

Programare Windows

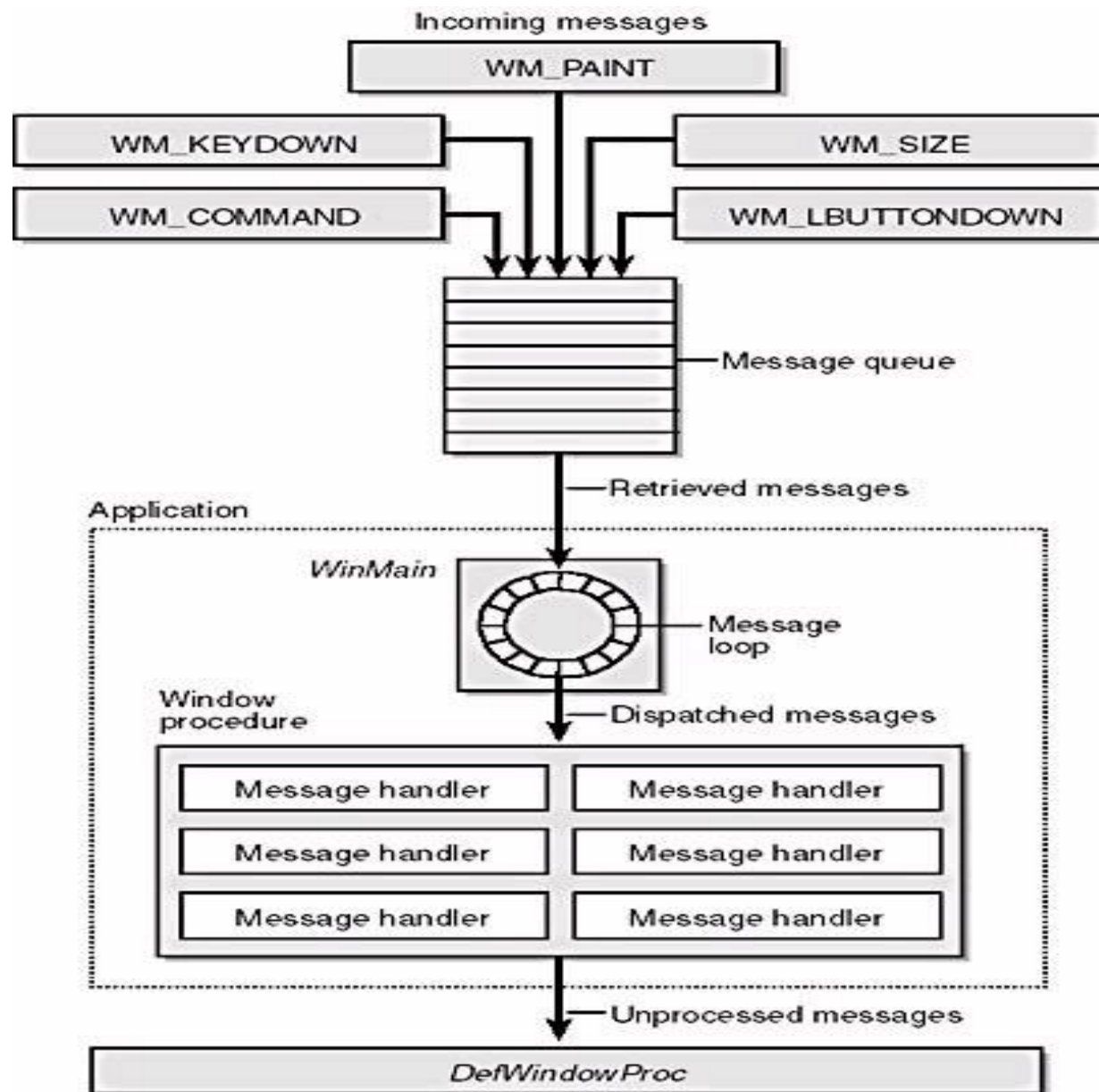
Modele de programare:

- Windows SDK (C)
- Windows MFC – Microsoft Foundation Classes (C, C++)
- Windows Forms (.NET , C#, etc.)
- WPF – Windows Presentation Foundation (.NET, C#)

Windows SDK

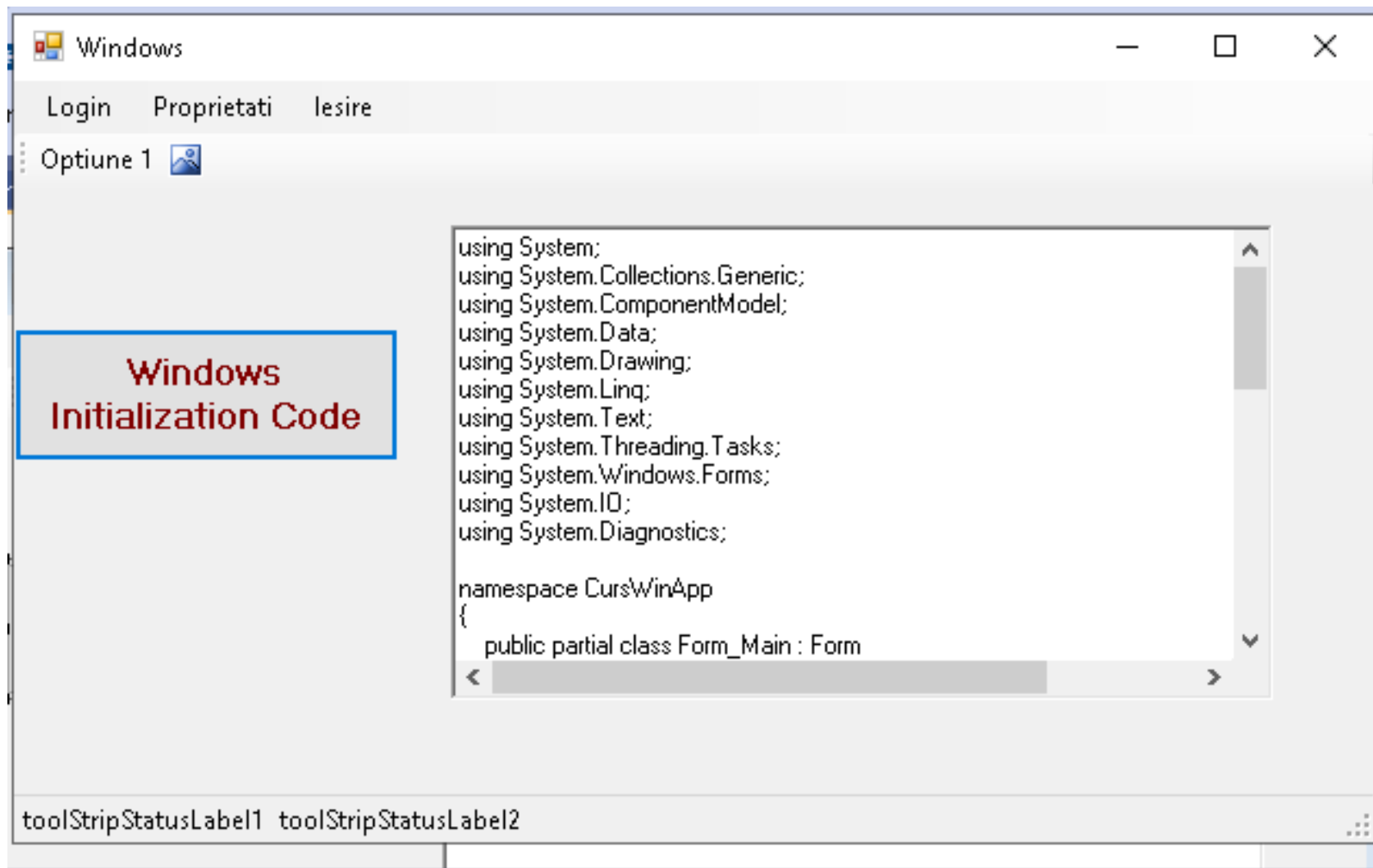
Forma generala a aplicatiei.

- ✓ Structura fereastră.
- ✓ Buclă de mesaje.
- ✓ Structura mesaj.



Forma vizuala a ferestrei

- ✓ Title bar.
- ✓ Menu bar.
- ✓ Toolbar.
- ✓ System menu.
- ✓ Scroll bars.
- ✓ Status bar.
- ✓ Client area.



System menu



Windows

- Evenimentele genereaza mesaje.
- Coada de mesaje:
 - Sistem.
 - Pentru fiecare fir cu interfata.
- Aplicatia functioneaza prin trimiterea de mesaje si tratarea corespunzatoare a acestora.

Windows

❖ O schema simplificata a acestui mecanism este:

- ✓ Windows receptioneaza evenimentul (actiunea utilizatorului, scurgerea unui interval de timp, etc.)
- ✓ Windows transforma actiunea în mesaj.
- ✓ Fereastra programului primeste mesajul.
- ✓ Programul executa un cod (numit bucla de mesaje) ce preia si distribuie mesajele determinand astfel functia ce se va executa (functia ce va trata mesajul).

