PROGRAMAÇÃO O.O.
(*C#*)

{ Entrada e Saída de Arquivos (Parte 02)

Professor: João Luiz Lagôas

}

Caixas de Diálogo

Colégio Pedro II

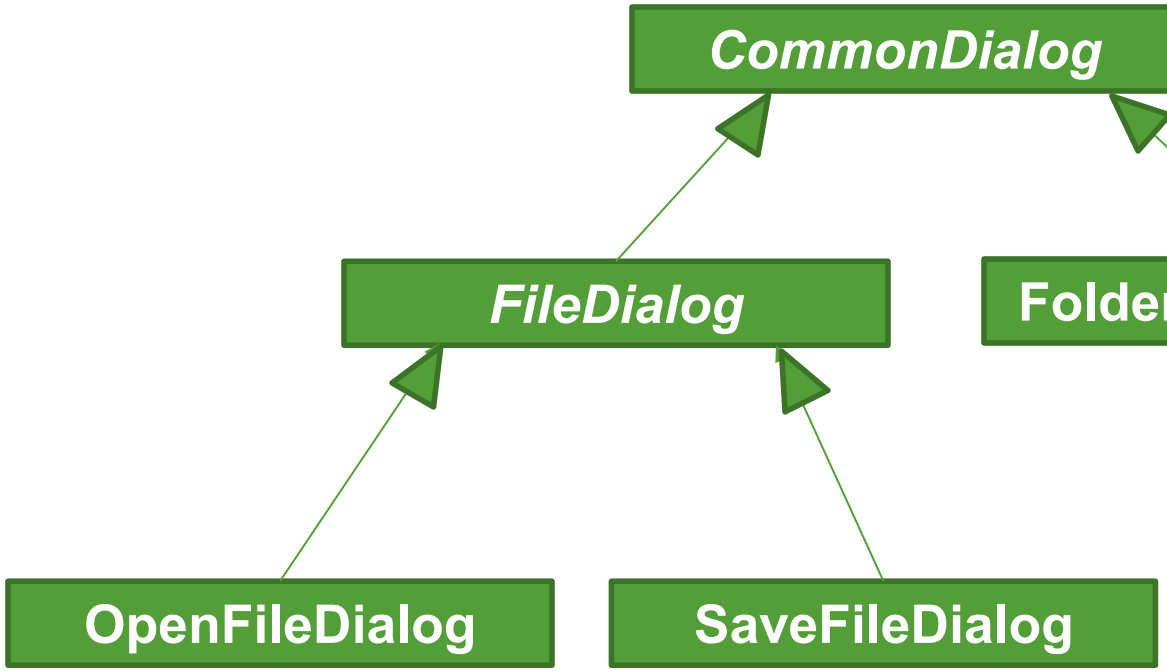
- Ao se trabalhar com um programa que lê e escreve arquivos, existe uma boa chance de se precisar mostrar uma **caixa de diálogo** em algum momento para solicitar ao usuário um nome de arquivo e/ou um diretório.

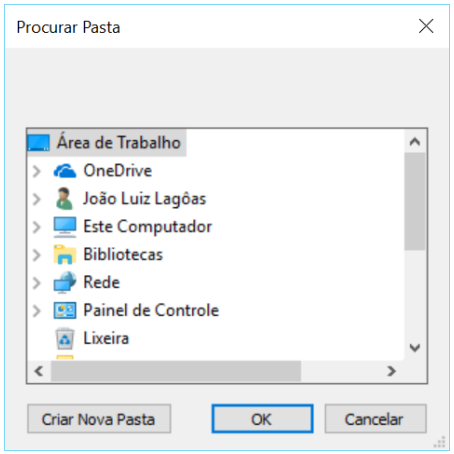
- A plataforma .NET conta com vários tipos de caixas de diálogo para serem usadas em nossos programas. Nós estudaremos três delas:
  - FolderBrowserDialog
  - OpenDialog
  - SaveDialog

