
THE LECTURE 16

IMAGEVIEWER

CREATE A NEW PROJECT

Visual Studio 2019

Open recent

As you use Visual Studio, any projects, folders, or files that you open will show up here for quick access.

You can pin anything that you open frequently so that it's always at the top of the list.

Get started



Clone or check out code

Get code from an online repository like GitHub or Azure DevOps



Open a project or solution

Open a local Visual Studio project or .sln file



Open a local folder

Navigate and edit code within any folder

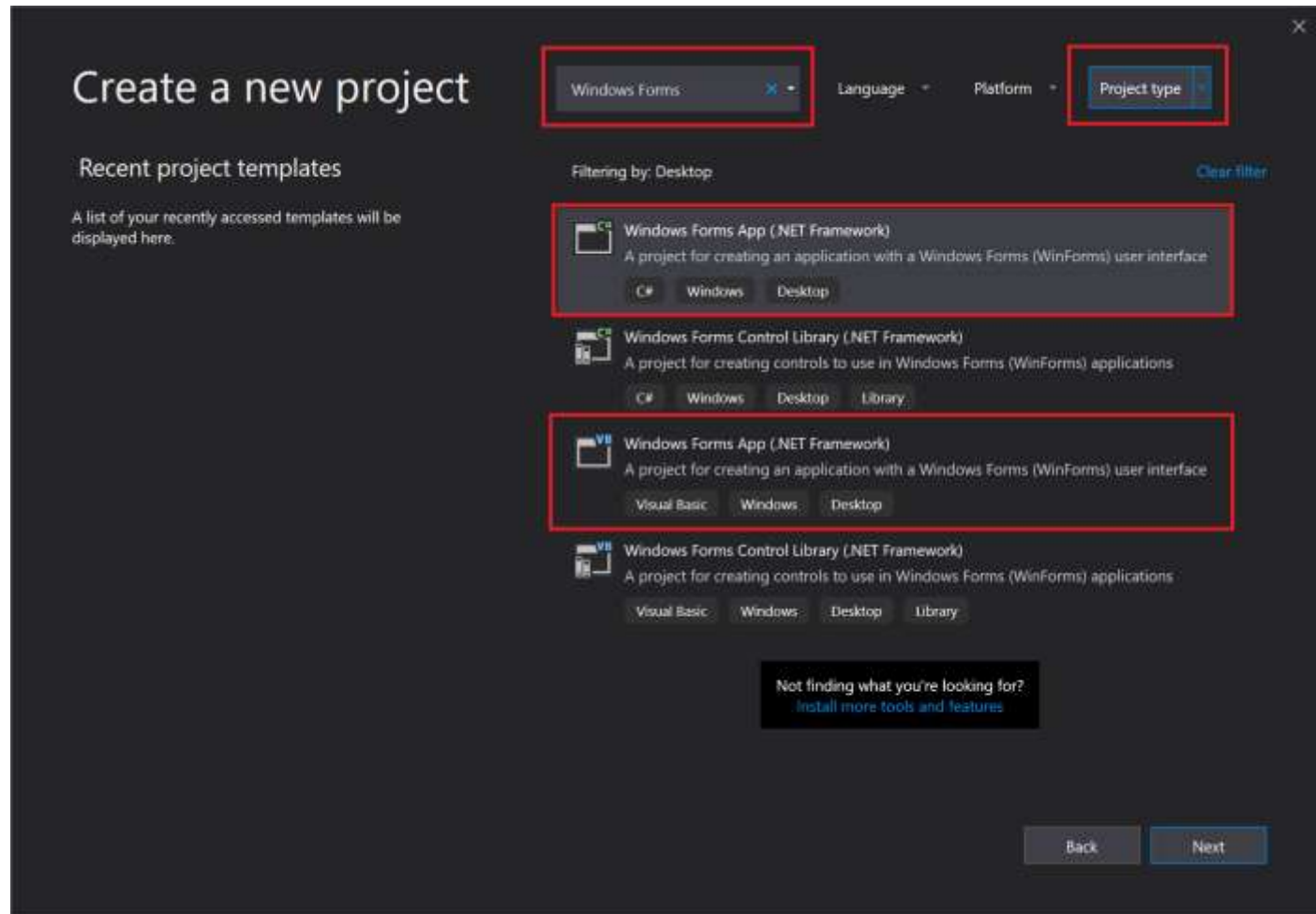


Create a new project

Choose a project template with code scaffolding to get started

[Continue without code →](#)

CREATE A NEW PROJECT



Windows Forms App

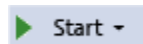
When you create a Windows Forms App project, you actually build a program that runs. In this tutorial, your picture viewer app doesn't do much yet—although it will. For now, it displays an empty window that shows **Form1** in the title bar.