Bucula de mesaje

```
MSG msg; // Structura ce descrie mesajul.  
while( GetMessage( &msg, NULL, 0, 0 ) )  
{  
    TranslateMessage( &msg );  
    DispatchMessage( &msg );  
}
```

Structura mesaj:

```
typedef struct tagMSG {  
    HWND hwnd; UINT message;  
    WPARAM wParam; // Parametrul 1 al mesajului  
    LPARAM lParam; // Parametrul 2 al mesajului  
    DWORD time; POINT pt;  
} MSG;
```

Observatie. Nu exista un asemenea cod in WinForms.

Procedura fereastra

Apelata de sistemul de operare. Prototip:

```
LRESULT CALLBACK WndProc(HWND hwnd, UINT uMsg,  
    WPARAM wParam, LPARAM lParam)  
{ switch(uMsg) {  
    case WM_PAINT:  
        Deseneaza(hwnd);  
        break;  
    case WM_DESTROY:  
        PostQuitMessage(0);  
        break;  
    ... } }
```

Etape in creare si lansare app GUI

- Inregistrare fereastră (structura).
- Stabilire attribute pentru fereastră.
- Desenare fereastră.
- Bucla de mesaje (am vazut codul mai sus).

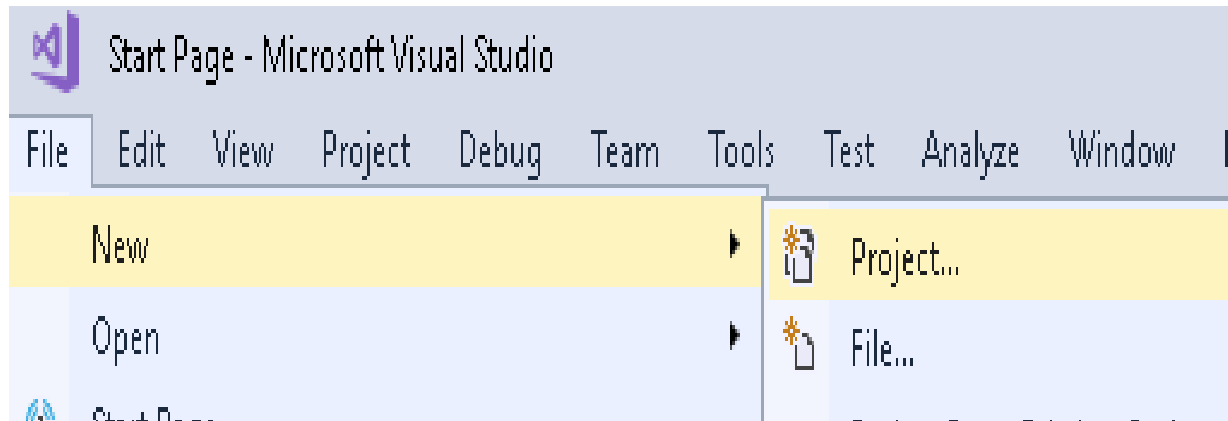
Windows MFC

- MFC – Microsoft Foundation Classes
- Biblioteca de clase (C++) peste functiile din SDK. Exemple de clase:
- CWinApp, CFrameWnd, CDocument, CView, etc.
- Nu le studiem.

.NET - Windows Form

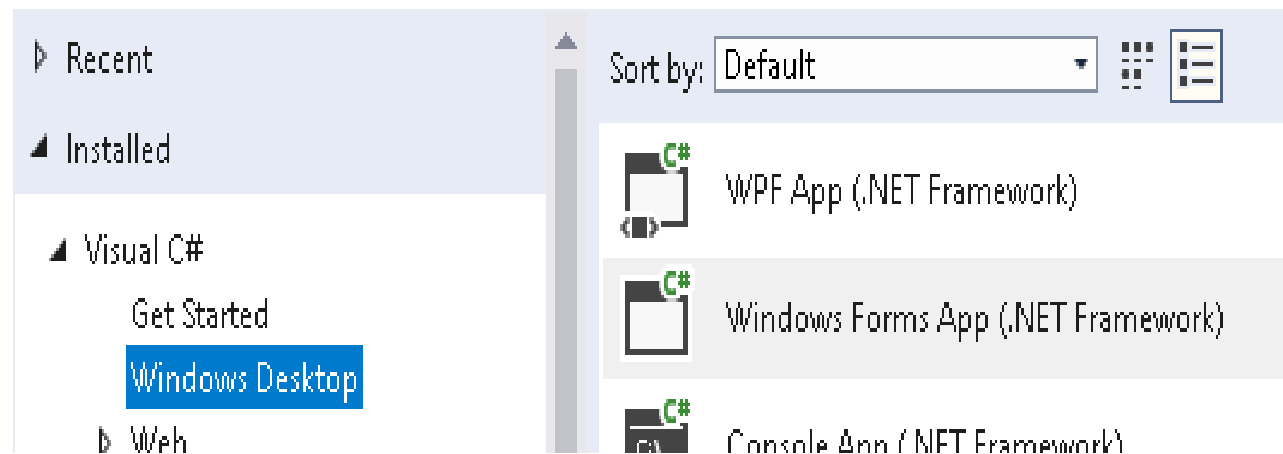
- WinForms este un “layer” peste controalele standard din Windows (TextBox, Button, etc.).
- Consideratii generale WinForms.
 - Are o anumita maturitate, testat mult timp.
 - Controale construite de terti.
 - Usor de utilizat.
 - Intellisense.

Hello WinForms



Tip proiect

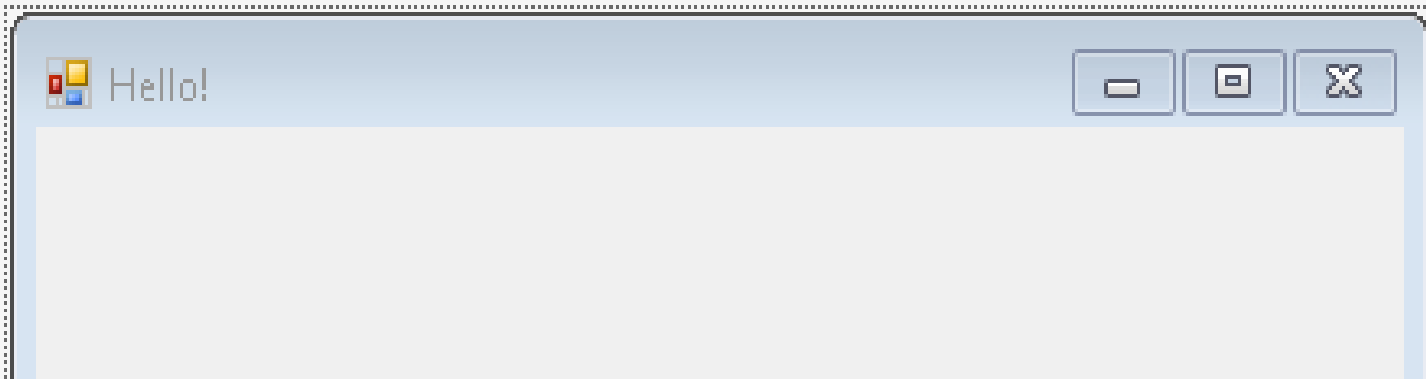
New Project



Nume proiect

Name:	WindowsFormsApp2
Location:	D:\Documente\Cursuri\Curs special NET\WinForms\
Solution name:	WindowsFormsApp2
Framework:	.NET Framework 4.6.1 ▼

Windows forms



Windows forms






- Fereastra principala a aplicatiei Windows este in fapt un container ce contine controale Windows.
- SO va inregistra ID-ul acestei ferestre in structurile sale interne. ID unic la nivel de SO.
- Control caracterizat de:
 - Proprietati.
 - Evenimente.



Control

- Controalele Windows Forms sunt componente reutilizabile ce incapsuleaza interfata cu utilizatorul si sunt folosite pe partea de client a unei aplicatii Windows.
- Clasa de baza din .NET este **Control**.
- Exemplul ce urmeaza exemplifica proprietati pentru o fereasta.

Properties

Form 1 System.Windows.Forms.Form

⊕ (ApplicationSettings)	
⊕ (DataBindings)	
(Name)	Form 1
AcceptButton	(none)
AccessibleDescription	
AccessibleName	
AccessibleRole	Default
AllowDrop	False
AutoScaleMode	Font
AutoScroll	False
⊕ AutoScrollMargin	0, 0
⊕ AutoScrollMinSize	0, 0
AutoSize	False
AutoSizeMode	GrowOnly
AutoValidate	EnablePreventFocusChange
BackColor	 Control
BackgroundImage	 (none)
BackgroundImageLayout	Tile
CancelButton	(none)
CausesValidation	True
ContextMenuStrip	(none)
ControlBox	True
Cursor	Default

Properties

Form 1 System.Windows.Forms.Form






Icons: List, Sort, Copy, Paste, Undo, Redo, Help

DoubleBuffered	False
Enabled	True
Font	Microsoft Sans Serif, 8.25pt
ForeColor	ControlText
FormBorderStyle	Sizable
HelpButton	False
Icon	(Icon)
ImeMode	NoControl
IsMdiContainer	False
KeyPreview	False
Language	(Default)
Localizable	False
Location	0, 0
Locked	False
MainMenuStrip	(none)
MaximizeBox	True
MaximumSize	0, 0
MinimizeBox	True
MinimumSize	0, 0
Opacity	100%
Padding	0, 0, 0, 0
RightToLeft	No
RightToLeftLayout	False

Text
The text associated with the control.

