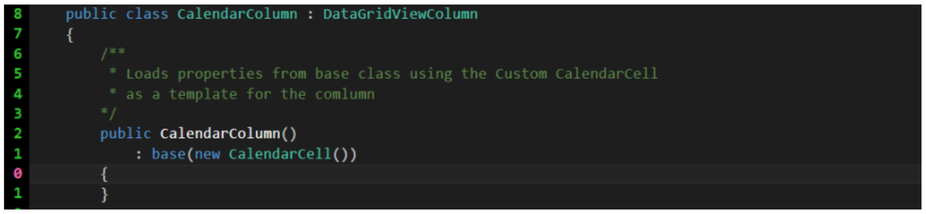
Calendar Column

The calendar column is an extension of the DatagridViewColumn that utilyzes three classes inheritting from the DatagridViewColumn, DataGridviewTextboxCell, and the DateTimePicker as well as incorporating the IDataGridViewEditingControl interface as a means of embedding the DateTimePicker class in the form of the CalendarEditingControl.

The Calendar Column

The Constructor

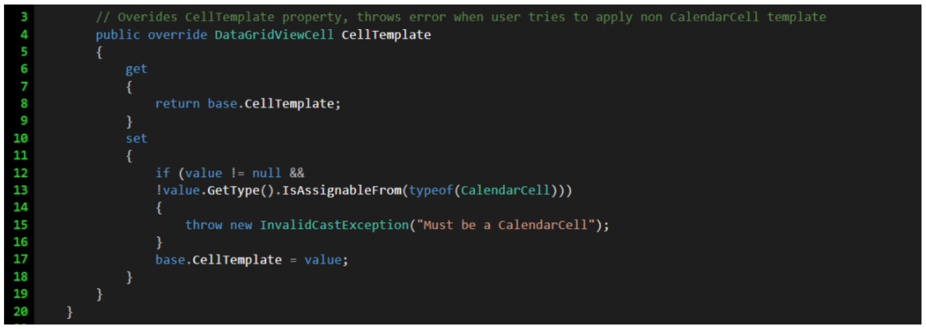


```
public class CalendarColumn : DataGridViewColumn
{
    /**
     * Loads properties from base class using the Custom CalendarCell
     * as a template for the comlumn
    */
    public CalendarColumn()
        : base(new CalendarCell())
    {
    }
```

Calendar Column

The CalendarColumn class inherits from the DatagridViewColumn. This control will ultimately be available under the DatagridView control using the add a column feature. The class can also be hard coded in later. The public CalendarColumn is the entry point for the class, the colon base(new CalendarCcell()) runs the base class constructor (DatagridViewColumn) and then adds the CalendarCell() class as the cell template (we will build the CalendarCell later).

CellTemplate Override



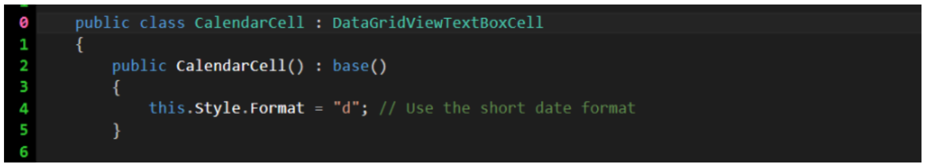
```
        // Overides CellTemplate property, throws error when user tries to apply non CalendarCell template
        public override DataGridViewCell CellTemplate
        {
            get
            {
                return base.CellTemplate;
            }
            set
            {
                if (value != null &&
                !value.GetType().IsAssignableFrom(typeof(CalendarCell)))
                {
                    throw new InvalidCastException("Must be a CalendarCell");
                }
                base.CellTemplate = value;
            }
        }
    }
```

Calendar Column CellTemplate Overide

Overides the CellTemplate property on the DatagridViewColumn. Causes any other CellTemplate setting other than the CalendarCell to throw an error.

