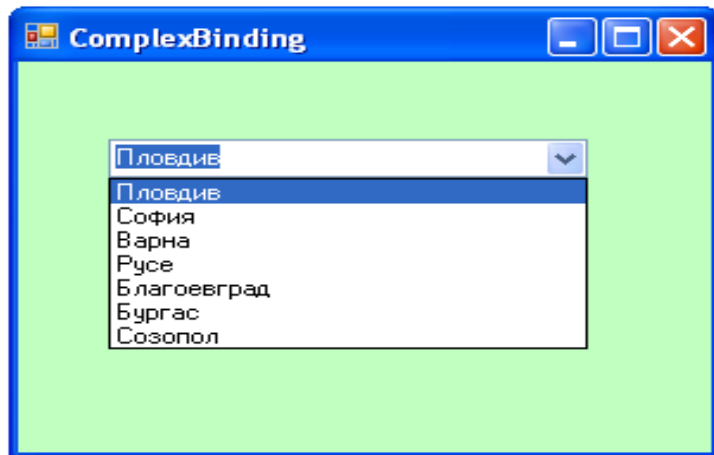
Пример с контрола **ComboBox**

1/2

- ▶ Създаваме нов проект - **ComplexBinding**
- ▶ Задаваме на главната форма Name = **MainForm** и Text = "**Complex Binding**". Променяме името на файла от **Form1.cs** на **MainForm.cs**
- ▶ Поставяме във формата един бутон с име **ButtonShow** и една **ComboBox** контрола с име **ComboBoxTowns**. На свойството **Text** на бутона задаваме стойност **Show**
- ▶ **ComboBox** контролата ще свържем с **масив от символни низове** - имена на градове, а чрез **бутона** ще показваме **стойността, избрана в нея**

Пример сложно свързване с ComboBox

2/2



```
private void ComplexBinding_Load(object sender, EventArgs e)
{
    string [] towns = {"Пловдив", "София", "Варна", "Русе",
                      "Благоевград", "Бургас", "Созопол"};
    comboBoxTowns.DataSource = towns;
    comboBoxTowns.DisplayMember = "";
}
private void buttonShow_Click(object sender, EventArgs e)
{
    MessageBox.Show(comboBoxTowns.SelectedValue.ToString());
}
```

Контролата DataGrid

- ▶ **DataGrid** контролата визуализира таблични данни
 - ▶ Осигурява **навигация** по редове и колони
 - ▶ Позволява **редактиране** на данните
 - ▶ Слаба функционалност
 - ▶ Ще бъде заменена в бъдещи версии на .NET Framework
- ▶ Използва се най-често с ADO.NET **DataSet** и **DataTable**
 - ▶ **DataSource** – задава източника на данни
 - ▶ **DataMember** – задава пътя до данните в рамките на източника

Контролата DataGridView

- ▶ По-важни свойства на **DataGridView**
 - ▶ **ReadOnly** – разрешава / забранява редакция
 - ▶ **CaptionVisible** – показва / скрива заглавието
 - ▶ **ColumnHeadersVisible** – показва / скрива заглавията на колоните
 - ▶ **RowHeadersVisible** – показва / скрива колоната вляво от редовете
 - ▶ **TableStyles** – задава стилове за таблицата
 - ▶ Активен е само първият стил
 - ▶ **MappingName** – задава таблицата, за която се отнася дефинираният стил
 - ▶ **GridColumnStyles** – задава форматирането на отделните колони – заглавие, ширина и др.

Валидация на данни

- ▶ **Валидацията на данни** е необходима, когато в дадена контрола трябва да се допуска **въвеждане само на коректни данни**, например цяло число, валидна дата и др.

В Windows Forms има **стандартни средства за валидация**:

- ▶ **Validating** – **събитие за валидация** на данните в класа **Control**.
 - ▶ На събитието се подава параметър от тип **CancelEventArgs**.
 - ▶ Ако на свойството **Cancel** на този обект се зададе стойност **true**, то на потребителя не се разрешава да напусне контролата.
- ▶ **ErrorProvider** – отбелязва **графично контроли с невалидни данни**.
 - ▶ До контролата с невалидни данни се **появява икона**, а когато показалецът на мишката застане над иконата, се **появява текст с описание на грешката**

Валидация на данни – пример 1

Настоящият пример илюстрира средствата за валидация на данни в Windows Forms

- ▶ събитието **Validating** и
- ▶ контролата **ErrorProvider**

Ще създадем просто приложение, състоящо се от **две форми**:

- ▶ главна форма и
- ▶ форма за въвеждане на ЕГН и година на раждане.