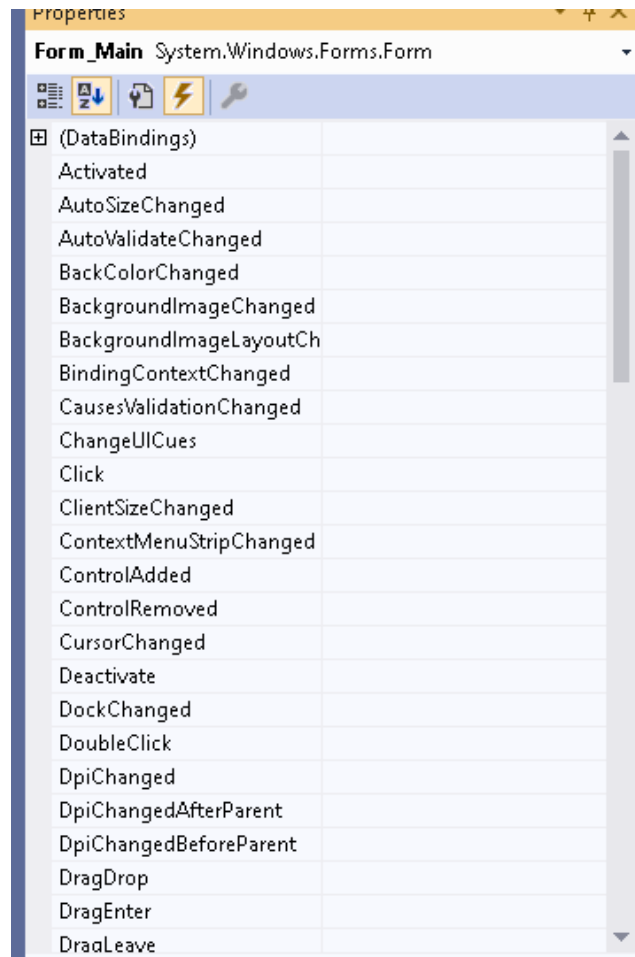
Properties

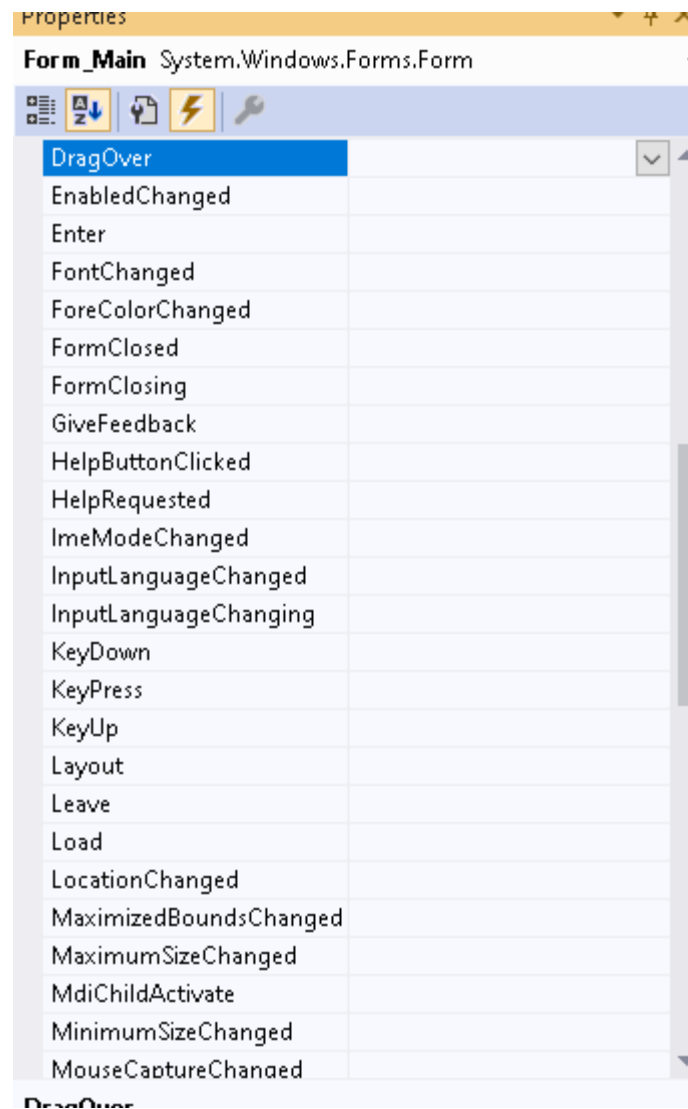
Form 1 System.Windows.Forms.Form

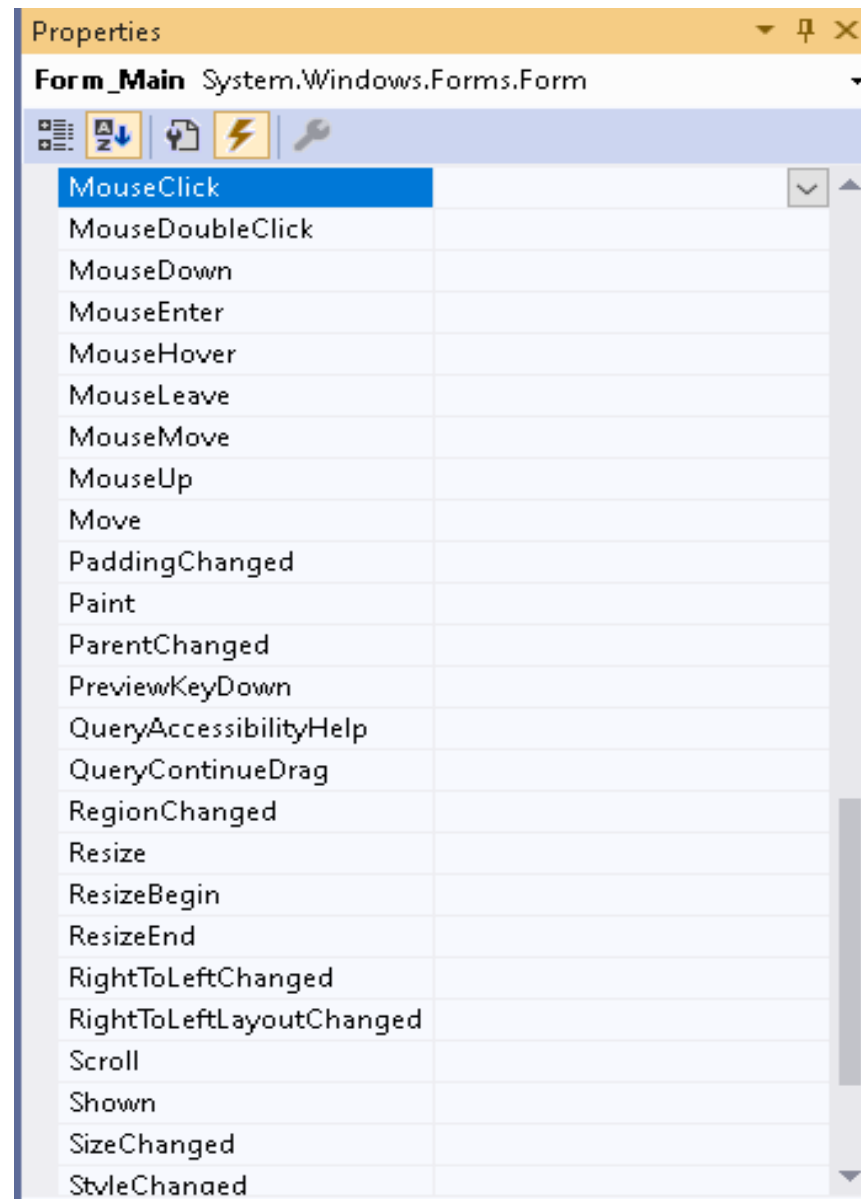






⊕ Location	0, 0
Locked	False
MainMenuStrip	(none)
MaximizeBox	True
⊕ MaximumSize	0, 0
MinimizeBox	True
⊕ MinimumSize	0, 0
Opacity	100%
⊕ Padding	0, 0, 0, 0
RightToLeft	No
RightToLeftLayout	False
ShowIcon	True
→ ShowInTaskbar	True
⊕ Size	436, 304
SizeGripStyle	Auto
→ StartPosition	WindowsDefaultLocation
→ Tag	
→ Text	Hello!
TopMost	False
TransparencyKey	<input type="text"/>
UseWaitCursor	False
→ WindowState	Normal