The Calendar Cell

The Constructor

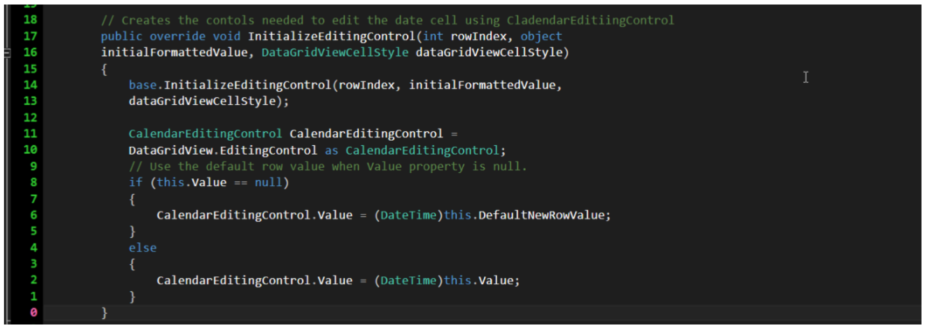


```
public class CalendarCell : DataGridViewTextBoxCell
{
    public CalendarCell() : base()
    {
        this.Style.Format = "d"; // Use the short date format
    }
```

Calendar Cell

The CalendarCell class inherits from the DatagridViewTextrBoxCell. Like the CalendarColumn, the CalendarCell constructor calls the base constructor as well. Then the Format for the custom cell is set to short date.

InitializeEditingControl Override

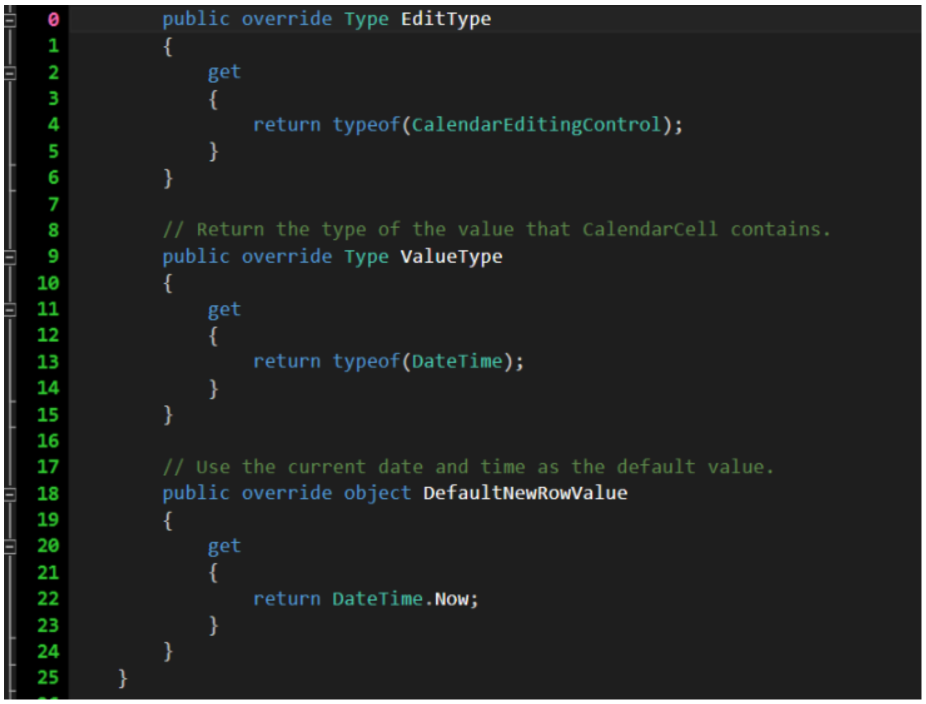


```
// Creates the contols needed to edit the date cell using CladendarEditiingControl
public override void InitializeEditingControl(int rowIndex, object
initialFormattedValue, DataGridViewCellStyle dataGridViewCellStyle)
{
    base.InitializeEditingControl(rowIndex, initialFormattedValue,
    dataGridViewCellStyle);

    CalendarEditingControl CalendarEditingControl =
    DataGridView.EditingControl as CalendarEditingControl;
    // Use the default row value when Value property is null.
    if (this.Value == null)
    {
        CalendarEditingControl.Value = (DateTime)this.DefaultNewRowValue;
    }
    else
    {
        CalendarEditingControl.Value = (DateTime)this.Value;
    }
}
```

Initialize Editing Control

When editing is initiated cell information is sent to the editing control. If the cell value is null, the CalendarEditingControl default value is set to today.