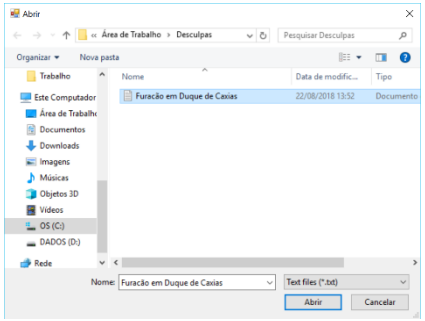
Caixas de Diálogo

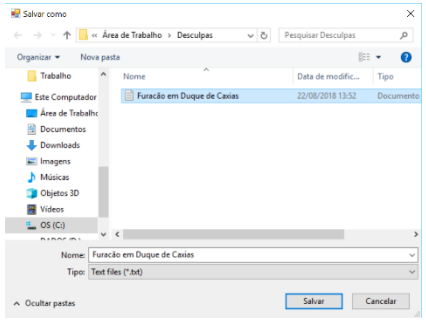
Tipos de Dialog Boxes

Colégio Pedro II







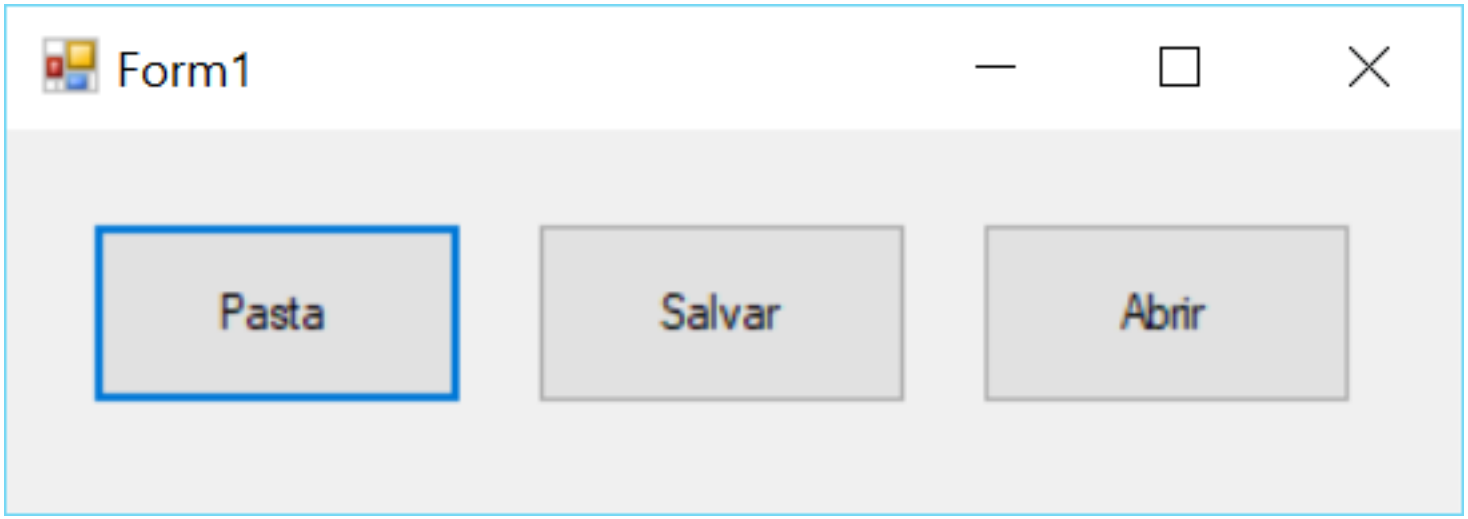


Caixa de Diálogo

Exemplo

Colégio Pedro II

- Vamos tomar o exemplo abaixo para entender como se é possível utilizar essas caixas de diálogo.



- Ao se clicar em cada botão, uma caixa de diálogo deve ser aberta com a respectiva função.