Evenimente







Evenimente

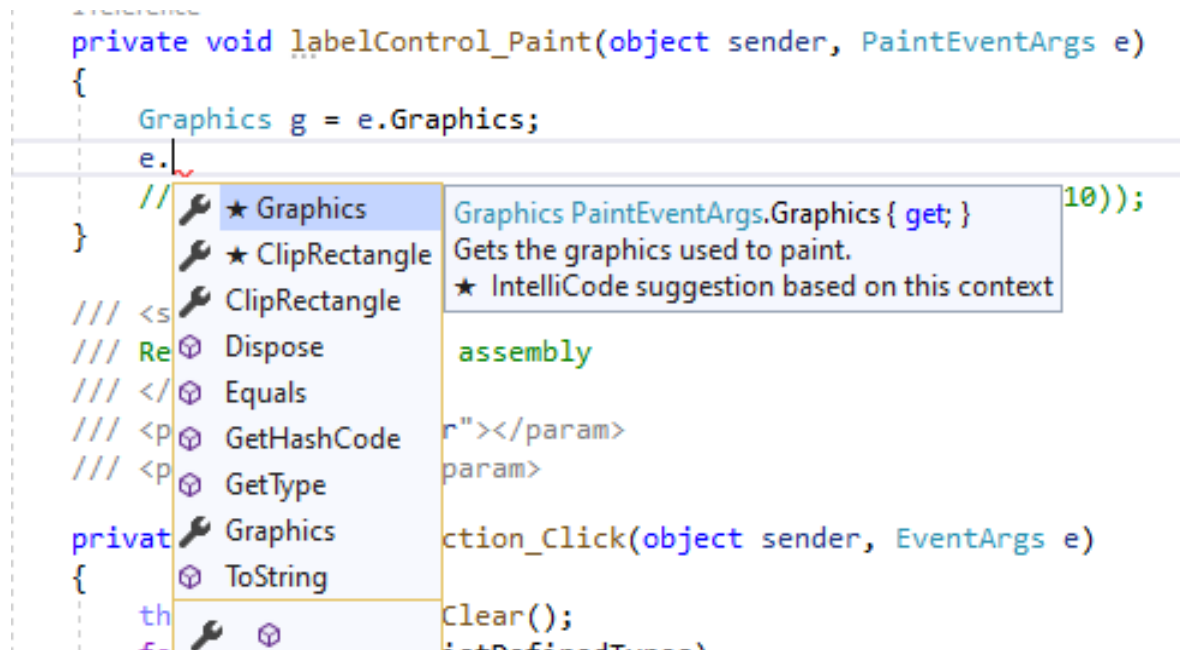
- Mai des folosite:
- Load – înainte de afisarea ferestrei.
- Cele legate de intrarile de la mouse.
- Schimbarea starii controlului (afisare, continut, pozitie, etc.).

Parametru mesaj

- Fiecare control este caracterizat de anumite evenimente specifice.
- Parametrul mesajului in Windows forms este descris de clasa **EventArgs** sau o clasa derivata din **EventArgs**.
- Exemplu (trateaza event **Paint** la fereastra):

```
protected override void OnPaint(PaintEventArgs e)
```

PaintEventArgs



Cod generat. Clasa Form1 derivata din Form

// Aceasta este structura pentru clasa fereastra

```
using System;
```

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

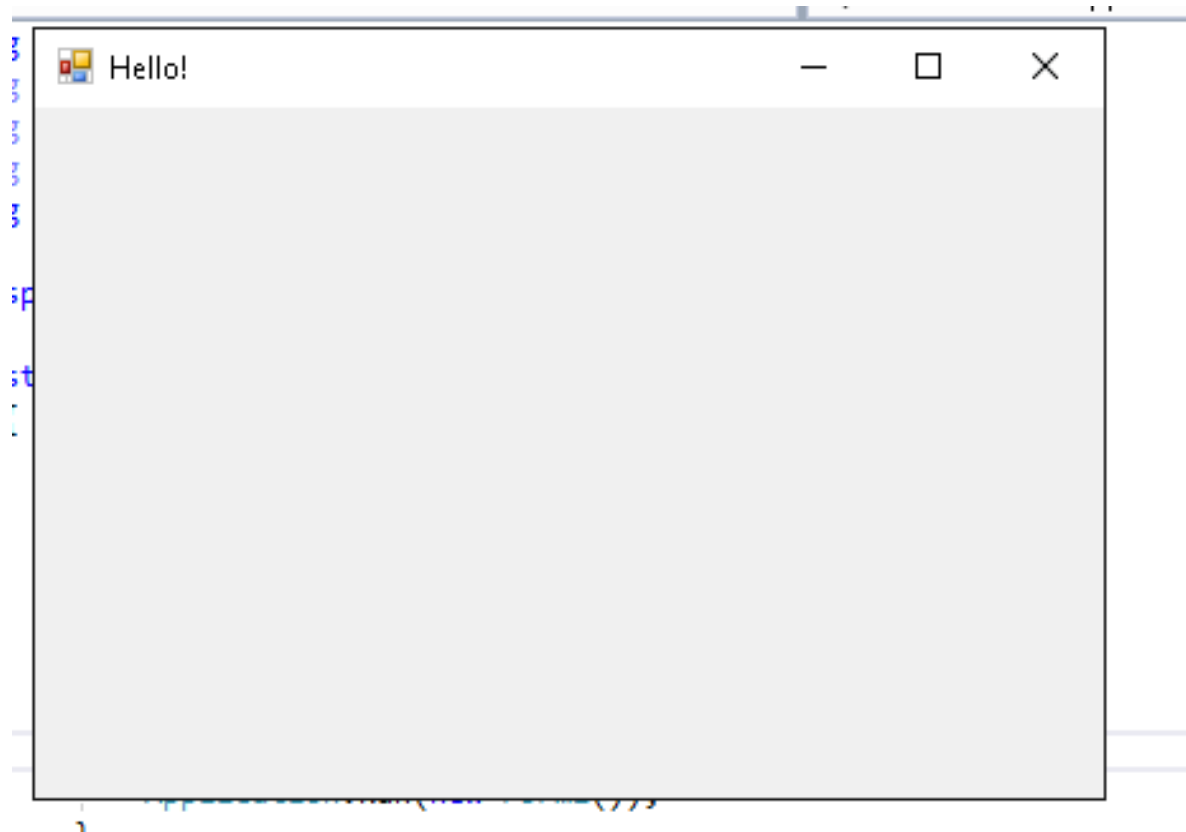
```
namespace WindowsFormsApp1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

Entry point

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

namespace WindowsFormsApp1
{
    static class Program
    {
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

Rezultat executie



Recapitulare executie program

- SO Windows creaza si initializeaza un proces care foloseste metoda Main ca “entry point” pentru executie. Operatiile efectuate sunt urmatoarele:
 - Instanta a clasei derivata din Forms.
 - Atribuire valori la proprietati.
 - Apelare metoda Run ce are ca parametru instanta ferestrei => afisare fereastră.
 - Asteapta si proceseaza mesaje din interactiunea cu utilizatorul.
 - Cand aplicatia se inchide: Application.Run returneaza apoi Main returneaza si in final executia se termina.

Desenare in zona client (fereastră)

- Trebuie sa suprascriem functia OnPaint().

```
protected override void OnPaint(  
    PaintEventArgs e)
```

```
{  
    base.OnPaint(e);  
    DrawString();  
}
```

DrawString()

Spatiu de nume: System.Drawing

```
private void DrawString()  
{  
    Graphics formGraphics = this.CreateGraphics();  
    string drawString = "Hello Windows Form!";  
    Font drawFont = new System.Drawing.Font("Arial", 16);  
    SolidBrush drawBrush = new SolidBrush(Color.Black);  
    float x = 150.0F;  
    float y = 50.0F;  
    StringFormat drawFormat = new StringFormat();  
    formGraphics.DrawString(drawString, drawFont, drawBrush, x, y, drawFormat);  
    drawFont.Dispose();  
    drawBrush.Dispose();  
    formGraphics.Dispose();  
}
```

Executie



Graphics

- Before you can draw lines and shapes, render text, or display and manipulate images with GDI+, you need to create a [Graphics](#) object.
The [Graphics](#) object represents a GDI+ drawing surface, and is the object that is used to create graphical images.
- There are two steps in working with graphics:
 - Creating a [Graphics](#) object.
 - Using the [Graphics](#) object to draw lines and shapes, render text, or display and manipulate images.

GDI+

- **(MSDN) GDI+ is the portion of the Windows operating system** that provides two-dimensional vector graphics, imaging, and typography. GDI+ improves on GDI (the Graphics Device Interface included with earlier versions of Windows) by adding new features and by optimizing existing features.

Graphics

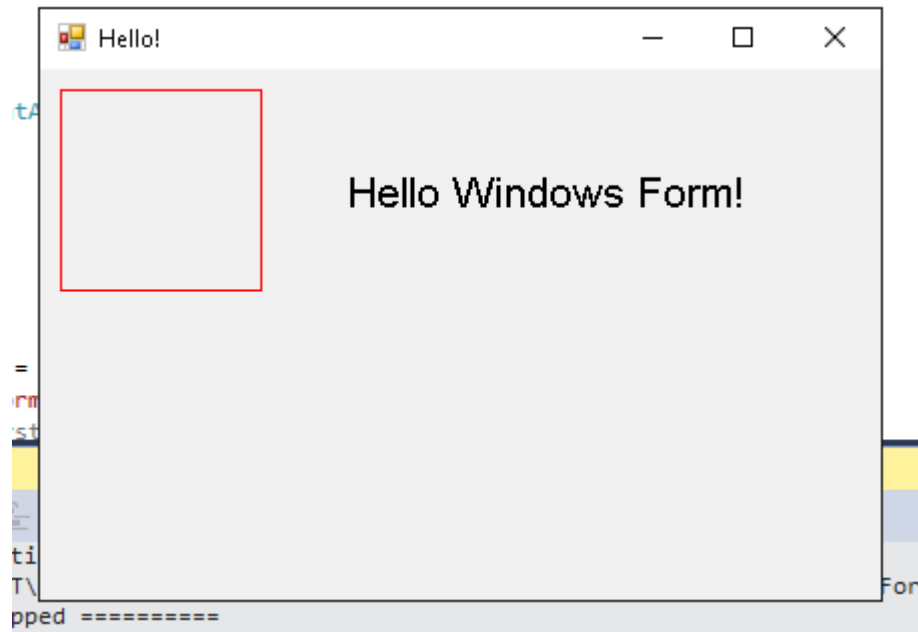
- Pentru a “desena” (afisarea unui text constituie operatie de desenare) e nevoie de urmatoarele obiecte:
- Graphics – e asociat cu un context de dispozitiv (Device Context) si indica dispozitivul pe care vom desena.
- Pen -
- Brush -
- Color -
- Font - optional

Desenam un patrat

// Tratatam evenimentul Paint al ferestrei.