Additional Properties



```
        public override Type EditType
        {
            get
            {
                return typeof(CalendarEditingControl);
            }
        }

        // Return the type of the value that CalendarCell contains.
        public override Type ValueType
        {
            get
            {
                return typeof(DateTime);
            }
        }

        // Use the current date and time as the default value.
        public override object DefaultNewRowValue
        {
            get
            {
                return DateTime.Now;
            }
        }
    }
```

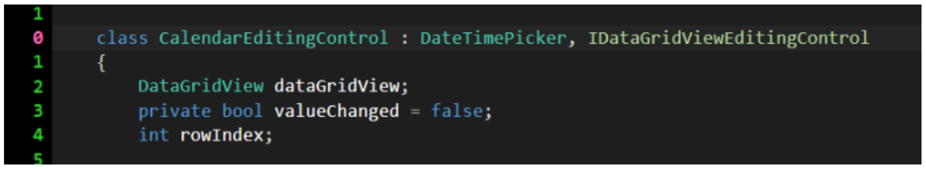
Additional Properties

These properites override basic information about the class.

The CalendarEditingControl

Will be called when the user starts to edit a cell in a calendar column. Inherits from the DateTimePicker and uses the IDatagridViewEditingControl inorder to apply DateTimePicker edits to the cell.

Properties

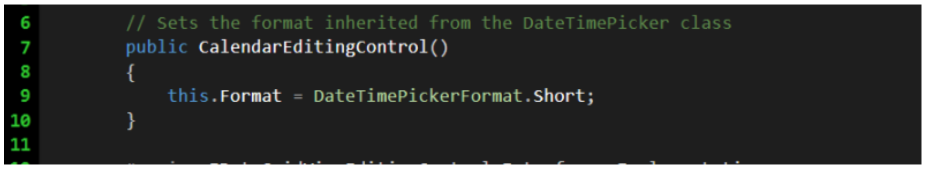


```
    class CalendarEditingControl : DateTimePicker, IDataGridViewEditingControl
    {
        DataGridView dataGridView;
        private bool valueChanged = false;
        int rowIndex;
```

Calendar Editing Control Properties

Properties needed for class

The Constructor



```
        // Sets the format inherited from the DateTimePicker class
        public CalendarEditingControl()
        {
            this.Format = DateTimePickerFormat.Short;
        }
```