Here's how to run your app.

1. Choose one of the following methods:

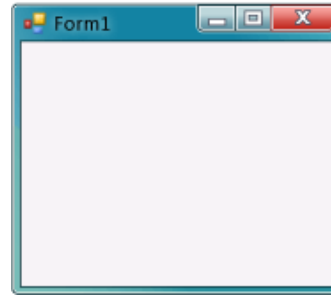
- Choose the **F5** key.
- On the menu bar, choose **Debug > Start Debugging**.
- On the toolbar, choose the **Start Debugging** button, which appears as follows:

Start Debugging toolbar button



Windows Forms App

•Visual Studio runs your app, and a window called **Form1** appears. The following screenshot shows the app you just built. The app is running, and you'll soon add to it.

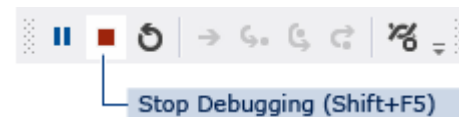


Windows Forms App, running

•Go back to the Visual Studio integrated development environment (IDE), and then look at the new toolbar. Additional buttons appear on the toolbar when you run an application. These buttons let you do things like stop and start your app, and help you track down any errors (bugs) it may have. For this example, we're using it to start and stop the app.

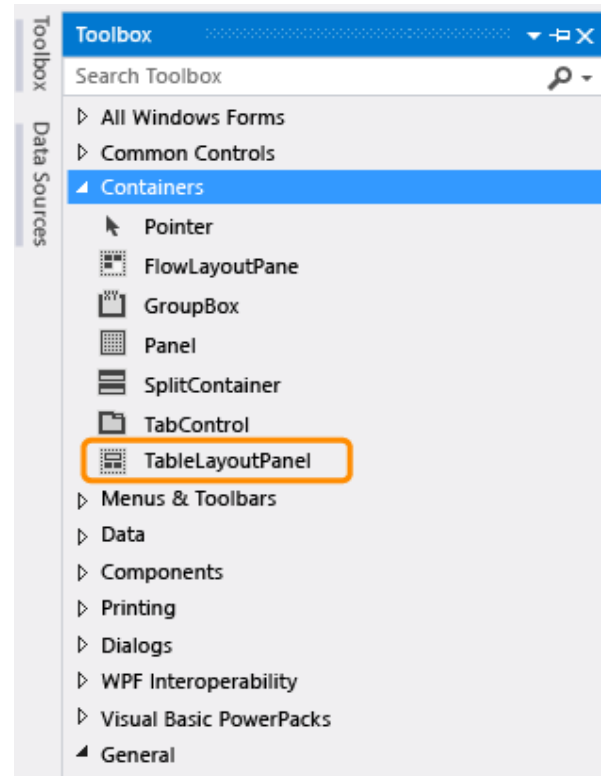
Debugging toolbar

- Use one of the following methods to stop your app:
- On the toolbar, choose the **Stop Debugging** button.
- On the menu bar, choose **Debug > Stop Debugging**.
- Use your keyboard and press **Shift+F5**.
- Choose the **X** button in the upper corner of the **Form1** window.