```
private void Form1_Paint(  
    object sender, PaintEventArgs e)  
{  
    g = e.Graphics;  
    g.DrawRectangle(new Pen(Color.Red),  
        new Rectangle(10, 10, 100, 100));  
}
```

Executie



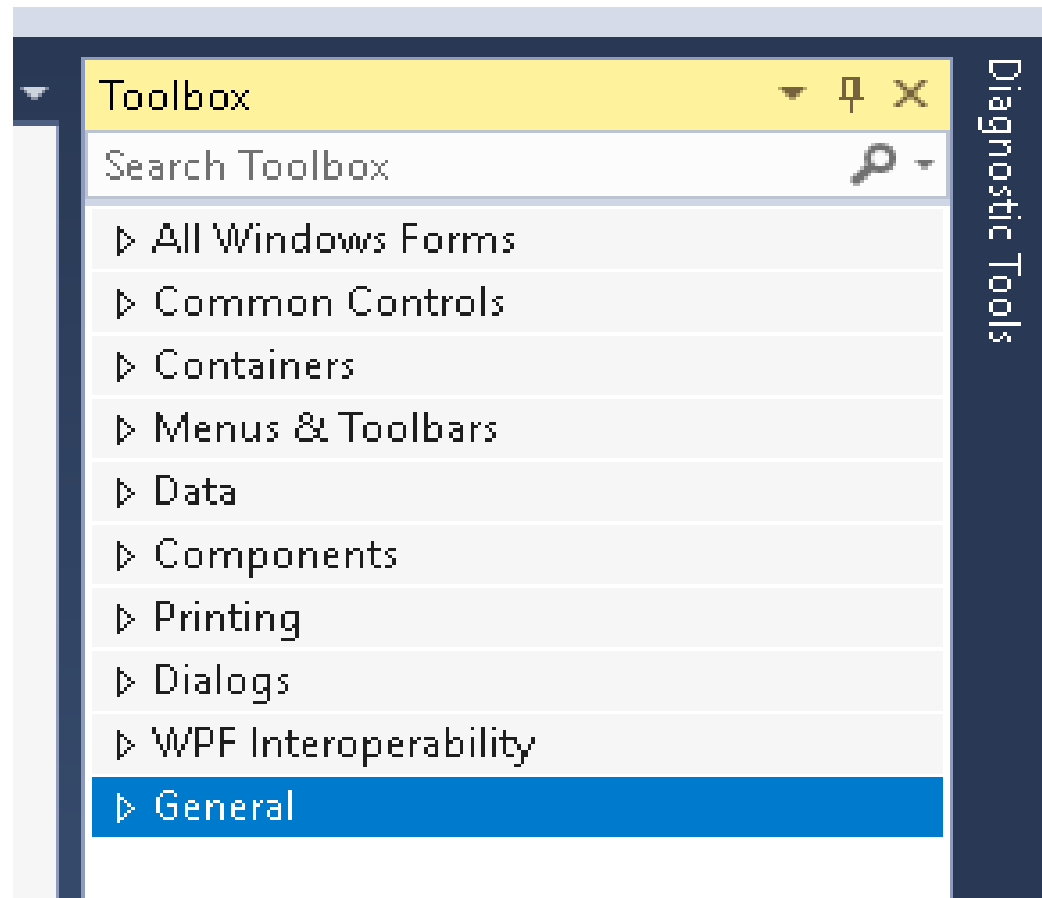
Graphics

- Consultati MSDN pentru a vedea toate facilitatile oferite de Graphics.

Controale

- O lista a controalelor ce pot fi folosite este afisata in fereastra Toolbox din VS.

Controale



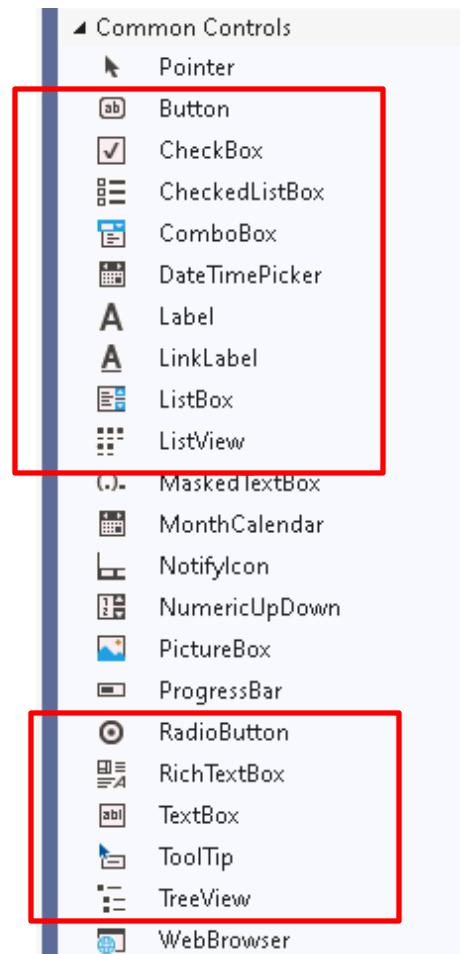
Controale comune

- Des utilizate.
 - Afiseaza text sau preiau text.
 - Label, TextBox, RichTextBox.
 - Butoane:
 - Button, CheckBox, RadioButton.
- Contin o colectie de articole:
 - ComboBox, ListBox, ListView, TreeView.

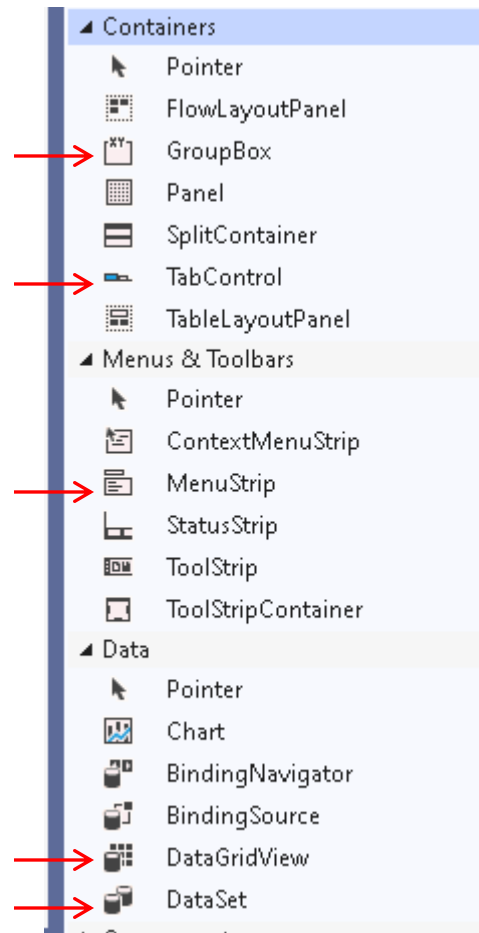
Handler evenimente

- `private void Metoda(object sender, EventArgs e);`
- Al doilea parametru poate fi si o clasa derivata din `EventArgs`.

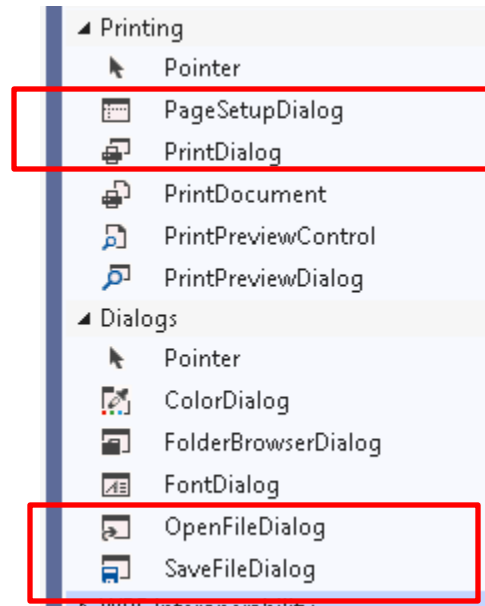
Common Controls



Common Controls



Common Controls



Label control

- Afisare text,
- Afisare text si imagine.
- Afisare imagine.

Label. Proprietati.

- Name
- Text
- ForeColor, Font
- Enable
- Tag
- Image
- Visible, etc.

Label. Evenimente.

- Click
- MouseEnter
- MouseLeave
- Paint, etc.

Label control. Exemple cod