Главната форма ще извиква формата за въвеждане на ЕГН и **година на раждане** и при успешно връщане от нея ще визуализира въведените данни.

Във формата за въвеждане на ЕГН и година на раждане ще сигнализираме на потребителя, когато той въведе некоректни данни.

Валидация на данни – пример 1

- ▶ Стартираме VS.NET и създаваме нов Windows Forms проект.
- ▶ Задаваме на главната форма име **MainForm** и заглавие "**Main Form**". Променяме и името на файла от **Form1.cs** на **MainForm.cs**.
- ▶ Създаваме и формата за въвеждане на ЕГН и година на раждане (**File | Add New Item ... | Windows Form**). Сменяме името ѝ на **ValidationForm**, а това на файла ѝ на **ValidationForm.cs**. Задаваме на свойствата **MinimizeBox** и **MaximizeBox** стойности **false**, а на свойството **FormBorderStyle** стойност **FixedDialog**.

Валидация на данни – пример 1

- ▶ В новосъздадената форма поставяме две текстови полета с имена **textBoxEGN** и **textBoxYear** за въвеждане на ЕГН и година на раждане. Над всяко от тях поставяме по един **Label** с текст, указващ предназначението на контролата. Поставяме и два бутона с имена **ButtonOK** и **ButtonCancel** за потвърждаване и отказване на формата. На свойството **DialogResult** на **ButtonCancel** задаваме стойност **Cancel**.
- ▶ Поставяме във формата контрола **ErrorProvider** с име **errorProvider**.
- ▶ Добавяме обработчик на събитието **Validating** на **textBoxEGN** контролата:

Валидация на данни – пример

```
-- private void textBoxEGN_Validating(object sender, CancelEventArgs e)
{
    ValidateEGN();
}
private bool ValidateEGN()
{
    if (IsEgnValid(textBoxEGN.Text))
    {
        errorProvider.SetError(textBoxEGN, "");
        return true;
    }
    else
    {
        errorProvider.SetError(textBoxEGN, "Невалидно ЕГН!");
        return false;
    }
}
private bool IsEgnValid(string aText)
{
    if (aText.Length != 10)
    {
        return false;
    }
    for (int i = 0; i < aText.Length; i++)
    {
        if (!Char.IsDigit(aText[i]))
        {
            return false;
        }
    }
    return true;
}
```

Валидация на данни – пример 1

- ▶ Аналогично добавяме обработчик на събитието **Validating** на **textBoxYear** контролата

```
private void textBoxYear_Validating(object sender, CancelEventArgs e)
{
    ValidateYear();
}
private bool ValidateYear()
{
    if (IsYearValid(textBoxYear.Text))
    {
        errorProvider.SetError(textBoxYear, "");
        return true;
    }
    else
    {
        errorProvider.SetError(textBoxYear, "Невалидна година!");
        return false;
    }
}
private bool IsYearValid(string aText)
{
    string year = textBoxYear.Text;
    if (year.Length != 4)
    {
        return false;
    }
    for (int i = 0; i < aText.Length; i++)
    {
        if (!Char.IsDigit(aText[i]))
        {
            return false;
        }
    }
    return true;
}
```

Валидация на данни – пример 1

```
private void buttonOK_Click(object sender, EventArgs e)
{
    if (ValidateForm())
    {
        DialogResult = DialogResult.OK;
    }
    else
    {
        MessageBox.Show(
            "Моля въведете валидни стойности във всички полета!",
            "Грешка", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private bool ValidateForm()
{
    if (!ValidateYear())
    {
        return false;
    }
    if (!ValidateEGN())
    {
        return false;
    }
    string egn = textBoxEGN.Text;
    string year = textBoxYear.Text;
    if (egn.Substring(0, 2) == year.Substring(2, 2))
    {
        errorProvider.SetError(buttonOK, "");
        return true;
    }
    else
    {
        errorProvider.SetError(buttonOK,
            "Годината на раждане не съответства на ЕГН-то!");
        return false;
    }
}
```

