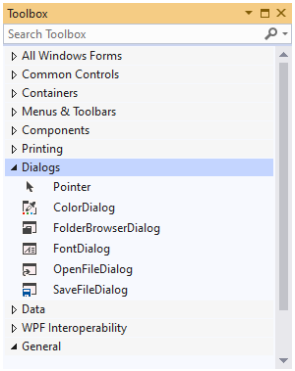
Dialogs

Most applications allow the user to browse to a specific location to store or retrieve files. Instead of hard coding the actual filename and path, dialog boxes are used to enter a name at runtime. This is very complicated functionality to implement if you have to write all the code yourself. Fortunately, C# makes it very simple for us to add dialogs to our programs – we just place a FileDialog control on the Form, and use its interface.

Placing a Dialog Controls on the Form

The Dialog controls are on the Dialogs tab of the ToolBox, note the North American spelling. There are several dialogs there, which you should experiment with them. Today we will concentrate on the OpenFileDialog and SaveFileDialog.

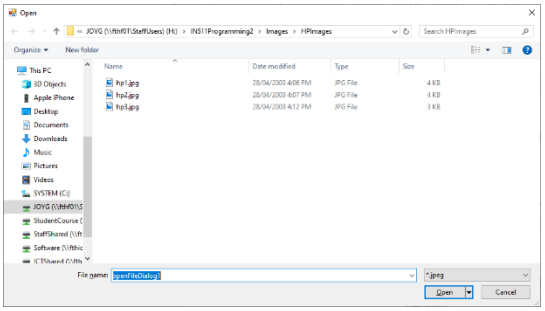


To include an OpenFileDialog in your application, click on the icon, then click on your Form, or double click on the icon. C# will name your component openDialog1. Again it does not matter where you place the icon, as it has no visual representation at runtime.

Using the OpenFileDialog Control

The OpenFileDialog control has an important method `ShowDialog()` and property `Filename()`. Assume that you have an OpenFileDialog control named openFileDialog1.

- **`ShowDialog()`** causes the program to display a standard Open File dialog box, as shown:



The **`ShowDialog()`** method returns a Boolean value. If the user selects a file in the dialog window and clicks OK, **ShowDialog()** returns True. If the user cancels the dialog window, **ShowDialog()** returns false. Thus we use the following statement to simultaneously present the Open File dialog window, and determine whether the user selected a file:

```
if (openFileDialog1.ShowDialog() == DialogResult.OK)
{
        ……
}
```

- **`FileName`** returns the name of the file selected by the user. This value is a string that can be passed directly to any method that accepts a string argument. For example, if you have an PictureBox control named pictureBox1, the following code fragment presents an Open File dialog window and, if the user selects a file, loads that file into the PictureBox component.

```
if (openFileDialog1.ShowDialog() == DialogResult.OK)
{
        string fileName = openFileDialog1.FileName;
        pictureBox1.Load(fileName);
}
```

  Thus you need write only three lines of code to provide functionality that actually requires hundreds. C# translates these three statements into the necessary underlying code.

  Note that we write no handlers for the OpenDialog control itself, we simply use its existing methods.

Task: Use OpenFileDialog and SaveFileDialog Controls

1. Create a new project.
2. Place two PictureBoxes, two Buttons and one OpenFileDialog control on the Form. Centre one button beneath each PictureBox component.
3. Write a handler for each button such that, when clicked, it allows the user to select (via an OpenFileDialog window) a file to load into the corresponding PictureBox, and displays the selected file.
4. Place two more buttons and one SaveFileDialog component on the Form.
5. Write a handler for each button so that, when clicked, it allows the user to save (via a SaveFileDialog window) the selected file.

Extra Option: OpenFileDialogs often restrict the type of file that the user can select. For example, if you invoke Open File from within Microsoft Word, the resulting window displays only those files that Word can actually open. If you navigate to a folder that contains, for example, a .bmp file, that file will not be shown. This feature is called a **filter,** and it protects the user from crashes caused by trying to open the wrong type of file. You can set a filter on your dialog component so that it displays only files of type .bmp and .jpg, by modifying its Filter property.
