**ATIVIDADE 8.2 – IMPLEMENTANDO TRATAMENTO DE EXCEPTION E MÉTODOS (STATIC) ESTÁTICOS**

Nesta atividade, você irá aprender como fazer o seguinte:

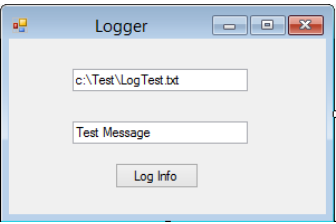
* Criar e chamar os métodos estático (static) de uma classe.
* Usar o Tratamento de Exception (exceção) estruturada.

**Criando métodos estáticos**

Para criar os métodos estáticos, siga estes passos:

1. Inicie o Visual Studio, Selecione *File* -> *New* -> *Project*.
2. Escolha um projeto *Windows Application*. Nomeie o projeto para *Activity8\_2*.
3. No Visual Studio crie um padrão de *form* (formulário) para o projeto com que você irá usar para criar um *form* de Login nomeado com *Logger*. Adicione os *controls* (controles) para o *form* e troque os valores das propriedades, listados abaixo. Em seguida o formulário ficará conforme o desenho (após as especificações.)

|  |  |  |
| --- | --- | --- |
| Object  (Objeto) | Properties (Propriedades) | Value  (Valores) |
| Form1 | Name  Text | frmLogger  Logger |
| Textbox1 | Name  Text | txtLogPath  c:\Test\LogTest.txt |
| Textbox2 | Name  Text | txtLogInfo  Test Message |
| Button1 | Name  Text | btnLogInfo  Log Info |



1. Selecione *Project* -> *Add Class*. Nomeie a classe para *Logger*.
2. Por causa disto, você irá usar a classe *System.IO* dentro da classe *Logger*, adicionamos uma declaração *using* no topo do arquivo da classe:

*using System.IO;*

1. Adicionamos um método estático *LogWrite p*ara a classe. Este método irá escrever a informação para um histórico do arquivo. Para abrir o arquivo, criamos um objeto *FileStream*. Então criamos um objeto *StreamWriter* para escrever a informação para o arquivo. Aviso do uso dos blocos *using* para (*dispose*) descartar adequadamente os objetos *FileStream* e *StreamWriter* e lançar os recursos.

*public static string LogWrite(string logPath, string logInfo)*

*{*

*using (FileStream oFileStream = new FileStream(logPath, FileMode.Open, FileAccess.Write))*

*{*

*using (StreamWriter oStreamWriter = new StreamWriter(oFileStream))*

*{*

*oFileStream.Seek(0, SeekOrigin.End);*

*oStreamWriter.WriteLine(DateTime.Now);*

*oStreamWriter.WriteLine(logInfo);*

*oStreamWriter.WriteLine();*

*}*

*}*

*return “Info Logged”;*

*}*

1. Abra o *frmLogger* na tela de editar o design. Dê duplo clique no botão (*button)* *btnLogInfo* para levantar a questão do método do evento *btnLogInfo\_Click* no editor do código. Adicionamos o seguinte código, com isso rodará o método *LogWrite* da classe *Logger* e mostra os resultados na propriedade text do formulário. Notamos que por causa de você desenhar o método *LogWrite* como estático (*static*), o cliente não precisa criar uma instância do objeto da classe *Logger*. Métodos estáticos são acessados diretamente através de uma referência da classe.

*private void btnLogInfo\_Click(object sender, EventArgs e)*

*{*

*this.Text = Logger.LogWrite(txtLogPath.Text, txtLogInfo.Text);*

*}*

1. Selecione *Build* -> *Build Solution*. Tenha certeza que não erros na janela *Error List*. Se tiver, então conserte-os, e então rode novamente.
2. Selecione *Debug* -> *Start* para rodar. Quando o lança o *form*, clique no *button* (botão) *Log Info*. Você poderia obter uma *exception* não-tratada do tipo *System.IO.DirectoryNotFoundException*. Pare o depurador.

**Criando o Manipulador de Exception (Exceção) Estruturado**

Para criar um manipulador de exceção (*exception*) estruturado, siga estes passos:

1. Abra o código da classe *Logger* no editor do código.
2. Localize o método *LogWrite* e adicione um bloco *try-catch* dentro do código atual. No bloco *catch*, retorna uma *string* afirmando que falha de acesso.

*try*

*{*

*using (FileStream oFileStream = new FileStream(logPath, FileMode.Open, FileAccess.Write))*

*{*

*using (StreamWriter oStreamWriter = new StreamWriter(oFileStream))*

*{*

*oFileStream.Seek(0, SeekOrigin.End);*

*oStreamWriter.WriteLine(DateTime.Now);*

*oStreamWriter.WriteLine(logInfo);*

*oStreamWriter.WriteLine();*

*}*

*}*

*return “Info Logged”;*

*}*

*catch*

*{*

*return “Logging Failed”;*

*}*

1. Selecione *Build* -> *Build Solution*. Tenha certeza que não erros na janela *Error List*. Se tiver, então conserte-os, e então rode novamente.
2. Selecione *Debug* -> *Run*. Quando o lança o *form*, clique no *button* (botão) *Log Info*. Neste tempo, você poderia não obter uma mensagem de *exception* (exceção) isto que estava manipulado pelo método *LogWrite*. Você deve ver uma mensagem *“Logging Failed”* na legenda do form. Feche o form.

**Filtrando Exceptions**

Para filtrar as exceptions (exceções), siga estes passos:

1. Depois o bloco *catch* retornar diferente mensagens, dependendo da *exception* é lançada.

*catch (FileNotFoundException ex)*

*{*

*return ex.Message;*

*}*

*catch (IOException ex)*

*{*

*return ex.Message;*

*}*

*catch*

*{*

*return “Logging Failed”;*

*}*

1. Setar um *break point* no método *LogWrite* da classe *Logger*.
2. Selecione *Debug* -> *Start* para rodar o projeto. Teste o bloco *catch* clicando no *botão Log Info*. Executando irá parar no *break point*. Passo através do código e avise isto para obter a captura pelo bloco *IOException*.
3. Depois de testado, feche o *form*.
4. Usando o Notepad, criamos o arquivo *LogText.txt* em uma pasta *Test* no drive (disco) *C* e feche o arquivo. Tenha a certeza que o *arquivo* e a *pasta* não estão marcada como apenas leitura (read only).
5. Selecione *Debug* -> *Start* para rodar. Testamos o método *WriteLog* clicando no botão *Log Info*. Neste tempo, o *form* capturado deve indicar que o histórico foi escrito com sucesso.
6. Para o depurador.
7. Abra o arquivo *LogTest.txt* usando Notepad e verificado que a informação estava logado.
8. Salve o projeto, e então saia do Visual Studio.