Валидация на данни – пример 1

- ▶ Добавяме свойства, чрез които да извличаме въведените във формата ЕГН и година на раждане:

```
public string EGN
{
    get
    {
        return textBoxEGN.Text;
    }
}
public string Year
{
    get
    {
        return textBoxYear.Text;
    }
}
```

Валидация на данни – пример 1

- Добавяме в главната форма бутон с име **ButtonShow** и текст **Форма за въвеждане**, две текстови кутии – **textBoxEGN** и **textBoxYear** и два етикета. В обработчика на събитието **Click** на този бутон ще извикваме формата за въвеждане на ЕГН и година на раждане в модален режим и при успешно връщане от нея ще визуализираме въведените данни:

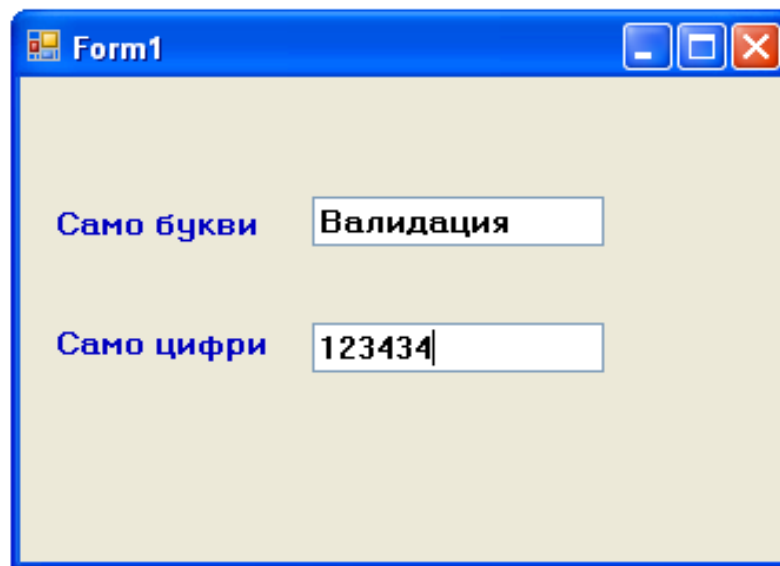
```
private void buttonShow_Click(object sender, EventArgs e)
{
    ValidationForm validationform = new ValidationForm();
    if (validationform.ShowDialog() == DialogResult.OK)
    {
        string s = String.Format("ЕГН: {0}\nГодина: {1}",
            validationform.EGN, validationform.Year);
        MessageBox.Show(s, "Резултат");
        textBoxEGN.Text = validationform.EGN;
        textBoxYear.Text = validationform.Year;
    }
    validationform.Dispose();
}
```

Валидация на данни – пример 1

The image shows two overlapping Windows forms. The background form, titled 'MainForm', has a light beige background and contains two labels: 'ЕГН' and 'Година'. The 'ЕГН' label is next to a text box containing the value '9412034467'. The 'Година' label is next to a text box containing the value '1994'. Below these is a button labeled 'Форма за въвеждане'. The foreground form, titled 'ValidationForm', has a yellow background and contains two labels: 'Въведете ЕГН:' and 'Въведете година на раждане:'. Each label is next to an empty text box. At the bottom of the 'ValidationForm' are two buttons: 'OK' and 'Cancel'.

Валидация на данни – Пример 2

- ▶ Ще демонстрираме контрол на данни по време на въвеждане на данни
- ▶ Създаваме нова форма с две текстови полета – **textBoxLetter** и **textBoxNumber** за въвеждане съответно само на букви и само на цифри във формите.



Валидация на данни – Пример 2

```
private void textBoxLetter_KeyPress(object sender, KeyPressEventArgs e)
{
    if (char.IsLetter(e.KeyChar) || e.KeyChar == 8)
    {
        e.Handled = false;
    }
    else
    {
        e.Handled = true;
    }
}

private void textBoxNumber_KeyPress(object sender, KeyPressEventArgs e)
{
    if (char.IsNumber(e.KeyChar) || e.KeyChar == 13)
    {
        e.Handled = false;
    }
    else
    {
        e.Handled = true;
    }
}
```

Проверява ASCII кода
на BS (backspace)

Проверява ASCII кода
на CR (carriage return)

Благодаря за вниманието!