```
private void labelControl_Click(object sender, EventArgs e)
{
    labelControl.ForeColor = Color.Blue;
    labelControl.Text = "Click event";
}

private void labelControl_MouseEnter(object sender, EventArgs e)
{
    labelControl.ForeColor = Color.Blue;
    labelControl.Text = "MouseEnter event";
}

private void labelControl_MouseLeave(object sender, EventArgs e)
{
    this.labelControl.Text = this.textLabelControl;
    this.labelControl.ForeColor = foreLabelControl;
    richTextBox1.Clear();
    richTextBox1.Text = "MouseLeave event pe labelControl";
}

private void labelControl_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    g.FillRectangle(Brushes.Red, new Rectangle(5, 5, 10, 10));
}
```

TextBox

- Afisare text.
- Preluare informatii de la tastatura.
- Informatiile afisate / preluate sunt de tip String.
- Text: “single line” sau “Lines – String[] Array”

TextBox. Element in Toolbox



TextBox. Proprietati.

- Name
- BackColor
- ForeColor
- **Enabled**
- **ReadOnly**
- **MaxLenght**
- **Text**
- Lines, MultiLine
- Password Char
- Tag, etc.

TextBox. Evenimente.

- TextChanged (prin cod).
- Evenimente de la mouse:
 - MouseEnter, MouseLeave, MouseHover, MouseClick.
- Evenimente de la tastatura:
 - TextChanged
 - KeyPress, KeyDown, KeyUp.

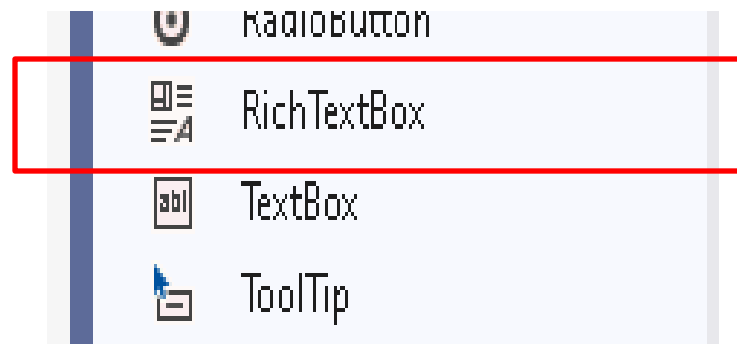
TextBox

- Preluare text:
`String temp = this.text_Name.Text;`
- Modificare continut:
`this.text_Name.Text = "Continut nou: 1234";`

Observatie

- ☐ Daca un TextBox este folosit pentru a prelua numere reale, atunci va trebui sa scriem cod in vederea validarii intrarilor numai de la tastele numerice si tastele plus (+), minus (-) si punct (.) sau virgula (,).
- ☐ Atentie la modificari de la tastatura si cele din cod.

RichTextBox. Element in Toolbox



RichTextBox. Proprietati.

With the [RichTextBox](#) control, the user can enter and edit text. The control also provides more advanced formatting features than the standard [TextBox](#) control. Text can be assigned directly to the control, or can be loaded from a rich text format (RTF) or plain text file. The text within the control can be assigned character and paragraph formatting.

RichTextBox. Proprietati.

- Name
- Text
- Enabled
- ReadOnly
- Tag, etc.

RichTextBox. Evenimente.

- Evenimente de la mouse:
 - MouseEnter, MouseLeave, MouseHover, etc.
- Evenimente de la tastatura:
 - KeyPress, KeyDown, KeyUp.
- TextChanged: programatic sau de la tastatura.

Controale de tip Button

- Button
 - CheckBox
 - RadioButton
-
- Comun pentru acestea este faptul ca au ca eveniment foarte des utilizat “Click mouse” sau “Enter” cand au focusul.

Button

- Suporta text si imagine.
- Proprietati:
 - Name
 - Text
 - Tag
 - Enabled, etc.

Button. Evenimente.

- Evenimente de la tastatura:
 - KeyPress, KeyDown, KeyUp.
- Evenimente de la mouse:
 - MouseEnter, MouseLeave, MouseHover.
- Click.

CheckBox

- Prezinta trei stari: “checked” , “unchecked” sau “indeterminate” echivalent “null” din baze de date. Proprietatea CheckState.
- Folosit in multe scenarii, de la activarea / dezactivarea unui grup de controale pana la determinarea “cursului” unei aplicatii.
- Putem avea selectii multiple.

CheckBox. Proprietati.

- Name
- Enabled
- Checked
- Tag
- Proprietati legate de desenare.

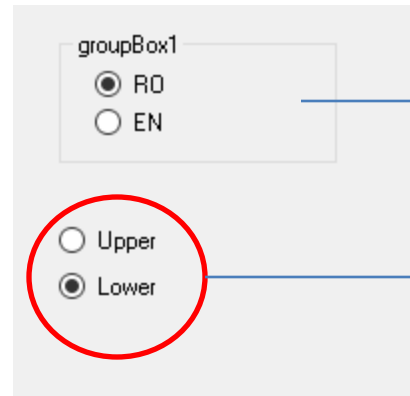
CheckBox. Evenimente.

- Click
- CheckedChanged
- CheckStateChanged
- TextChanged
- Evenimente generate de mouse.
- Evenimente generate de la tastatura.

RadioButton

- Prezinta trei stari ca si CheckBox.
- Functioneaza in grupuri de minim doua controale RadioButton.
- Gestionare automata cand sunt plasate in grup. Se considera grup implicit controlul in care sunt plasate controalele RadioButton.
- Se folosesc cand e nevoie sa facem o unica alegere.

RadioButton



Grupate la
nivel de
groupBox1

Grupate la nivel de fereastra

RadioButton. Proprietati.

- Name
- Enabled
- Text
- Tag
- Checked, etc.

RadioButton. Evenimente.

- Click.
- Evenimente generate de mouse.
- Evenimente generate de la tastatura.

Controale ce contin o colectie de articole

- ListBox
- ComboBox
- ListView
- TreeView

ListBox

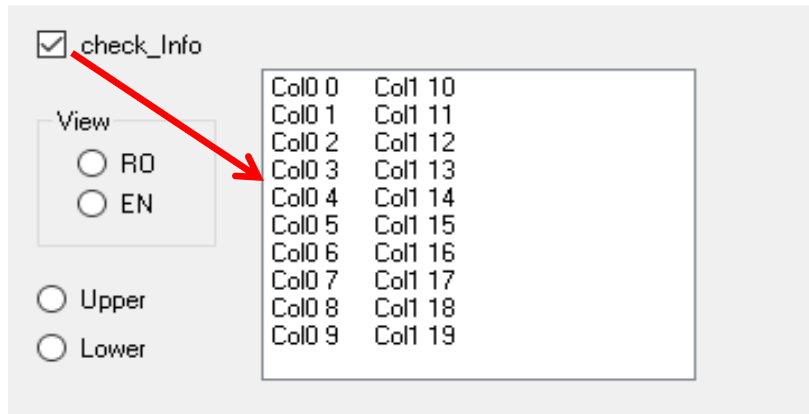
- Prezinta o colectie de articole (zero based index).
- Permite selectie simpla sau multipla.
- Permite afisare simpla sau pe mai multe coloane.
- Operatii asupra continutului LB: adaugare, stergere, editare, cautare.

ListBox. Proprietati.

- Name
- Items – colectie de articole.
- Sorted
- Tag
- SelectionMode (one, multiple)
- DataSource
- DisplayMember

ListBox

```
private void checkBox1_Click(object sender, EventArgs e)
{
    if (checkBox1.Checked)
    {
        for(int i=0; i<10; i++)
        {
            string temp = String.Format("Col0 {0}\tCol1 {1}", i, i + 10);
            listBox_Info.Items.Add(temp);
        }
    }
    else
    {
        listBox_Info.Items.Clear();
    }
}
```



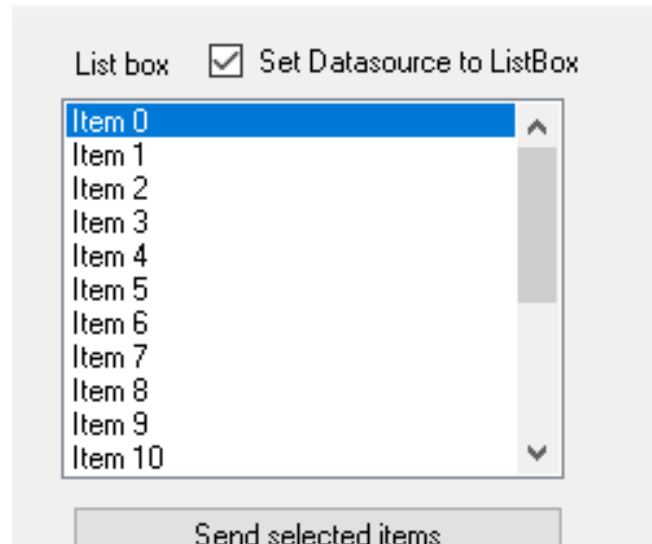
The screenshot shows a Windows application window with a light gray background. On the left side, there is a checkbox labeled "check_Info" which is checked. Below it is a "View" button. To the right of the "View" button is a list box containing 10 rows of text, each with two columns: "Col0" and "Col1". A red arrow points from the "View" button to the list box. Below the list box, there are three radio buttons: "RO", "EN", and "Upper".

Col0 0	Col1 10
Col0 1	Col1 11
Col0 2	Col1 12
Col0 3	Col1 13
Col0 4	Col1 14
Col0 5	Col1 15
Col0 6	Col1 16
Col0 7	Col1 17
Col0 8	Col1 18
Col0 9	Col1 19

ListBox - DataSource