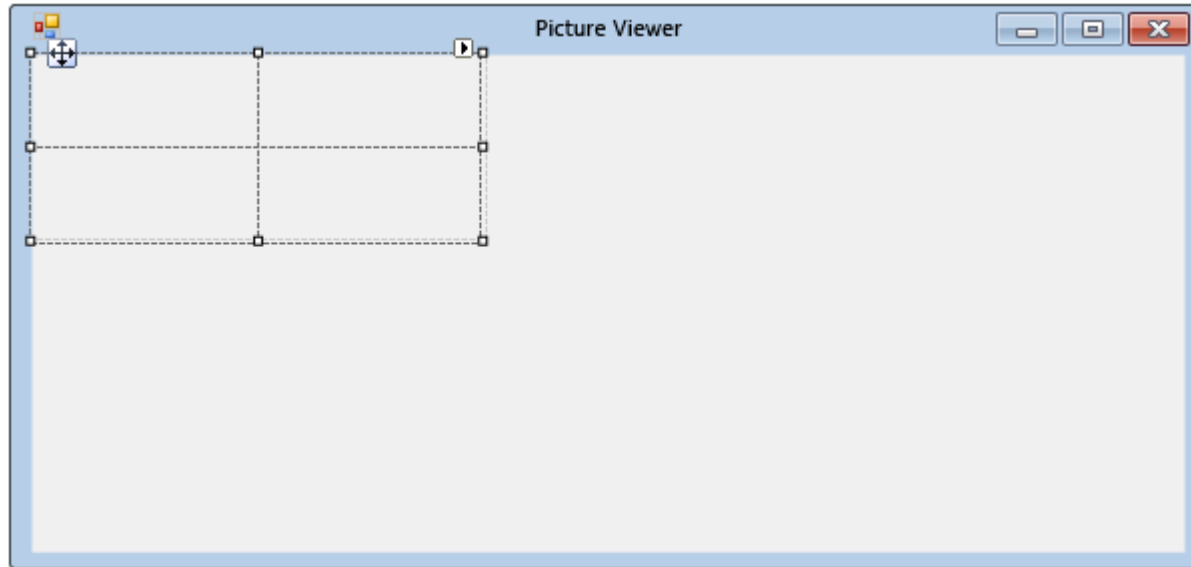
TABLELAYOUTPANEL CONTROL

- On the left side of the Visual Studio IDE, choose the **Toolbox** tab.
(Alternatively, choose **View > Toolbox** from the menu bar, or press **Ctrl+Alt+X**.)
- Choose the small triangle symbol next to the **Containers** group to open it, as shown in the following screenshot.



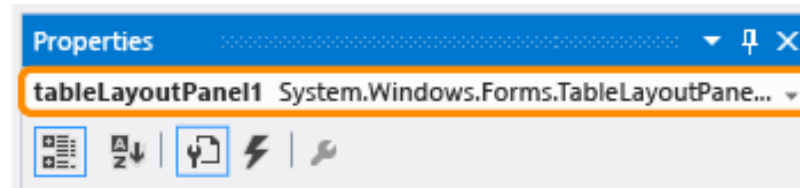
TABLELAYOUTPANEL CONTROL

- You can add controls like buttons, check boxes, and labels to your form. Double-click the TableLayoutPanel control in the **Toolbox**. (Or, you can drag the control from the toolbox onto the form.) When you do this, the IDE adds a TableLayoutPanel control to your form, as shown in the following screenshot.



TABLELAYOUTPANEL CONTROL

- Be sure the `TableLayoutPanel` is selected by choosing it. You can verify what control is selected by looking at the drop-down list at the top of the **Properties** window, as shown in the following screenshot.