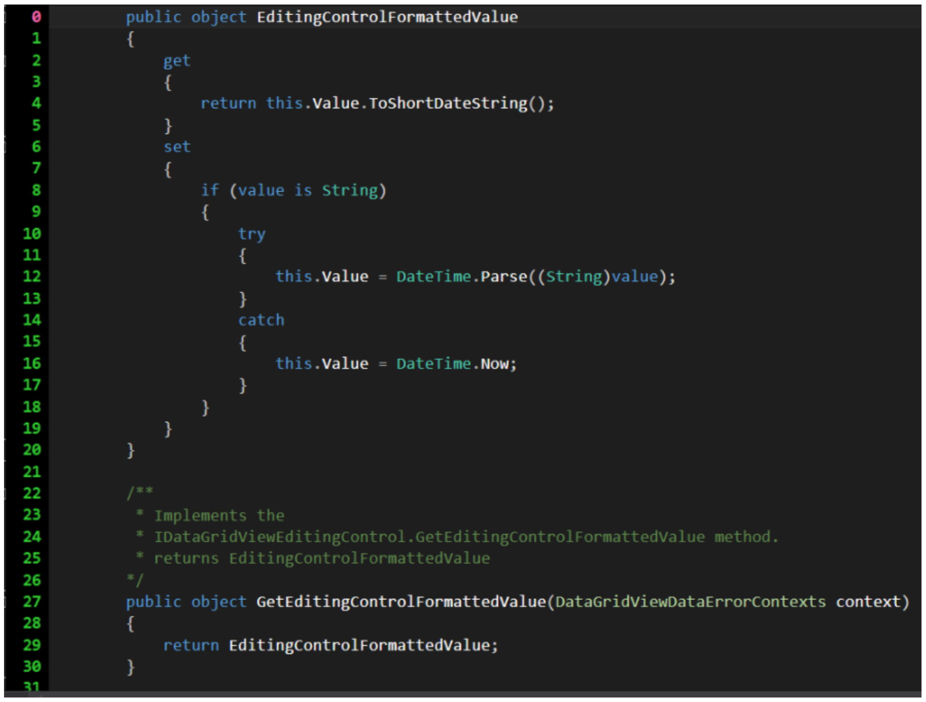
Calendar Editing Control Constructor

Sets the editing format to Short Date

Interface Implemenatons

The following sections of code are the required implementations for the IDatagridViewEditingControl. These are the parts of the code that need to be defined so that when the DatagridView that parents our CalendarColumn is able to apply the user's edits.

Editing Control Formatted Value

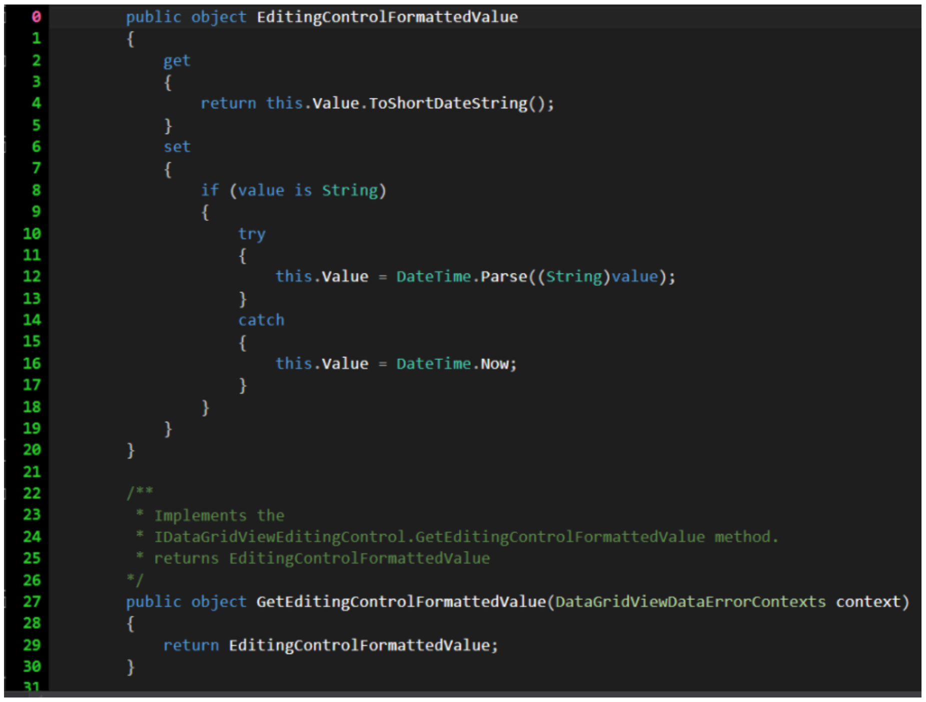


```
        public object EditingControlFormattedValue
        {
            get
            {
                return this.Value.ToShortDateString();
            }
            set
            {
                if (value is String)
                {
                    try
                    {
                        this.Value = DateTime.Parse((String)value);
                    }
                    catch
                    {
                        this.Value = DateTime.Now;
                    }
                }
            }
        }

        /**
         * Implements the
         * IDataGridViewEditingControl.GetEditingControlFormattedValue method.
         * returns EditingControlFormattedValue
        */
        public object GetEditingControlFormattedValue(DataGridViewDataErrorContexts context)
        {
            return EditingControlFormattedValue;
        }
```

Editing Control Formatted Value

Handles getting and setting the value of the Editing Control, returns the value as a short date, as well as attempts to store it as a short date. If the value cannot be stored as a short date, the value is set to todays date. GetEditingControlFormattedValue handles retrieving the formatted value, and also has it's own set of error handling.

