**ATIVIDADE 8-3. CHAMANDO OS MÉTODOS ASSÍNCRONAMENTE**

Nesta atividade, você irá aprender como fazer o seguinte:

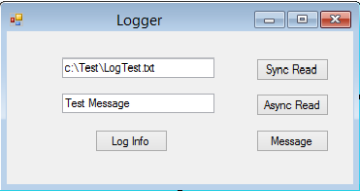
* Chamar o método sincronamente
* Chamar o método assincronamente.

**Criando um método e chamando isto sincronamente**

Para criar o método e chamar isto sincronamente, siga estes passos:

1. Inicie o Visual Studio, Selecione *File* -> *Open* -> *Project*.
2. Abra o arquivo na *Solution* que você completou na *Activity8\_2*.
3. Adicione os botões mostrados na tabela abaixo para o (*form*) *frmLogger*. Após a tabela mostra o *(form)* completado.

|  |  |  |
| --- | --- | --- |
| Object  (Objeto) | Properties (Propriedades) | Value  (Valores) |
| Button1 | Name  Text | btnSyncRead  Sync Read |
| Button2 | Name  Text | btnAsyncRead  Async Read |
| Button3 | Name  Text | btnMessage  Message |



1. Abra a classe *Logger* no editor de código.
2. Chame novamente, pois que você está usando a *namespace System.IO* dentro da classe *Logger*, você irá adicionar usando uma declaração para o topo do arquivo. Você também está indo para o uso do *System*.

*using System.Threading;*

1. Adicione uma *function (método) LogRead* estática para a classe. Esta function irá ler as informações de um arquivo de relatório (log). Para abrir o arquivo, criamos um objeto *FileStream*. Então criamos o objeto *StreamReader* para ler a informação do arquivo. Você também está usando a classe *Thread* para suspender processando por cinco segundos para simular uma longa chamada de um lado de uma rede de internet lenta.

*public static string LogRead(string filePath)*

*{*

*StreamReader oStreamReader;*

*string fileText;*

*try*

*{*

*oStreamReader = File.OpenText(filePath);*

*fileText = oStreamReader.ReadToEnd();*

*oStreamReader.Close();*

*Thread.Sleep(5000);*

*return fileText;*

*}*

*catch (FileNotFoundException ex)*

*{*

*return ex.Message;*

*}*

*catch (IOException ex)*

*{*

*return ex.Message;*

*}*

*catch*

*{*

*return “Logging Failed”;  
}*

*}*

1. Abra o *form* (*frmLogger*) na tela de editor de Design. Dê um duplo clique no *button* (botão) *btnMessage* para levantar a questão do evento do método *btnMessage\_Click* no editor de código. Adicione o código na tela de uma *message box* (caixa de mensagem).

*private void btnMessage\_Click(object sender, EventArg e)*

*{*

*MessageBox.Show(“Hello”);*

*}*

1. Abra o *form* (frmLogger) no editor de Design. Dê um duplo clique no *button* (botão) *btnSyncRead* para levar a questão do evento do método *btnSyncRead\_Click* no editor do código. Adicione um código que irá chamar o método *LogRead* da classe *Logger* e mostre o resultado em uma *message box* (caixa de mensagem).

*public void btnSyncRead\_Click(object sender, EventArgs e)*

*{*

*MessageBox.Show(Logger.LogRead(txtLogPath.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* -> *Run*. Quando o lança o form, clique no *button* (botão) *Sync Read*. Depois clicando no *button* (botão) *Sync Read*, tente clicar no *button* (botão) *Message*. Você não poderia obter uma resposta quando clica no *button* (botão) *Message* por causa de você chamar o método *ReadLog* sincronamente. Depois o método *ReadLog* retorna um resultado, o *button* (botão) *Message* irá responder quando for clicado.
3. Quando você finaliza o teste, feche o *form* (formulário).

**Criando e chamando um método assíncrono**

Para criar um método assíncrono, siga estes passos:

1. Abra a classe *Logger* no editor de código.
2. Verifique para a seguinte declaração *using* no topo do arquivo. Esta *namespace* expõem a classe *Task* e outros tipos que são usados para implementar a programação assíncrona.

*using System.Threading.Task;*

1. Crie um método assíncrono que ler um arquivo texto. O uso do método *Task´s Delay* é para simular um processo de longo tempo.

*public static async Task<string> LogReadAsync(string filePath)*

*{*

*string fileText;*

*try*

*{*

*using(StreamReader oStreamReader = File.OpenText(filePath))*

*{*

*fileText = await oStreamReader.ReadToEndAsync();*

*}*

*await Task.Delay(10000);*

*return fileText;*

*}*

*catch (FileNotFoundException ex)*

*{*

*return ex.Message;*

*}*

*catch (IOException ex)*

*{*

*return ex.Message;*

*}*

*catch*

*{*

*return “Logging Failed”;*

*}*

*}*

1. Abra o *form* (formulário) *frmLogger* na tela do editor de design. Dê um duplo clique no *button* (botão) *btnAsyncRead* para levantar a questão do método do evento *btnAsynRead\_Click* no editor de código. Altere o método assim que isto é assíncrono

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

1. Adicione o código para chamar o método *LogReadAsync* da classe *Logger* e mostre os resultados na *message box* (caixa de mensagem).

*btnAsyncRead.Enabled = false;*

*string s = await Logger.LogReadAsync(txtLogPath.Text);*

*MessageBox.Show(s);*

*btnAsyncRead.Enabled = true;*

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 lançamos o *form* (formulário), clique no *button* (botão) *Async Read*. Depois clique no *button* (botão) *Async Read*, clique no *button* (botão) *Message*. Este tempo, você poderia obter uma resposta por causa que você chamou o método assincronamente *ReadLog*. Depois de cinco segundos você deve ver uma mensagem contendo o resultado do método *Logger.LogReadAsync*.
3. Quando você finaliza o teste, feche o *form* (formulário).
4. Salve o projeto, e então sai do Visual Studio.