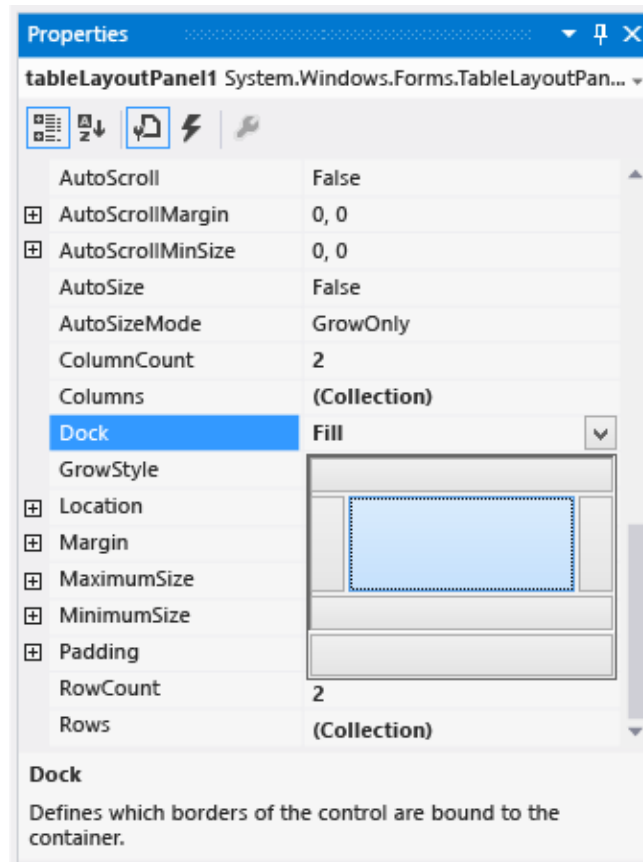


Properties window showing `TableLayoutPanel` control

- Choose the **Alphabetical** button on the toolbar in the **Properties** window. This sorts the list of properties in the **Properties** window in alphabetical order, which makes it easier to locate properties in this tutorial.
- The control selector is a drop-down list at the top of the **Properties** window. In this example, it shows that a control called `tableLayoutPanel1` is selected. You can select controls either by choosing an area in **Windows Forms Designer** or by choosing from the control selector.
- Now that the `TableLayoutPanel` is selected, find the **Dock** property and choose **Dock**, which should be set to **None**. Notice that a drop-down arrow appears next to the value. Choose the arrow, and then select the **Fill** button (the large button in the middle), as shown in the following screenshot.

TABLELAYOUTPANEL CONTROL

- *Docking* in Visual Studio refers to when a window is attached to another window or area in the IDE. For example, the **Properties** window can be undocked—that is, unattached and free-floating within Visual Studio—or it can be docked against **Solution Explorer**.



TABLELAYOUTPANEL CONTROL

- After you set the TableLayoutPanel **Dock** property to **Fill**, notice that the panel fills the entire form. If you resize the form again, the TableLayoutPanel stays docked, and resizes itself to fit.
- Currently, the TableLayoutPanel has two equal-size rows and two equal-size columns. Let's resize them so the top row and right column are both much bigger. In **Windows Forms Designer**, select the TableLayoutPanel. In the upper-right corner, there is a small black triangle button, which appears as follows.



Triangle button

This button indicates that the control has tasks that help you set its properties automatically

TABLELAYOUTPANEL CONTROL

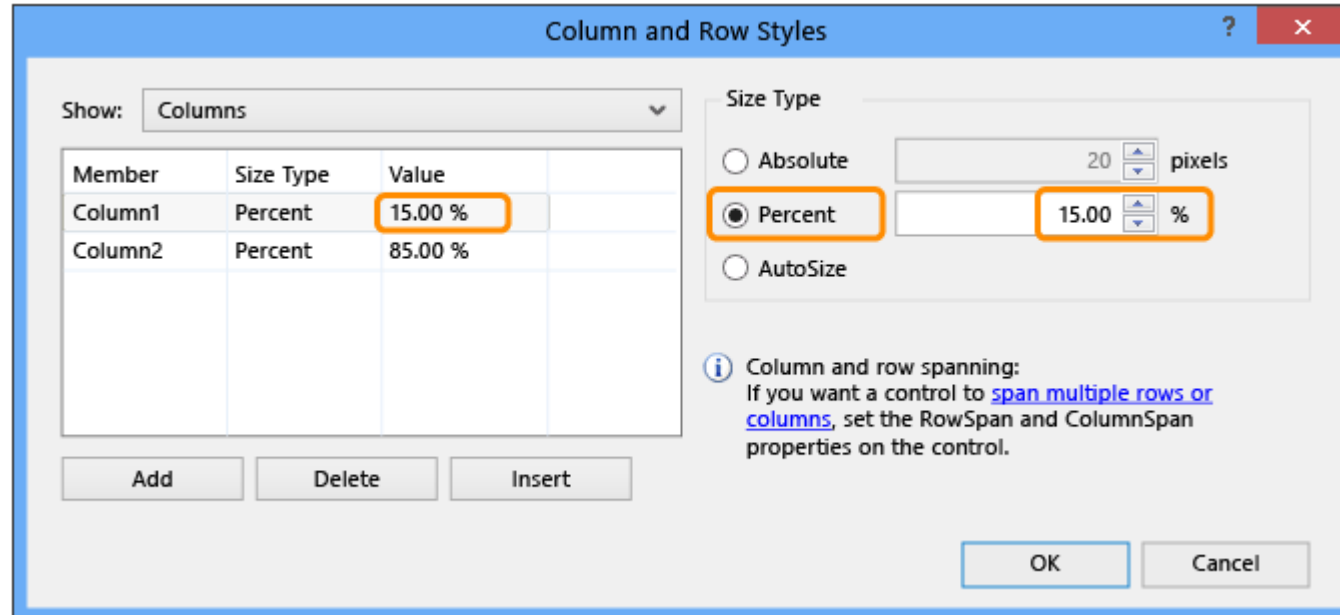
- Choose the triangle to display the control's task list, as shown in the following screenshot.