Caixa de Diálogo

OpenFileDialog

Colégio Pedro II

```
private void openButton_Click(object sender, EventArgs e)
        {
            OpenFileDialog openDialog = new OpenFileDialog();

            openDialog.InitialDirectory = @"C:\";
            openDialog.Filter = "Text files (*.txt)|*.txt|All files (*.*)|*.*";

            DialogResult result = openDialog.ShowDialog();

            if (result == DialogResult.OK)
            {
                MessageBox.Show("O usuário clicou em OK!");
                MessageBox.Show("O caminho do arquivo clicado é: " +
                                openDialog.FileName);
            }

            else if (result == DialogResult.Cancel)
            {
                MessageBox.Show("O usuário clicou em Cancel!");
            }
        }
```

Caixa de Diálogo

OpenFileDialog

Colégio Pedro II

- Um objeto OpenFileDialog mostra a janela de “Abrir” padrão do Windows.

- Para se usar um objeto OpenFileDialog, é comum seguir três passos:

1. Criar uma instância da classe OpenFileDialog com o comando new.
2. Alterar seus atributos de modo a personalizar seu objeto.
3. Chamar o método ShowDialog() para exibir a janela na tela.

Nota: Esse método retorna uma variável do tipo DialogResult armazenando a opção de clique do usuário.

Nota: O atributo **FileName** retorna o caminho do arquivo carregado.

Caixa de Diálogo

SaveFileDialog

Colégio Pedro II

```
private void saveButton_Click(object sender, EventArgs e)
        {
            SaveFileDialog saveDialog = new SaveFileDialog();

            saveDialog.InitialDirectory = @"C:\";
            saveDialog.Filter = "Text files (*.txt)|*.txt|All files (*.*)|*.*";

            DialogResult result = saveDialog.ShowDialog();

            if (result == DialogResult.OK)
            {
                MessageBox.Show("O usuário clicou em OK!");
                MessageBox.Show("O caminho do arquivo a ser salvo é: " +
                                saveDialog.FileName);
            }

            else if (result == DialogResult.Cancel)
            {
                MessageBox.Show("O usuário clicou em Cancel!");
            }
        }
```

Caixa de Diálogo

SaveFileDialog

Colégio Pedro II

- Um objeto SaveFileDialog mostra a janela de “Salvar” padrão do Windows.

- Para se usar um objeto SaveFileDialog, é comum seguir três passos:
  1. Criar uma instância da classe SaveFileDialog com o comando new.
  2. Alterar seus atributos de modo a personalizar seu objeto.
  3. Chamar o método ShowDialog() para exibir a janela na tela.

  Nota: Esse método retorna uma enum DialogResult armazenando a opção de clique do usuário.

  Nota: O atributo **FileName** retorna o caminho especificado do arquivo a ser salvado.

Caixa de Diálogo

FolderBrowserDialog

Colégio Pedro II

```
private void folderButton_Click(object sender, EventArgs e)
        {
            FolderBrowserDialog folderDialog = new FolderBrowserDialog();

            //Pode mudar algum atributo de folderDialog aqui...

            DialogResult result = folderDialog.ShowDialog();

            if (result == DialogResult.OK)
            {
                MessageBox.Show("O usuário clicou em OK!");
                MessageBox.Show("O caminho da pasta clicada é: " +
                                            folderDialog.SelectedPath);
            }

            else if(result == DialogResult.Cancel)
            {
                MessageBox.Show("O usuário clicou em Cancel!");
            }
        }
```

Caixa de Diálogo

FolderBrowserDialog

Colégio Pedro II

- Um objeto FolderBrowserDialog mostra a janela de “Abrir Pasta” padrão do Windows.

- Para se usar um objeto FolderBrowserDialog, é comum seguir três passos:
  1. Criar uma instância da classe FolderBrowserDialog com o comando new.
  2. Alterar seus atributos de modo a personalizar seu objeto.
  3. Chamar o método FolderBrowserDialog() para exibir a janela na tela.

  Nota: Esse método retorna uma enum DialogResult armazenando a opção de clique do usuário.

  Nota: Observe também o atributo **SelectedPath** do objeto FolderBrowserDialog para verificar qual caminho foi selecionado.

Classes *File* e *Directory*

Ganhando intuição

Colégio Pedro II

- O .NET possui duas classes nativas com vários métodos estáticos para trabalhar com arquivos e pastas. A classe File fornece métodos para trabalhar com arquivos e a Directory permite lidar com diretórios. Escreva o que você acha que fazem essas linhas de código.