Apply Cell Styling to Editing Control

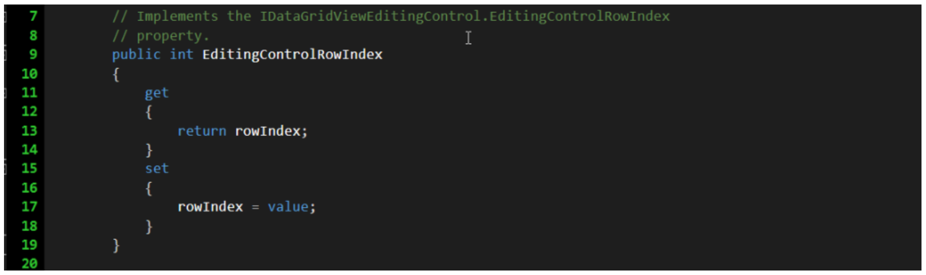


```
        public object EditingControlFormattedValue
        {
            get
            {
                return this.Value.ToShortDateString();
            }
            set
            {
                if (value is String)
                {
                    try
                    {
                        this.Value = DateTime.Parse((String)value);
                    }
                    catch
                    {
                        this.Value = DateTime.Now;
                    }
                }
            }
        }

        /**
         * Implements the
         * IDataGridViewEditingControl.GetEditingControlFormattedValue method.
         * returns EditingControlFormattedValue
        */
        public object GetEditingControlFormattedValue(DataGridViewDataErrorContexts context)
        {
            return EditingControlFormattedValue;
        }
```

Apply Cell Styling to Editing Control

Sets the look of the CalendarEditingControl to match the parent DatagridView

Editing Control Row Index

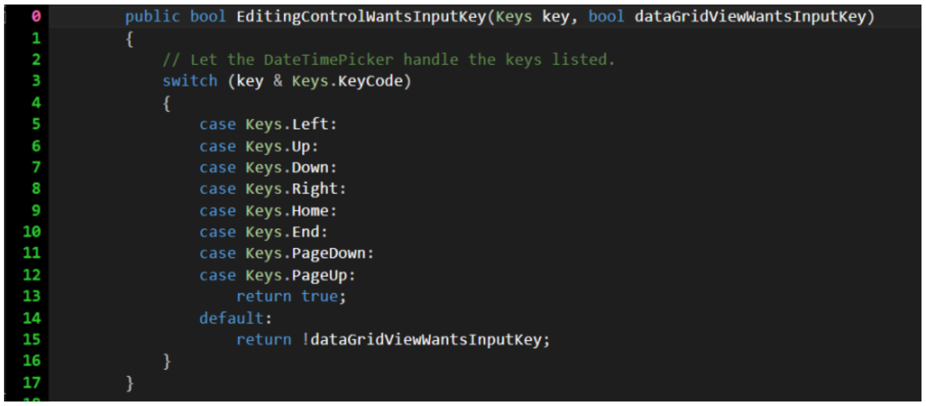


```
        // Implements the IDataGridViewEditingControl.EditingControlRowIndex
        // property.
        public int EditingControlRowIndex
        {
            get
            {
                return rowIndex;
            }
            set
            {
                rowIndex = value;
            }
        }
```

Editing Control Row Index

Property to store the cell index that is being editted.

Editing Control Wants Input Key



```
        public bool EditingControlWantsInputKey(Keys key, bool dataGridViewWantsInputKey)
        {
            // Let the DateTimePicker handle the keys listed.
            switch (key & Keys.KeyCode)
            {
                case Keys.Left:
                case Keys.Up:
                case Keys.Down:
                case Keys.Right:
                case Keys.Home:
                case Keys.End:
                case Keys.PageDown:
                case Keys.PageUp:
                    return true;
                default:
                    return !dataGridViewWantsInputKey;
            }
        }
```

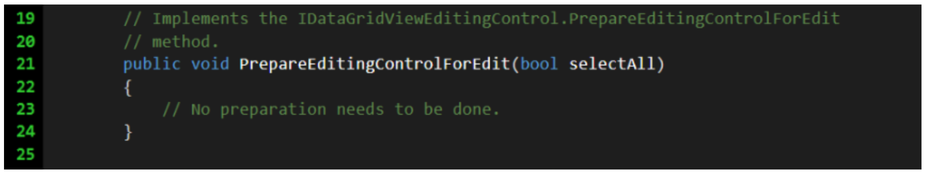
Editing Control Wants Input Key

Defines how to handle the editing control needing a key input.