- TableLayoutPanel tasks
- Choose the Edit Rows and Columns task to display the Column and Row Styles window. Choose Column1, and set its size to 15 percent by being sure the Percent button is selected and entering 15 in the Percent box. (That's a NumericUpDown control, which you'll use in a later tutorial.) Choose Column2 and set it to 85 percent. Don't choose the OK button yet, because the window will close. (But if you do, you can reopen it by using the task list.)

TABLELAYOUTPANEL CONTROL

- Choose the **Edit Rows and Columns** task to display the **Column and Row Styles** window. Choose **Column1**, and set its size to 15 percent by being sure the **Percent** button is selected and entering **15** in the **Percent** box. (That's a NumericUpDown control, which you'll use in a later tutorial.) Choose **Column2** and set it to 85 percent. Don't choose the **OK** button yet, because the window will close. (But if you do, you can reopen it by using the task list.)



TABLELAYOUTPANEL CONTROL

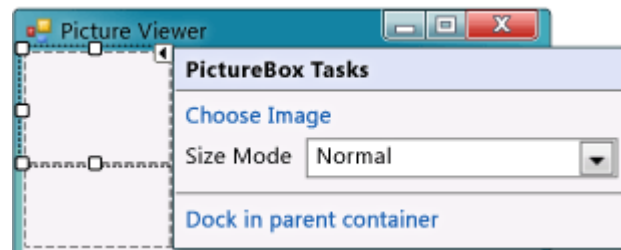
From the Show drop-down list at the top of the Column and Row Styles window, choose Rows. Set Row1 to 90 percent and Row2 to 10 percent.

Choose the OK button. Your TableLayoutPanel should now have a large top row, a small bottom row, a small left column, and a large right column. (You can resize the rows and columns in the TableLayoutPanel by choosing tableLayoutPanel1 in the form and then dragging its row and column borders.)



ADD CONTROLS

- Choose the **Toolbox** tab on the left side of the Visual Studio IDE (or press **Ctrl+Alt+X**), and then expand the **Common Controls** group. This shows the most common controls that you see on forms.
- Double-click the **PictureBox** item to add a PictureBox control to your form. Because the TableLayoutPanel is docked to fill your form, the IDE adds the PictureBox control to the first empty cell (the upper left corner).
- Choose the new **PictureBox** control to select it, and then choose the black triangle on the new PictureBox control to display its task list, as shown in the following screenshot.

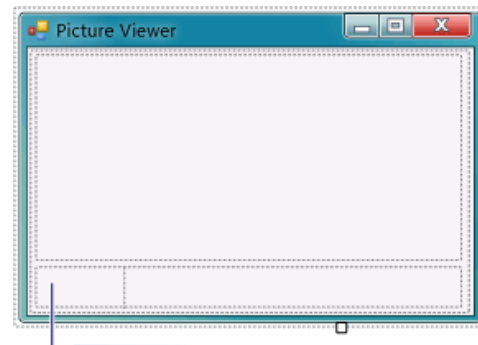


ADD CONTROLS

In the PictureBox Tasks menu from the PictureBox control, choose the Dock in parent container link. This automatically sets the PictureBox Dock property to Fill. To see this, choose the PictureBox control to select it, go to the Properties window, and be sure that the Dock property is set to Fill.

Make the PictureBox span both columns by changing its ColumnSpan property. In the PictureBox, choose the PictureBox control and set its ColumnSpan property to 2. Also, when the PictureBox is empty, you want to show an empty frame. Set its BorderStyle property to Fixed3D.

