Add Buttons to DataGrid Example

Embed buttons inside DataGrid cells.

How-to Examples Feature Operator Buttons to DataGrid

Add Buttons to DataGrid Rows: The integration of interactive buttons within DataGrid rows is a valuable aspect of the platform, enhancing user interface design and functionality. This process allows for a dynamic and intuitive navigation system within an application, with each button acting as a gateway to different displays.

- Integrating Buttons to a DataGrid
- Step-by-Step

Integrating Buttons to a DataGrid

This process involves integrating buttons within a DataGrid, an effective tool for displaying, editing, and sorting tabular data. A custom column is created for these buttons, with each row housing a unique button.

In our example, we illustrate a DataGridWindow with a column dedicated to buttons. This setup enables each row of the DataGrid to have a button that, when clicked, redirects the user to a different display.

The steps include creating a Button-type column, instantiating Button objects, assigning click events to these buttons, and adding text to the buttons. Finally, the instantiated buttons are assigned to the record's Button-type column.

One crucial aspect of this setup is ensuring that any click events associated with the buttons are appropriately removed when the display is closed. This action ensures resources are efficiently managed, preventing potential memory leaks.

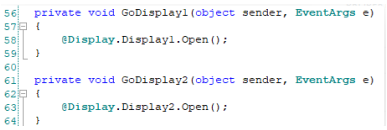
Step-by-Step

To create buttons in DataGrids that direct to other displays, follow these steps:

1. **Create Button Column**: Add a column of type Button to the DataGrid (Line 12).
2. **Instantiate Buttons**: Create button objects (Lines 2 and 3).
3. **Add Click Events**: Assign click event handlers to the buttons (Lines 14 and 24).
4. **Set Button Text**: Define the text for each button (Lines 15 and 25).
5. **Assign Buttons to Column**: Add the buttons to the Button-type column in the DataGrid (Lines 21 and 31).



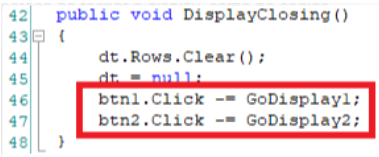
6. The click events are defined in the methods below:



```
private void GoDisplay1(object
sender, EventArgs e)
{
    @Display.Display1.Open();
}

private void GoDisplay2(object
sender, EventArgs e)
{
    @Display.Display2.Open();
}
```

7. You should also define the removal of the click events added to the buttons when closing the display, as in the following code:



```
public void DisplayClosing()
{
    dt.Rows.Clear();
    dt = null;

    btn1.Click -= GoDisplay1;
    btn2.Click -= GoDisplay2;
}
```

```
DataTable dt;
Button btn1 = new Button();
Button btn2 = new Button();

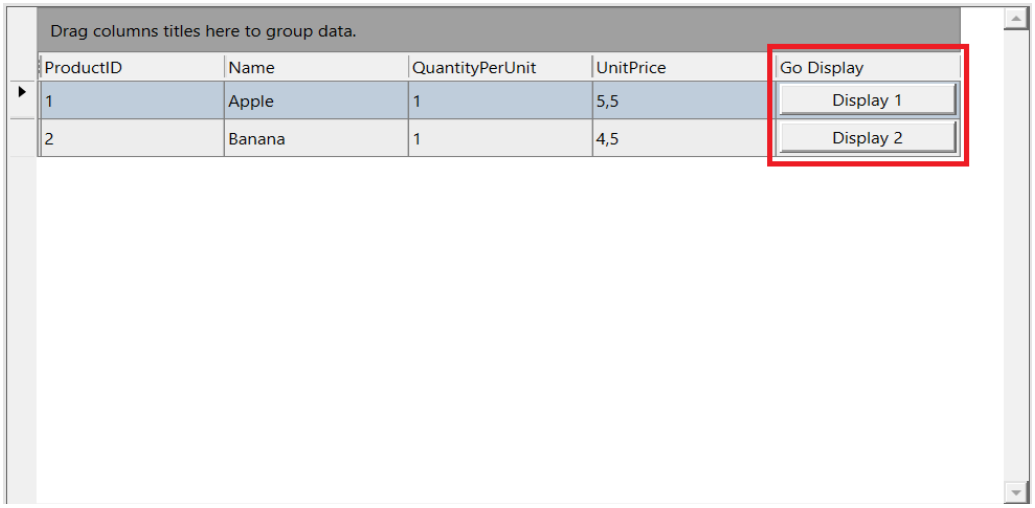
public void DisplayOpening()
{
    dt = new DataTable
("Products");
    dt.Columns.Add("ProductID",
typeof(int));
    dt.Columns.Add("Name",
typeof(string));
    dt.Columns.Add
("QuantityPerUnit", typeof
(string));
    dt.Columns.Add("UnitPrice",
typeof(double));
    dt.Columns.Add("GoDisplay",
typeof(Button));

    btn1.Click += new
RoutedEventHandler(GoDisplay1);
    btn1.Content = "Display 1";
    DataRow row1 = dt.NewRow();
    row1["ProductID"] = 1;
    row1["Name"] = "Apple";
    row1["QuantityPerUnit"] = 1;
    row1["UnitPrice"] = 5.50;
    row1["GoDisplay"] = btn1;
    dt.Rows.Add(row1);

    btn2.Click += new
RoutedEventHandler(GoDisplay2);
    btn2.Content = "Display 2";
    DataRow row2 = dt.NewRow();
    row2["ProductID"] = 2;
    row2["Name"] = "Banana";
    row2["QuantityPerUnit"] = 1;
    row2["UnitPrice"] = 4.50;
    row2["GoDisplay"] = btn2;
    dt.Rows.Add(row2);

    @Dataset.Table.
DataTable_Grid.LocalContents.
Initialize(dt);
}
```

The generated table contains the last column with buttons that direct to different displays, as shown in the image:



In this section...