Classes *File* e *Directory*

Ganhando intuição

Colégio Pedro II

Código	O que o código faz
`if(!Directory.Exists(@"C:\LP2")){ Directory.CreateDirectory(@"C:\LP2"); }`	
`if (Directory.Exists(@"C:\LP2")){ Directory.Delete(@"C:\LP2"); }`	
`File.Copy(@"C:\LP2\notas.txt", @"D:\LP2\nt.txt");`	
`DateTime myTime = Directory.GetCreationTime(@"C:\LP2\notas.txt");`	
`File.Delete(@"C:\LP2\notas.txt");`	
`File.WriteAllText(@"C:\LP2\materia2C.txt", @"Programação O.O., muitos controles, encapsulamento, herança avançada, polimorfismo, acesso a arquivos");`	

Classes *File* e *Directory*

Ganhando intuição

Colégio Pedro II

Código	O que o código faz
`if(!Directory.Exists(@"C:\LP2")){ Directory.CreateDirectory(@"C:\LP2"); }`	Checa se a pasta LP2 existe. Se não, uma pasta LP2 é criada.
`if (Directory.Exists(@"C:\LP2")){ Directory.Delete(@"C:\LP2"); }`	Checa se a pasta LP2 existe. Se sim, ela é deletada.
`File.Copy(@"C:\LP2\notas.txt", @"D:\LP2\nt.txt");`	Copia o arquivo notas.txt para o arquivo nt.txt.
`DateTime myTime = Directory.GetCreationTime(@"C:\LP2\notas.txt");`	Declara uma variável myTime e a atribui ao retorno do método. Note que esse método retorna um objeto do tipo DateTime.
`File.Delete(@"C:\LP2\notas.txt");`	Deleta o arquivo notas.txt.
`File.WriteAllText(@"C:\LP2\materia2C.txt", @"Programação O.O., muitos controles, encapsulamento, herança avançada, polimorfismo, acesso a arquivos");`	Cria um arquivo chamado materia2C.txt (se ele não existir) e escreve o conteúdo passado como segundo parâmetro.

Classes *File* e *Directory*

- Assim como StreamWriter, a classe **File** cria, nos bastidores, streams para você trabalhar com arquivos. Você pode usar seus métodos para realizar ações mais comuns sem ter de criar FileStreams primeiro.

- Os objetos **Directory** permitem trabalhar com diretórios inteiros cheios de arquivos e pode-se usá-los para fazer alterações na estrutura de pastas facilmente.

- A **documentação** contendo todos os atributos/propriedades e métodos dessas classes pode ser encontrada em:
  - https://msdn.microsoft.com/pt-br/library/system.io.file(v=vs.110).aspx
  - https://msdn.microsoft.com/pt-br/library/system.io.directory(v=vs.110).aspx

Classe *File*

Colégio Pedro II

- O que você pode fazer com a classe File

1. Descobrir se um arquivo existe: você pode checar para ver se um arquivo existe usando método **Exists**().
2. Ler e escrever no arquivo: você pode usar o método **OpenRead**() para acessar dados de um arquivo, ou o método **Create**() ou **OpenWrite**() para escrever nele.
3. Concatenar texto em um arquivo: O método **AppendAllText**() permite anexar texto a um arquivo criado. Inclusive, ele cria o arquivo se este não estiver lá quando o método executar.
4. Obter informações:

- o método **GetLastAccessTime**() e **GetLastWriteTime**() retornam a data e hora do último acesso e alteração do arquivo.

Classe *Directory*

Colégio Pedro II

- O que você pode fazer com a classe Directory

1. Criar um novo diretório: Crie um diretório usando o método **CreateDirectory**(). Tudo a fazer é fornecer o caminho que o método fará todo o resto.
2. Obter uma lista de arquivos no diretório: você pode criar uma matriz de arquivos em um diretório usando o método **GetFiles**(). Apenas informe ao método um diretório e ele se encarregará de tudo.
3. Apagar um diretório é bem simples também. Use o método **Delete**().