Choose the **TableLayoutPanel** on the form and then add a CheckBox control to the form. Double-click the **CheckBox** item in the **Toolbox** to add a new CheckBox control to the next free cell in your table. Because a PictureBox takes up the first two cells in the TableLayoutPanel, the CheckBox control is added to the lower-left cell. Choose the **Text** property and type in the word **Stretch**, as shown in the following image.



ADD BUTTONS

Choose the new FlowLayoutPanel that you added. Go to Common Controls in the Toolbox and double-click the Button item to add a button control called button1 to your FlowLayoutPanel. Repeat to add another button. The IDE determines that there's already a button called button1 and calls the next one button2.

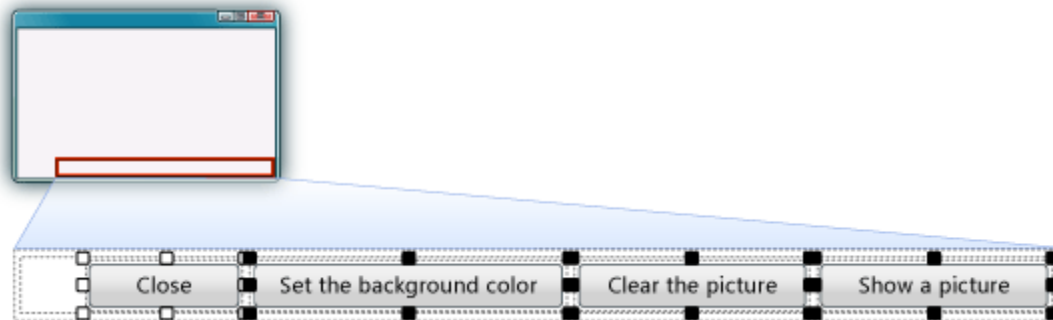
Typically, you add the other buttons by using the Toolbox. This time, choose button2, and then from the menu bar, choose Edit > Copy (or press Ctrl+C). Next, choose Edit > Paste from the menu bar (or press Ctrl+V) to paste a copy of your button. Now paste it again. Notice that the IDE adds button3 and button4 to the FlowLayoutPanel.

- Choose the first button and set its **Text** property to **Show a picture**. Then set the **Text** properties of the next three buttons to **Clear the picture**, **Set the background color**, and **Close**.
- Let's size the buttons and arrange them so they align to the right side of the panel. Choose the **FlowLayoutPanel** and look at its **FlowDirection** property. Change it so it's set to **RightToLeft**.

The buttons should align themselves to the right side of the cell, and reverse their order so that the **Show a picture** button is on the right.

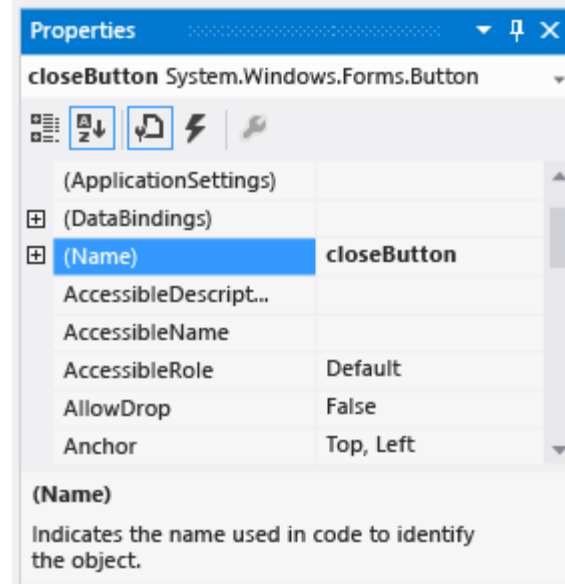
ADD BUTTONS

- Choose the **Close** button to select it. Then, to choose the rest of the buttons at the same time, press and hold the **Ctrl** key and choose them, too.
- After you've selected all the buttons, go to the **Properties** window and scroll up to the **AutoSize** property. This property tells the button to automatically resize itself to fit all of its text. Set it to **True**.
- Your buttons should now be sized properly and be in the right order. (As long as all four buttons are selected, you can change all four **AutoSize** properties at the same time.) The following image shows the four buttons.