Prepare Editing Control for Edit

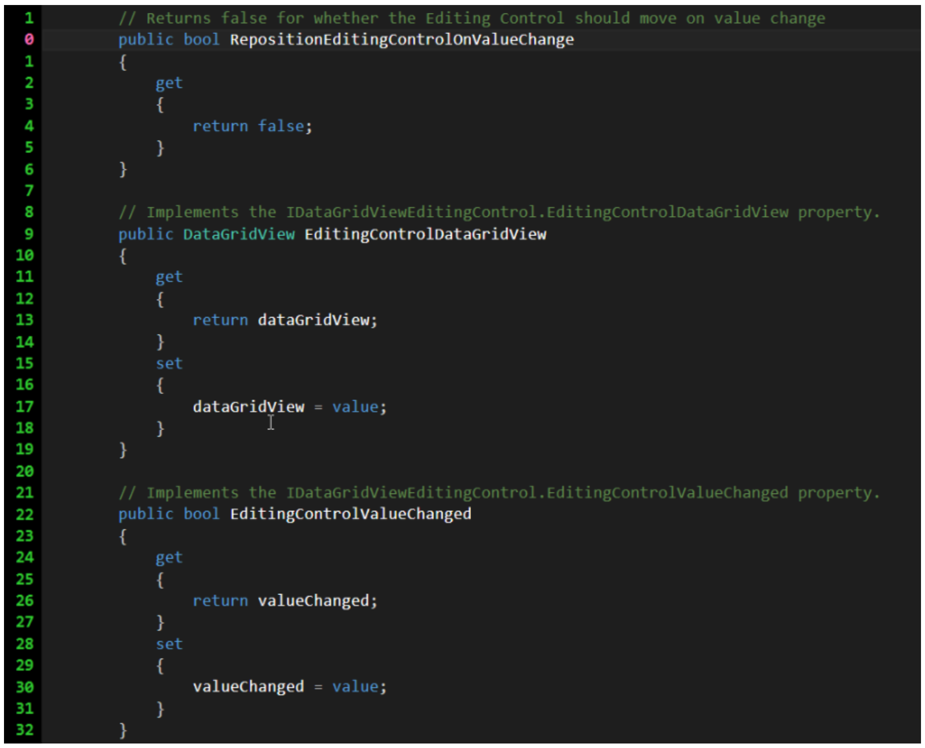
No preparation is needed prior to edit

Misc. Implemenations



```
        // Implements the IDataGridViewEditingControl.PrepareEditingControlForEdit
        // method.
        public void PrepareEditingControlForEdit(bool selectAll)
        {
            // No preparation needs to be done.
        }
```

Prepare Editing Control for Edit

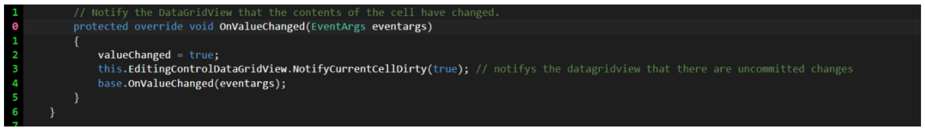


```
        // Returns false for whether the Editing Control should move on value change
        public bool RepositionEditingControlOnValueChange
        {
            get
            {
                return false;
            }
        }

        // Implements the IDataGridViewEditingControl.EditingControlDataGridView property.
        public DataGridView EditingControlDataGridView
        {
            get
            {
                return dataGridView;
            }
            set
            {
                dataGridView = value;
            }
        }

        // Implements the IDataGridViewEditingControl.EditingControlValueChanged property.
        public bool EditingControlValueChanged
        {
            get
            {
                return valueChanged;
            }
            set
            {
                valueChanged = value;
            }
        }
```

Repositon Editing Control on Value Change

PrepareEditingControlforEdit does not need set as there is no preparation needed before editing can begin. RepositionEditingControlOnValueChange is set to return false so the control won't reposition when the value changes. We have the EditingControlDatagridView set to return our class' dataGridView property. EditingControlValueChanged boolean is set to return to return the our class' valueChanged boolean. Finally the EditingPanelCursor is set to return the default DateTimePicker cursor.

