

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace MyFileStream

{

class Program

{

private object ex;

static void Main(string[] args)

{

var di = Directory.CreateDirectory(@"D:\new folder");

Thread th = new Thread(new ParameterizedThreadStart(CreateDirectory));

th.Start(di);

Thread thDel = new Thread(new ParameterizedThreadStart(DeleteDirectory));

thDel.Start(di);

Console.ReadKey();

}

static public void CreateDirectory(Object dirInf)

{

var di = dirInf as DirectoryInfo;

Console.WriteLine("Ready 1 - " + Thread.CurrentThread.Name);

if (di.Exists)

{

lock (di)

{

for (int i = 0; i < 10;i++)

{

di.CreateSubdirectory(string.Format("directory {0}", i));

Console.WriteLine("Create - directory {0}", i);

Thread.Sleep(300);

}

}

}

}

static public void DeleteDirectory(Object dirInf)

{

var di = dirInf as DirectoryInfo;

Console.WriteLine("Ready 2 -"+Thread.CurrentThread.Name);

lock (di)

{

if (di.Exists)

{

di.Delete(true);

Console.WriteLine("Delete - directory");

}

}

}

}

}



using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace FileStreamNew

{

class Program

{

static void Main(string[] args)

{

var di=Directory.CreateDirectory(@"D:\FileStream");

Thread th=new Thread(new ParameterizedThreadStart(CreateFile));

th.Start(di);

Console.ReadKey();

}

static void CreateFile(object dirIn)

{

var di = dirIn as DirectoryInfo;

if (di.Exists)

{

var fs = new FileStream(di.FullName+"\\text.txt",FileMode.OpenOrCreate,FileAccess.Write);

var fsCopyTo =

new FileStream(

@"C:\Users\Виктор\Desktop\c#\Biblija\_C\_CD\_Bookforall.ru\Уроки\ProfessionalC#\213.txt",

FileMode.Open, FileAccess.Read);

MemoryStream ms=new MemoryStream();

byte[] arr=new byte[1024];

int size;

while ((size=fsCopyTo.Read(arr, 0, arr.Length)) > 0)

{

ms.Write(arr,0,size);

}

ms.Position = 0;

StreamReader sr=new StreamReader(ms,Encoding.GetEncoding(1251));

while (!sr.EndOfStream)

{

Console.WriteLine(sr.ReadLine());

}

sr.BaseStream.Position = 0;

ms.CopyTo(fs);

sr.Close();

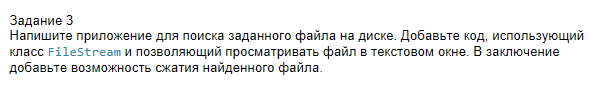
fs.Close();

}

}

}

}



using System;

using System.Collections.Generic;

using System.IO;

using System.IO.Compression;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

namespace WPF\_ChangeFile

{

class ZipClass

{

public void ZipCompress(object path)

{

var pathTmp = path.ToString();

if (!File.Exists(pathTmp))

throw new Exception("File not found");

var fs = new FileStream(pathTmp,FileMode.Open,FileAccess.Read);

pathTmp = Path.GetPathRoot(pathTmp) + "tmp.zip";

var fsCreate = File.Open(pathTmp, FileMode.OpenOrCreate, FileAccess.Write);

var zc = new GZipStream(fsCreate,CompressionMode.Compress);

int b;

while ((b=fs.ReadByte())!=-1)

{

zc.WriteByte((byte)b);

}

zc.Close();

fs.Close();

MessageBox.Show("Zip Complete");

}

}

}

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace WPF\_ChangeFile

{

class Model

{

class MyElem

{

public string Str { get; set; }

public StringBuilder Sb { get; set; }

public MyElem()

{

Sb=new StringBuilder();

}

}

public StringBuilder GetTextFromFile(string str)

{

if(!File.Exists(str))

throw new Exception("File not found");

MyElem me = new MyElem { Str = str};

Thread th=new Thread(ReadFile);

th.Start(me);

th.Join();

return me.Sb;

}

private void ReadFile(object s)

{

var fs = new FileInfo((s as MyElem).Str).OpenRead();

var sr = new StreamReader(fs);

var ss = s as MyElem;

while (!sr.EndOfStream)

{

ss.Sb.AppendLine(sr.ReadLine());

Thread.Sleep(50);

}

sr.Close();

}

public void SetIzolatedStoarage(object s)

{

var str = s.ToString();

IsolatedStorageFile user = IsolatedStorageFile.GetUserStoreForAssembly();

if (!user.DirectoryExists("MyWPF"))

user.CreateDirectory("MyWPF");

IsolatedStorageFileStream ifs = new IsolatedStorageFileStream(@"MyWPF\MyStoarage.set", FileMode.OpenOrCreate, user);

StreamWriter sw=new StreamWriter(ifs);

sw.WriteLine(str);

sw.Close();

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Diagnostics;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows;

namespace WPF\_ChangeFile

{

class Presenter

{

private Model model;

private MainWindow wind;

BackgroundWorker bw = new BackgroundWorker();

public Presenter(MainWindow mw)

{

wind = mw;

wind.ButOk.Click += ButOk\_Click;

wind.BZip.Click += BZip\_Click;

model = new Model();

bw.RunWorkerCompleted += Comleted;

bw.DoWork += bw\_DoWork;

}

void BStorage\_Click(object sender, RoutedEventArgs e)

{

new Thread(new ParameterizedThreadStart(model.SetIzolatedStoarage)).Start(wind.ListB.Text);

}

void BZip\_Click(object sender, RoutedEventArgs e)

{

new Thread(new ParameterizedThreadStart(new ZipClass().ZipCompress)).Start(wind.TxtBox.Text);

}

void bw\_DoWork(object sender, DoWorkEventArgs e)

{

try

{

e.Result = model.GetTextFromFile(e.Argument.ToString());

}

catch (Exception ex)

{

e.Result = "file not found";

MessageBox.Show(ex.Message);

}

}

private void Comleted(object sender, RunWorkerCompletedEventArgs e)

{

wind.ListB.Text = e.Result.ToString();

}

void ButOk\_Click(object sender, System.Windows.RoutedEventArgs e)

{

if(!bw.IsBusy)

bw.RunWorkerAsync(wind.TxtBox.Text);

}

}

}