NAME YOUR BUTTON CONTROLS

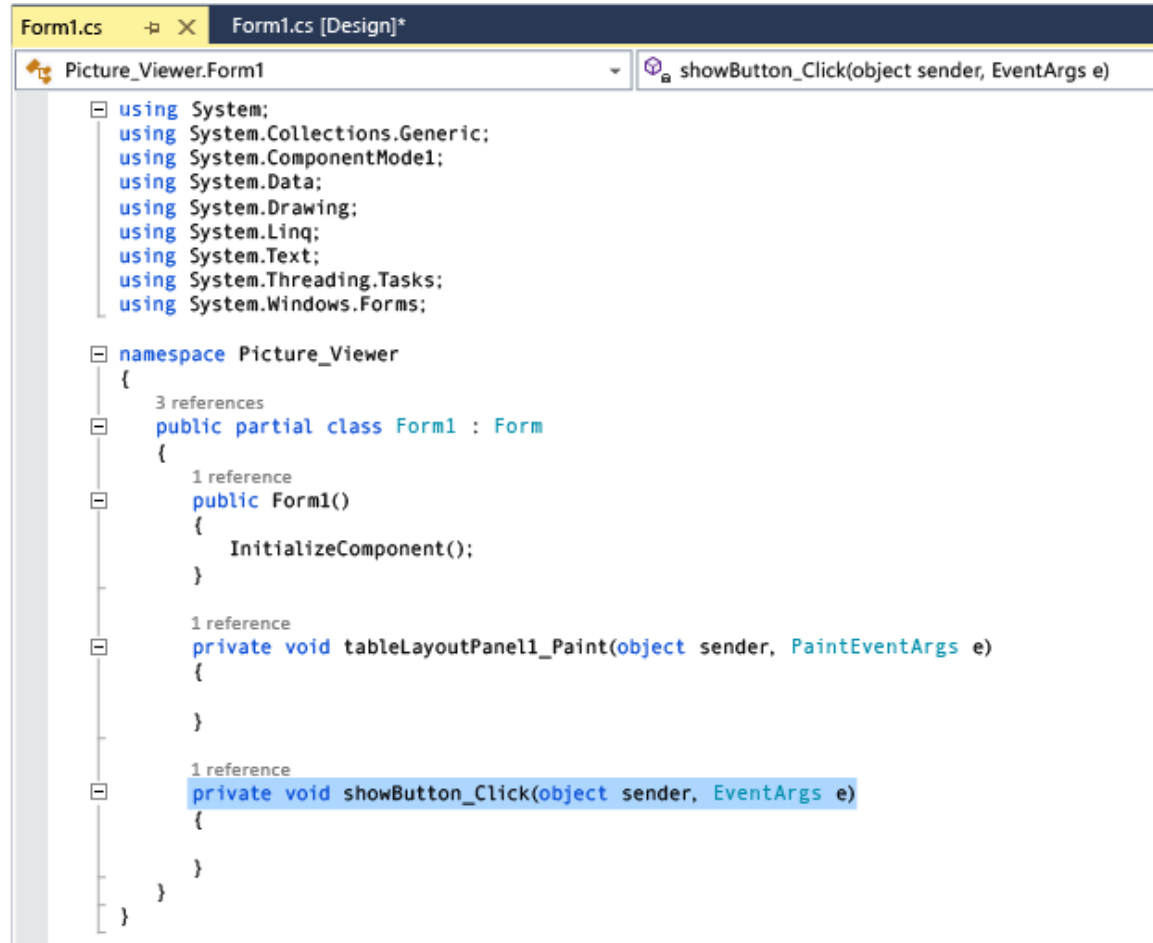
- On the form, choose the **Close** button. (If you still have all the buttons selected, choose the **Esc** key to cancel the selection.) Scroll in the **Properties** window until you see the **(Name)** property. (The **(Name)** property is near the top when the properties are alphabetical.) Change the name to **closeButton**, as shown in the following screenshot.



NAME YOUR BUTTON CONTROLS

- Rename the other three buttons to `backgroundButton`, `clearButton`, and `showButton`. You can verify the names by choosing the control selector drop-down list in the Properties window. The new button names appear.
- Double-click the Show a picture button on the form. As an alternative, choose the Show a picture button on the form, and then press the Enter key. When you do, the IDE opens an additional tab in the main window named `Form1.cs`. (If you're using Visual Basic, the tab is named `Form1.vb`).
- This tab displays the code file behind the form, as shown in the following screenshot.