On Value Changed



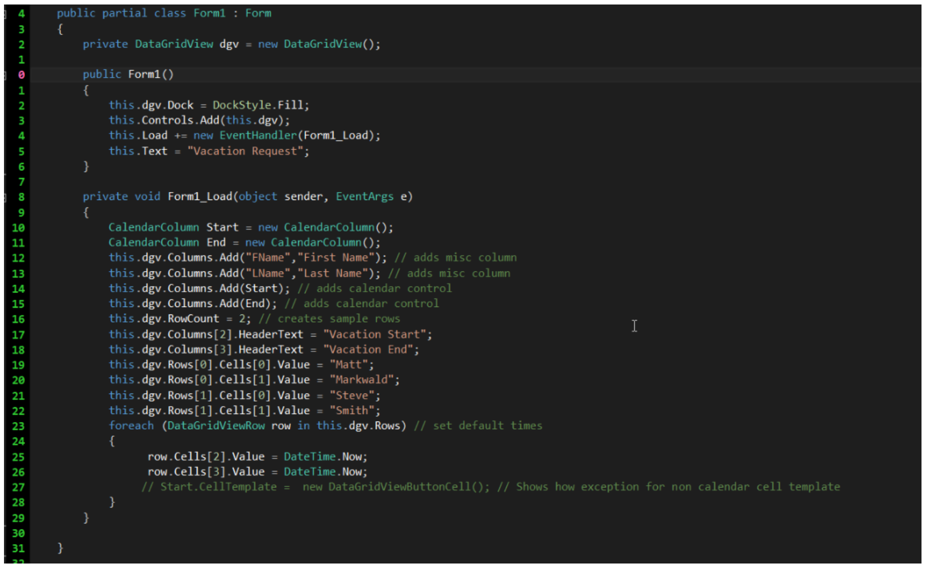
```
        // Notify the DataGridView that the contents of the cell have changed.
        protected override void OnValueChanged(EventArgs eventargs)
        {
            valueChanged = true;
            this.EditingControlDataGridView.NotifyCurrentCellDirty(true); // notifys the datagridview that there are uncommitted changes
            base.OnValueChanged(eventargs);
        }
    }
```

On Value Changed

The last bit of code we need is an override for our base class void, OnValueChanged. This handles when the value changes by setting our valueChanged boolean to true, then notify the parent DatagridView, before calling the base OnValueChanged sending the eventargs.

Demo

Go to form1 constructor and add the following code.



```
public partial class Form1 : Form
{
    private DataGridView dgv = new DataGridView();

    public Form1()
    {
        this.dgv.Dock = DockStyle.Fill;
        this.Controls.Add(this.dgv);
        this.Load += new EventHandler(Form1_Load);
        this.Text = "Vacation Request";
    }

    private void Form1_Load(object sender, EventArgs e)
    {
        CalendarColumn Start = new CalendarColumn();
        CalendarColumn End = new CalendarColumn();
        this.dgv.Columns.Add("FName","First Name"); // adds misc column
        this.dgv.Columns.Add("LName","Last Name"); // adds misc column
        this.dgv.Columns.Add(Start); // adds calendar control
        this.dgv.Columns.Add(End); // adds calendar control
        this.dgv.RowCount = 2; // creates sample rows
        this.dgv.Columns[2].HeaderText = "Vacation Start";
        this.dgv.Columns[3].HeaderText = "Vacation End";
        this.dgv.Rows[0].Cells[0].Value = "Matt";
        this.dgv.Rows[0].Cells[1].Value = "Markwald";
        this.dgv.Rows[1].Cells[0].Value = "Steve";
        this.dgv.Rows[1].Cells[1].Value = "Smith";
        foreach (DataGridViewRow row in this.dgv.Rows) // set default times
        {
              row.Cells[2].Value = DateTime.Now;
              row.Cells[3].Value = DateTime.Now;
             // Start.CellTemplate =  new DataGridViewButtonCell(); // Shows how exception for non calendar cell template
        }
    }

}
```

Demo

Run the program. Notice what happens when you click on the Vacation Start and Vacation End cells.