**UNIVERSITATEA DE STAT DIN MOLDOVA**

**FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ**

**DEPARTAMENTUL DE INFORMATICĂ**

**POSIBILITĂȚI OFERITE DE**

**CONTROLUL DATAGRIDVIEW**

Referat

Verificat: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **Arnăut Vsevolod, profesor universitar**

Executor: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **Țurcanu Cristian, grupa MIA2201**

CUPRINS

[1 Proprietatile clasei DataGridView 3](#_Toc150429192)

[1.1 CreateParams 3](#_Toc150429193)

[1.2 CurrentCell 3](#_Toc150429194)

[1.3 CurrentCellAddress 3](#_Toc150429195)

[1.4 CurrentRow 4](#_Toc150429196)

[1.5 Cursor 4](#_Toc150429197)

[1.6 DataBindings 4](#_Toc150429198)

[1.7 DataContext 4](#_Toc150429199)

[1.8 DataMember 5](#_Toc150429200)

[2 Metodele clasei DataGridView 6](#_Toc150429201)

[2.1 OnBackgroundImageLayoutChanged(EventArgs) 6](#_Toc150429202)

[2.2 OnBindingContextChanged(EventArgs) 6](#_Toc150429203)

[2.3 OnBorderStyleChanged(EventArgs) 6](#_Toc150429204)

[2.4 OnCancelRowEdit(QuestionEventArgs) 6](#_Toc150429205)

[2.5 OnCausesValidationChanged(EventArgs) 7](#_Toc150429206)

[2.6 OnCellBeginEdit(DataGridViewCellCancelEventArgs) 7](#_Toc150429207)

[2.7 OnCellBorderStyleChanged(EventArgs) 7](#_Toc150429208)

[2.8 OnCellClick(DataGridViewCellEventArgs) 8](#_Toc150429209)

[2.9 OnCellContentClick(DataGridViewCellEventArgs) 8](#_Toc150429210)

[2.10 OnCellContentDoubleClick(DataGridViewCellEventArgs) 8](#_Toc150429211)

# 1 Proprietatile clasei DataGridView

## 1.1 CreateParams

Această proprietate conține parametrii necesari pentru crearea unui control. Ea nu se folosește direct, în locul acestuia folosindu-se proprietățile interne a clasei CreateParams, precum Width, Height și Style. Însă aceasta ar putea fi suprascrisă în cazul creării unor controale cu un set mai limitat de opțiuni, precum un buton care folosește un Icon în loc de un Image.

protected override CreateParams CreateParams  
{  
 get  
 {  
 // Extend the CreateParams property of the Button class.  
 CreateParams cp = base.CreateParams;  
 // Update the button Style.  
 cp.Style |= 0x00000040; // BS\_ICON value  
  
 return cp;  
 }  
}

## 1.2 CurrentCell

Această proprietate conține celula selectată de utilizator, și poate fi atât citită, cât și setată în cod:

// Citim celula selectată  
DataGridViewCell currentCell = dataGridView1.CurrentCell;  
  
// Setăm celula selectată  
dataGridView1.CurrentCell = dataGridView1[1, 2]; // column 1, row 2

## 1.3 CurrentCellAddress

Este de tip Point și conține adresa pe rânduri și coloane a celulei selectate. Poate fi utilă în cazul în care avem nevoie să obținem ceva date conexe legate de celula selectată, însă care nu se conțin în propriu-zis celula dată.

Ca și exemplu, aceasta ar putea fi folosită pentru recolorarea rândului în cazul în care utilizatorul selectează o altă celulă:

void dataGridView1\_CurrentCellChanged(object sender, EventArgs e)  
{  
 if (oldRowIndex != -1)  
 {  
 this.dataGridView1.InvalidateRow(oldRowIndex);  
 }  
 oldRowIndex = this.dataGridView1.CurrentCellAddress.Y;  
}

## 1.4 CurrentRow

Proprietate de tip DataGridViewRow care conține rândul celulei selectate. Poate fi utilă pentru a manipula date din rândul selectat sau a modifica și chiar șterge întreg rândul.

## 1.5 Cursor

Proprietate de tip Cursor care permite modificarea cursorului când acesta trece peste controlul dat. Spre exemplu, poate fi temporar schimbat într-un cursor de încărcare în cazul în care programul efectuează un proces de durată.

// Schimbăm cursorul în unul de așteptare în timpul unei tranzacții  
dataGridView1.Cursor = Cursors.WaitCursor;  
  
// Schimbăm cursorul când tranzacția s-a terminat  
dataGridView1.Cursor = Cursors.Default;

## 1.6 DataBindings

Proprietate de tip System.Windows.Forms.ControlBindingsCollection care conține toate Bindingurile unui control. Aceasta poate fi folosită pentru a adăuga bindinguri unui control, astfel legând ceva logică de conținutul unui control.

// Creăm un nou binding  
Binding binding = new Binding("Text", myDataSource, "MyProperty");  
  
// Adăugăm bindingul unui control  
myControl.DataBindings.Add(binding);

## 1.7 DataContext

DataContext-ul este o proprietate “ambient”, adică copii unui control primesc contextul de la părinte. Acesta se folosește pentru data binding în WPF, iar în Windows Forms în locul său se folosește DataBindings.