NAME YOUR BUTTON CONTROLS



```
Form1.cs  Form1.cs [Design]*
Picture_View.Form1 showButton_Click(object sender, EventArgs e)

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Picture_View
{
    3 references
    public partial class Form1 : Form
    {
        1 reference
        public Form1()
        {
            InitializeComponent();
        }

        1 reference
        private void tableLayoutPanel1_Paint(object sender, PaintEventArgs e)
        {
        }

        1 reference
        private void showButton_Click(object sender, EventArgs e)
        {
        }
    }
}
```

BUTTON CLICKS

- You're looking at code called `showButton_Click()` (alternatively, `ShowButton_Click()`). The IDE added this to the form's code when you opened the code file for the `showButton` button. At design-time, when you open the code file for a control in a form, code is generated for the control if it doesn't already exist. This code, known as a method, runs when you run your app and choose the control - in this case, the Show a picture button.
- Choose the Windows Forms Designer tab again (`Form1.cs [Design]`), and then open the code file for the Clear the picture button to create a method for it in the form's code. Repeat this for the remaining two buttons. Each time, the IDE adds a new method to the form's code file.
- To add one more method, open the code file for the `CheckBox` control in Windows Forms Designer to make the IDE add a `checkBox1_CheckedChanged()` method. That method is called whenever the user selects or clears the check box.

BUTTON CLICKS

- `private void clearButton_Click(object sender, EventArgs e)`
- `{`
- `}`

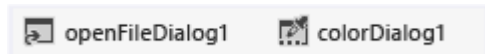
- `private void backgroundButton_Click(object sender, EventArgs e)`
- `{`
- `}`

- `private void closeButton_Click(object sender, EventArgs e)`
- `{`
- `}`

- `private void checkBoxI_CheckedChanged(object sender, EventArgs e)`
- `{`
- `}`

ADD DIALOG COMPONENTS TO THE FORM

- To add a component called `openFileDialog1` to your form, double-click `OpenFileDialog`. To add a component called `colorDialog1` to your form, double-click `ColorDialog` in the Toolbox. (You use that one in the next tutorial step.) You should see an area at the bottom of Windows Forms Designer (beneath the `PictureViewer` form) that has an icon for each of the two dialog components that you added, as shown in the following image.



- Dialog components
- Choose the `openFileDialog1` icon in the area at the bottom of the Windows Forms Designer. Set two properties:
- Set the `Filter` property to the following (you can copy and paste it):
`JPEG Files (*.jpg)|*.jpg|PNG Files (*.png)|*.png|BMP Files (*.bmp)|*.bmp|All files (*.*)|*.*`
- Set the `Title` property to the following: `Select a picture file`
- The `Filter` property settings specify the kinds of file types that will display in the `Select a picture file` dialog box.

ADD METHOD

```
■ private void showButton_Click(object sender, EventArgs e)
■ {
■     if (openFileDialog1.ShowDialog() == DialogResult.OK)
■     {
■         pictureBox1.Load(openFileDialog1.FileName);
■     }
■ }
```


ADDITIONAL BUTTONS

```
■ private void clearButton_Click(object sender, EventArgs e)
■ {
■     // Clear the picture.
■     pictureBox1.Image = null;
■ }

■ private void backButton_Click(object sender, EventArgs e)
■ {
■     // Show the color dialog box. If the user clicks OK, change the
■     // PictureBox control's background to the color the user chose.
■     if (colorDialog1.ShowDialog() == DialogResult.OK)
■         pictureBox1.BackColor = colorDialog1.Color;
■ }
```

ADDITIONAL BUTTONS

```
■ private void closeButton_Click(object sender, EventArgs e)
■ {
■     // Close the form.
■     this.Close();
■ }

■ private void checkBox1_CheckedChanged(object sender, EventArgs e)
■ {
■     // If the user selects the Stretch check box,
■     // change the PictureBox's
■     // SizeMode property to "Stretch". If the user clears
■     // the check box, change it to "Normal".
■     if (checkBox1.Checked)
■         pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;
■     else
■         pictureBox1.SizeMode = PictureBoxSizeMode.Normal;
■ }
```