```
private void checkBox_SetDS_Click(object sender, EventArgs e)
{
    List<String> item = new List<string>();
    listBox_Info.DataSource = null; // Ne asiguram ca nu mai e conectat la o sursa de date
    this.checkBox1.Checked = false; // ???
    for (int i = 0; i < 20; i++)
        item.Add("Item " + i.ToString());
    if (checkBox_SetDS.Checked)
    {
        listBox_Info.DataSource = item;
    }
    else
    {
        listBox_Info.Items.Clear();
    }
}
```

ListBox – executie cu DataSource



ListBox – Selectie multipla

```
private void btn_Transfer_Click(object sender, EventArgs e)
{ // Vezi si SelectedItems
    ListBox.SelectedIndexCollection indices =
        listBox_Info.SelectedIndices;
    foreach(var x in indices)
    {
        string t = listBox_Info.Items[(int)x].ToString();
        richTextBox1.AppendText("\n" + t);
    }
}
```

ListBox -

The screenshot displays a Windows Forms application interface. On the left, a text area contains the text "Item 1", "Item 3", "Item 4", "Item 6", and "Item 8". Above this text area is a button labeled "Windows Initialization Code". To the right of the text area are two buttons: "Reflection" and "MSIL - view". Further right is a "Label control" with an empty text box. Below the "Label control" are three radio buttons: "RO", "EN", and "Upper". To the right of these radio buttons is a checkbox labeled "check_Info". On the far right, there is a "List box" control. Above it is a checkbox labeled "Set Datasource to ListBox" which is checked. The list box contains ten items, labeled "Item 0" through "Item 10". Items "Item 1", "Item 3", "Item 4", "Item 6", and "Item 8" are highlighted in blue. Below the list box is a button labeled "Send selected items".

ListBox. Evenimente.

- SelectedIndexChanged
- SelectedValueChanged

❏ Cod:

```
private void listBox_Info_SelectedIndexChanged(  
    object sender, EventArgs e  
{  
    // Vezi si SelectedItem  
    int index = this.listBox_Info.SelectedIndex;  
    if (index == -1)  
        { // Nu exista selectie }  
}
```


ListBox – Tag property: completare si selectie

Windows Initialization Code

Reflection

MSIL - view

Label control

☒ check_Info

View

☐ RD

☐ EN

List box ☐ Set Datasource to ListBox

Col0 0	Col1 10
Col0 1	Col1 11
Col0 2	Col1 12
Col0 3	Col1 13
Col0 4	Col1 14
Col0 5	Col1 15
Col0 6	Col1 16

Tag property value: Poligon 100

```
if (checkBox1.Checked)
{
    // Tag este la nivel de control si nu articol din control
    Poligon p = new Poligon() { Name = "Poligon 100" , Number = 100 };
    listBox_Info.Tag = p;
    for (int i=0; i<10; i++)
    {
        if (listBox_Info.Tag != null)
        {
            richTextBox1.Clear();
            richTextBox1.Text = "Tag property value: " + ((Poligon)listBox_Info.Tag).Name;
        }
    }
}
```

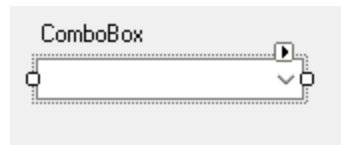
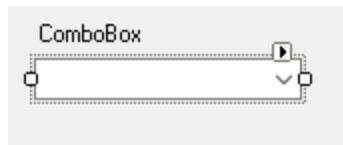
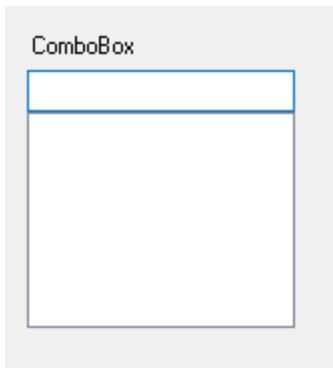
CheckedListBox

- Articolele sunt precedate de un CheckBox.
- ListView prezinta ceva asemanator.

ComboBox

- Prezinta colectie de articole.
- Format din TextBox si ListBox.
- Stiluri de prezentare:
 - Simple
 - DropDown
 - DropDownList

ComboBox - stil



ComboBox. Proprietati.

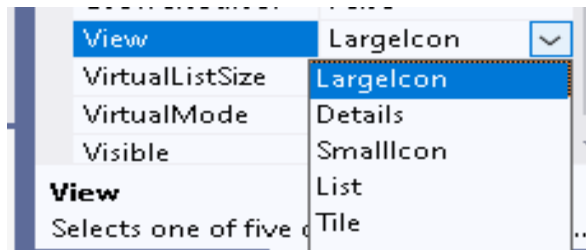
- Name
- Text
- DropDownStyle
- DropDownHeight
- DropDownWidth
- Items
- Etc.

ComboBox. Evenimente.

- Mouse.
- Tastatura.
- SelectedIndexChanged.
- SelectedItemChanged.

ListView







- Control ce afiseaza o colectie de articole.
- Modalitati de afisare:



ListView. List.

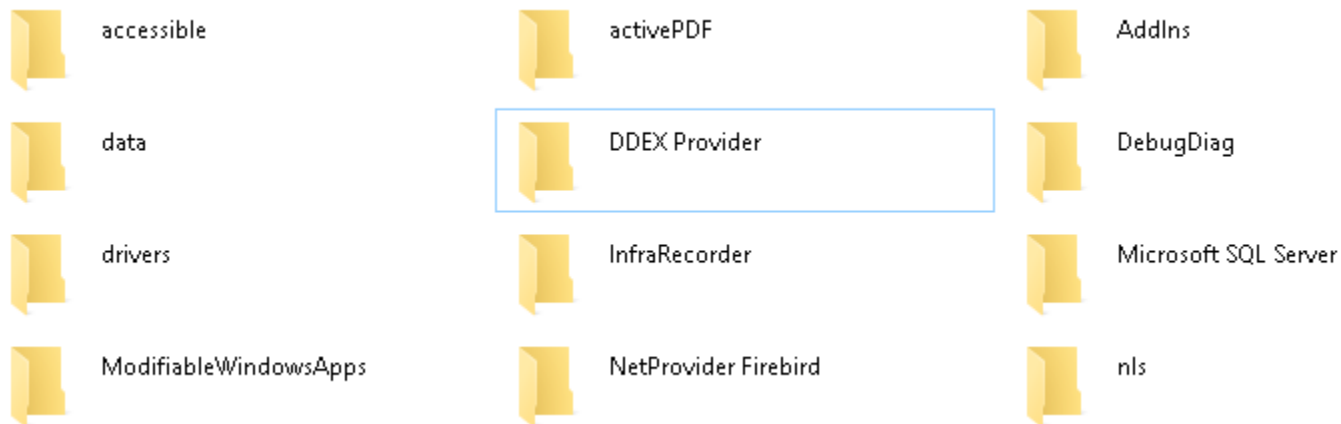
- accessible
- activePDF
- AddIns
- bin
- data
- DDEX Provider
- DebugDiag
- doc
- drivers
- InfraRecorder
- Microsoft SQL Server
- Microsoft VS Code
- ModifiableWindowsApps
- NetProvider Firebird
- nls

ListView. Details.

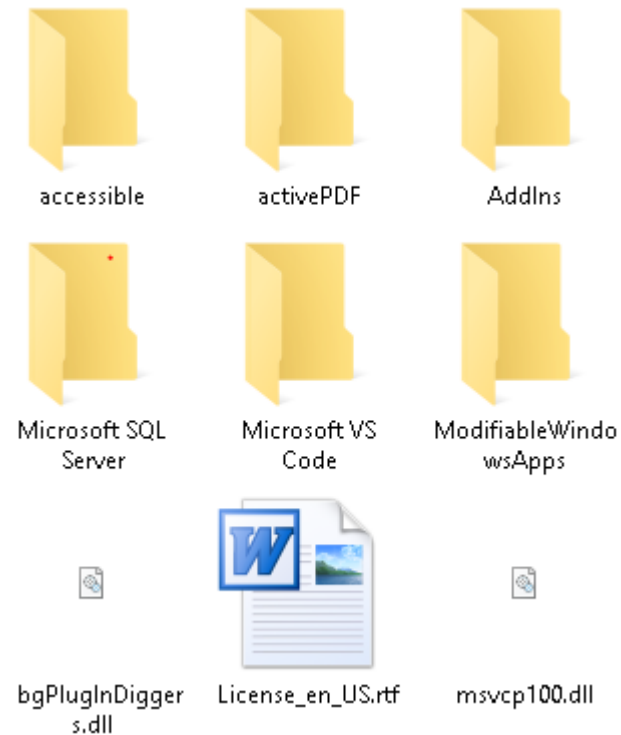
Name	^	Date modified	Type	Size
 accessible		4/8/2016 08:27	File folder	
 activePDF		12/2/2012 09:10	File folder	
 AddIns		12/4/2012 17:34	File folder	
 bin		2/4/2012 17:34	File folder	
 data		2/4/2012 17:34	File folder	
 DDEX Provider		12/2/2012 21:12	File folder	

Date created: 12/2/2012 09:10
Size: 11.2 MB
Folders: PrimoPDF

ListView. Tile.



ListView. LargeIcon



ListView. SmallIcon



ListView. Proprietati.

- Name
- View
- Tag – la nivel de control, articol, grup
- Sorted
- CheckBoxes
- GridLines
- FullRowSelect, etc.

ListView

- Columns