## 1.8 DataMember

Setează lista sau tabelul pentru care DataGridView-ul arată datele. Util în cazul în care se citesc sau editează în același timp date din mai multe tabele.

private void BindData()  
{  
 customersDataGridView.AutoGenerateColumns = true;  
 customersDataGridView.DataSource = customersDataSet;  
 customersDataGridView.DataMember = "Customers"; // Setăm tabelul Customers să fie arătat primul  
}

# 2 Metodele clasei DataGridView

Toate metodele următoare au rolul de a chema un event anumit, care este legat de o acțiune specifică, iar metodele pot fi utilizate atât pentru a chema acest event, cât și pentru a fi suprascris în cazul unor controale customizate, adăugând logică proprie.

## 2.1 OnBackgroundImageLayoutChanged(EventArgs)

Cheamă event-ul BackgroundImageLayoutChanged, care este chemată atunci când se schimbă proprietatea BackgroundImageLayout. Poate fi suprascris pentru a adăuga logică schimbării fundalului pentru un control customizat.

protected override void OnBackgroundImageLayoutChanged(EventArgs e)  
{  
 // Chemăm metoda de bază  
 base.OnBackgroundImageLayoutChanged(e);  
  
 // Adăugăm logica nouă  
 Console.WriteLine("Background image layout changed");  
}

## 2.2 OnBindingContextChanged(EventArgs)

Cheamă eventul BindingContextChanged, care este folosit atunci când BindingContext-ul unui control este schimbat. De obicei, acesta este chemat doar la crearea controlului, și de obicei nu trebuie folosit.

## 2.3 OnBorderStyleChanged(EventArgs)

Cheamă eventul BorderStyleChanged, folosit atunci când se schimbă valoarea proprietății BorderStyle. Iarăși, eventul este actual doar la inițializarea obiectului, așa că poate fi util doar în cazul debugging-ului unor proprietăți interne, și de obicei nu trebuie folosit.

## 2.4 OnCancelRowEdit(QuestionEventArgs)

Cheamă eventul CancelRowEdit, folosit atunci când editarea unui rând este anulată. Util pentru resetarea ultimelor schimbări.

private void dataGridView1\_CancelRowEdit(object sender,  
 System.Windows.Forms.QuestionEventArgs e)  
{  
 if (this.rowInEdit == this.dataGridView1.Rows.Count - 2 &&  
 this.rowInEdit == this.customers.Count)  
 {  
 // If the user has canceled the edit of a newly created row,   
 // replace the corresponding Customer object with a new, empty one.  
 this.customerInEdit = new Customer();  
 }  
 else  
 {  
 // If the user has canceled the edit of an existing row,   
 // release the corresponding Customer object.  
 this.customerInEdit = null;  
 this.rowInEdit = -1;  
 }  
}

## 2.5 OnCausesValidationChanged(EventArgs)

Cheamă eventul CausesValidationChanged, folosit atunci când se schimbă valoarea proprietății CausesValidation. Iarăși, chemat doar la inițializare, și astfel util doar pentru debugging.

## 2.6 OnCellBeginEdit(DataGridViewCellCancelEventArgs)

Cheamă eventul CellBeginEdit, care este chemat atunci când o celulă începe a fi modificată. Util când e necesar de accentuat faptul că celula e modificată:

private void dataGridView1\_CellBeginEdit(object sender,  
 DataGridViewCellCancelEventArgs e)  
{  
 string msg = String.Format("Editing Cell at ({0}, {1})",  
 e.ColumnIndex, e.RowIndex);  
 this.Text = msg;  
}  
  
private void dataGridView1\_CellEndEdit(object sender,  
 DataGridViewCellEventArgs e)  
{  
 string msg = String.Format("Finished Editing Cell at ({0}, {1})",  
 e.ColumnIndex, e.RowIndex);  
 this.Text = msg;  
}

## 2.7 OnCellBorderStyleChanged(EventArgs)

Total analog eventului OnBorderStyleChanged, cu singura diferență că cheamă eventul CellBorderStyleChanged. Altfel este total identic.

## 2.8 OnCellClick(DataGridViewCellEventArgs)

Cheamă eventul CellClick care este chemat atunci când o celulă este apăsată. În afara utilității evidente de detectare a clickurilor asupra unei celule, acesta poate fi folosit pentru a simula un joc de “крестики-нолики”:

private void dataGridView1\_CellClick(object sender,  
 DataGridViewCellEventArgs e)  
{  
  
 if (turn.Text.Equals(gameOverString)) { return; }  
  
 DataGridViewImageCell cell = (DataGridViewImageCell)  
 dataGridView1.Rows[e.RowIndex].Cells[e.ColumnIndex];  
  
 if (cell.Value == blank)  
 {  
 if (IsOsTurn())  
 {  
 cell.Value = o;  
 }  
 else  
 {  
 cell.Value = x;  
 }  
 ToggleTurn();  
 }  
 if (IsAWin())  
 {  
 turn.Text = gameOverString;  
 }  
}

## 2.9 OnCellContentClick(DataGridViewCellEventArgs)

Cheamă eventul CellContentClick, care folosit atunci cânddd conținutul unei celule este tastat. În afară de cursor, acest event poate fi chemat tastând space atunci când o celulă este selectată.

## 2.10 OnCellContentDoubleClick(DataGridViewCellEventArgs)

Total analog metodei OnCellContentClick și eventului lui asociat, cu singura diferență că eventul este chemat la dublu clic.