```
listView1.Columns.Add("Coloana 0", 80);
```

```
listView1.Columns.Add("Coloana 1", 70);
```

```
listView1.Columns.Add("Coloana 2", 80);
```

- ListViewItem, ListViewItem.SubItems

```
ListViewItem item = new ListViewItem("Principal 1");
```

```
item.SubItems.Add("sub item 1");
```

```
item.SubItems.Add("sub item 2");
```


```
listView1.Items.Add(item);
```

ListView. Selectie

- SelectedIndexChanged => ListView.SelectedIndexCollection

```
}
1 reference
private void addColumns()
{
    listView1.Columns.Add("Coloana 0")
    listView1.Columns.Add("Coloana 1")
    listView1.Columns.Add("Coloana 2")
}

1 reference
private void listView1_SelectedIndexChanged(object sender, EventArgs e)
{
    ListView.SelectedIndexCollection indices = listView1.SelectedIndices;
    if (indices.Count == 0)
        return;
    object obj = listView1.Items[0].
}
```



ListView. Selectie

```
private void listView1_SelectedIndexChanged(object sender, EventArgs e)
{
    ListView.SelectedIndexCollection indices = listView1.SelectedIndices;
    StringBuilder sb = new StringBuilder();
    if (indices.Count == 0)
        return;
    for (int i = 0; i < indices.Count; i++)
    {
        // Coloana 0. Articolul principal
        string principal = listView1.SelectedItems[i].SubItems[0].Text;
        sb.Append("\n" + principal);
        // Coloana 1, 2, ... Sub articole
        for (int j=1; j < listView1.SelectedItems[i].SubItems.Count; j++)
            sb.Append("\t" + listView1.SelectedItems[i].SubItems[j].Text);
        richTextBox1.Text = sb.ToString();
    }
}
```


TreeView

- Afiseaza o colectie ierarhica de articole etichetate, fiecare reprezentat de un `TreeNode`.

TreeView. Proprietati.

- Name
- Text
- Tag
- Nodes
- Enabled
- FullRowSelect

TreeView. Evenimente

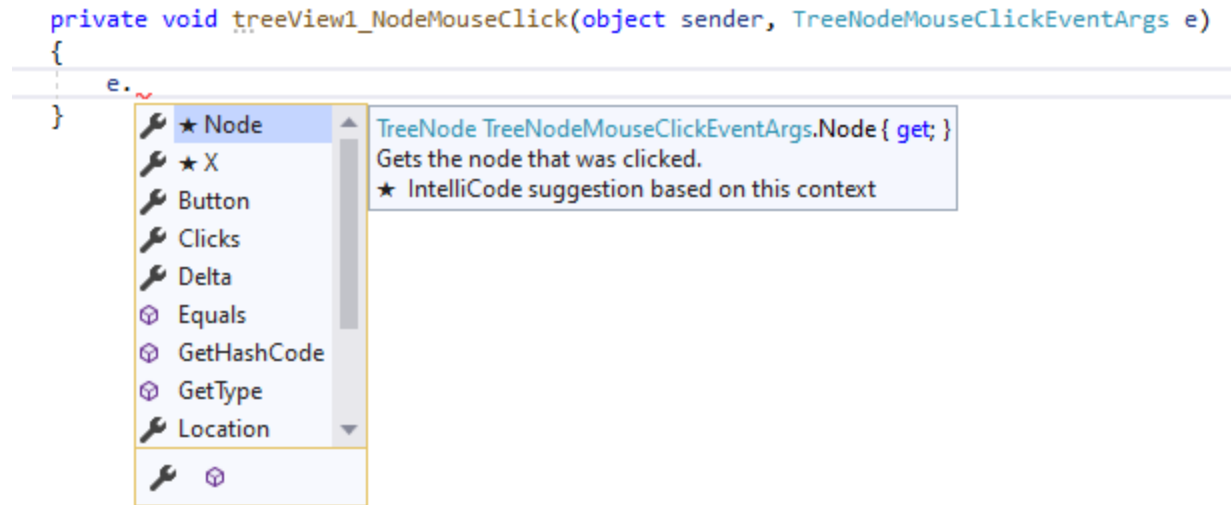
- Click
- Generate de mouse.
 - NodeMouseClicked
- De la tastatura.

TreeView. Completare.

- `treeView1.Nodes.Add("Parent");`
`treeView1.Nodes[0].Nodes.Add("Child 1");`
`treeView1.Nodes[0].Nodes.Add("Child 2");`
`treeView1.Nodes[0].Nodes[1].Nodes.Add("Grandchild");`
`treeView1.Nodes[0].Nodes[1].Nodes[0].Nodes.Add("Great Grandchild");`

TreeView. Regasire articol.

```
void treeView1_NodeMouseClicked(object sender,  
    TreeNodeMouseClickEventArgs e)  
{  
    textBox1.Text = e.Node.Text;  
}
```



DataGridView

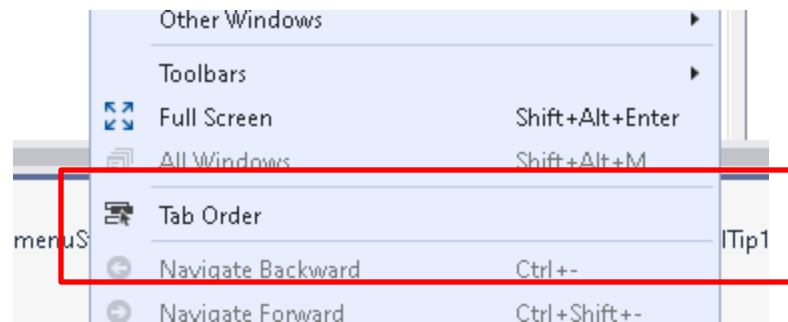
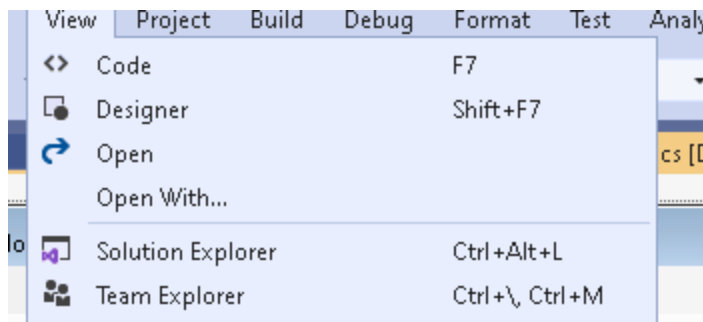
- Colectie de articole, etc.

?

- Intrebari
- Trucuri de programare
- Componente
 - Backgroundworker
 - EventLog

? Ordine navigare controale

- Implicit: ordinea de plasare control in fereastra.
- Explicit: Optiunea "Tab Order" din menu "View".



In loc de incheiere

- Preluare numere naturale.

```
private void numeric(object sender,  
    KeyPressEventArgs e)  
{  
    char c = e.KeyChar;  
    e.Handled = !(Char.IsDigit(c) ||  
                  Char.IsControl(c));  
    this.NextControl(sender, e);  
}
```

Navigare cu tasta “Enter”

```
private void NextControl(object sender,  
                        KeyPressEventArgs e)  
{  
    //if (e.KeyCode = Keys.Enter)  
    if (e.KeyChar == 13)  
        SendKeys.Send("{TAB}");  
}
```

OnKeyDown

```
protected override void OnKeyDown(KeyEventArgs e)
{
    switch (e.KeyCode)
    {
        case Keys.Insert:
            e.Handled = true;
            btn_Adaug.PerformClick(); // Mesaj clic pe butonul cu Name=btn_Adaug
            break;
        case Keys.F2:
            e.Handled = true;
            // Cod ce trebuie executat
            break;
        // ...
        default:
            break;
    }
    base.OnKeyDown(e);
}
```

