EXAMPLE OF THREAD

using System;

using System.Threading;

using System.Threading.Tasks;

namespace ThreadExample1

{

class Program

{

static void Main(string[] args)

{

Thread ob = Thread.CurrentThread;

ob.Name = "Mainthread";

Console.WriteLine(ob.Name);

}

}

}

STATIC THREAD FUNCTIONS

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace ThreadExample1

{

class CustomThread

{

public static void Threadfun()

{

for (int i = 0; i < 20; i++)

Console.WriteLine(i);

}

}

class StaticThreadclass

{

Threadstart

Is delegate

static void Main(string[] args)

{

//Thread t1 = new Thread(new ThreadStart(CustomThread.Threadfun));

Thread t1 = new Thread(delegate() { CustomThread.Threadfun(); });

Thread t2 = new Thread(new ThreadStart(CustomThread.Threadfun));

Thread t3 = new Thread(() => CustomThread.Threadfun());

t1.Start();

t2.Start();

t3.Start();

}

}

}

NON STATIC THREAD METHODS

using System;

using System.Threading;

namespace ThreadExample1

{

class CustomThread1

{

public void Threadfun()

{

for (int i = 0; i < 20; i++)

Console.WriteLine(i);

}

}

class NonStatic

{

static void Main(string[] args)

{

CustomThread1 myth = new CustomThread1();

Thread t1 = new Thread(myth.Threadfun);

Thread t2 = new Thread(myth.Threadfun);

//output can be anything cause there is context switching

t1.Start();

t2.Start();

}

}

}

THREAD WITH PARAMETER NOT SAFE

using System;

using System.Threading;

namespace ThreadExample1

{

class Myparam

{

public void showme(object x)

{

int n = 0;

if (int.TryParse(x.ToString(), out n))

{

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

Console.WriteLine(Thread.CurrentThread.Name + " ............. " + i);

}

}

}

class Threadwithparamcs

{

static void Main(string[] args)

{

Myparam ob = new Myparam();

Console.WriteLine("Enter the parameter/number");

object num = Console.ReadLine();

ParameterizedThreadStart pa = new ParameterizedThreadStart(ob.showme);

Thread t = new Thread(pa);

//Thread t = new Thread(ob.showme);

t.Start(num);

}

}

}

THREAD WITH PARAMETER SAFE

using System;

using System.Threading;

namespace ThreadExample1

{

class Num

{

private int num = 0;

public Num(int num)

{

this.num = num;

}

public void showme(object x)

{

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

Console.WriteLine(Thread.CurrentThread.Name + " ............. " + i);

}

}

class Safeparameterized

{

static void Main(string[] args)

{

Console.WriteLine("Enter the parameter/number");

int num = int.Parse(Console.ReadLine());

Num ob = new Num(num);

Thread d = new Thread(ob.showme);

d.Start();

}

}

}

Retrieving the sum of natural numbers with threads and delegate

using System;

using System.Threading;

namespace ThreadExample1

{

public delegate void sumofnum(int val);

class Num1

{

private int num = 0;

sumofnum callbackmeth;

public Num1(int num,sumofnum inc)

{

this.num = num;

callbackmeth = inc;

}

public void showmeval()

{

int newval = 0;

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

newval += i;

if (callbackmeth != null)

callbackmeth(newval);

}

}

class RetrievingData

{

public static void sumofnumfun(int sum)

{

Console.WriteLine("Sum of nums = "+sum);

}

static void Main(string[] args)

{

Console.WriteLine("Enter the parameter/number");

int num = int.Parse(Console.ReadLine());

sumofnum call = new sumofnum(sumofnumfun);

Num1 ob = new Num1(num,call);

Thread d = new Thread(new ThreadStart(ob.showmeval));

d.Start();

}

}

}

STATIC DIFFERENT TASKS FROM EACH THREAD

using System;

using System.Threading;

namespace ThreadExample1

{

class Mythread2

{

public static void fun1()

{

Console.WriteLine("Function1");

}

public static void fun2()

{

Console.WriteLine("Function2");

}

public static void fun3()

{

Console.WriteLine("Function3");

}

}

class Class1

{

static void Main(string[] args)

{

Thread a1 = new Thread(new ThreadStart(Mythread2.fun1));

Thread a2 = new Thread(new ThreadStart(Mythread2.fun2));

Thread a3 = new Thread(new ThreadStart(Mythread2.fun3));

a1.Start();

a2.Start();

a3.Start();

}

}

}

using System;

using System.Threading;

using System.Threading.Tasks;

namespace ThreadExample1

{

class Mythread3

{

public void fun()

{

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

{

Console.WriteLine( Thread.CurrentThread.Name +" " +i);

Thread.Sleep(400);

}

}

}

class SleepThread

{

static void Main(string[] args)

{

Mythread3 ob = new Mythread3();

Thread t1 = new Thread(new ThreadStart(ob.fun));

Thread t2 = new Thread(new ThreadStart(ob.fun));

t1.Name = "One";

t2.Name = "Two";

t1.Start();

t2.Start();

}

}

}

ABORT UNPREDICTABLE OUTPUT

using System;

using System.Threading;

namespace ThreadExample1

{

class MyThread4

{

public void fun()

{

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

{

Console.WriteLine(i);

Thread.Sleep(200);

}

}

}

class Abortmeth

{

static void Main(string[] args)

{

MyThread4 ob = new MyThread4();

Thread t1 = new Thread(new ThreadStart(ob.fun));

Thread t2 = new Thread(new ThreadStart(ob.fun));

t1.Name = "One";

t2.Name = "Two";

t1.Start();

t2.Start();

try

{

t1.Abort();

t2.Abort();

}

catch(ThreadAbortException e)

{

Console.WriteLine(e.ToString());

}

}

}

}

JOIN

It causes all the calling threads to wait until the current thread (joined thread) is terminated or completes its task.

using System;

using System.Threading;

using System.Threading.Tasks;

namespace ThreadExample1

{

class MyThread5

{

public void fun()

{

for (int i = 0; i < 5; i++)

{

Console.WriteLine(Thread.CurrentThread.Name + " " +i);

Thread.Sleep(200);

}

}

}

class Join

{

static void Main(string[] args)

{

MyThread5 ob = new MyThread5();

Thread t1 = new Thread(new ThreadStart(ob.fun));

Thread t2 = new Thread(new ThreadStart(ob.fun));

Thread t3 = new Thread(new ThreadStart(ob.fun));

t1.Name = "One";

t2.Name = "Two";

t3.Name = "Three";

t1.Start();

t1.Join();

t2.Start();

t3.Start();

}

}

}

PRIORITY

The high priority thread can be executed first. But it is not guaranteed because thread is highly system dependent. It increases the chance of the high priority thread to execute before low priority thread.

using System;

using System.Threading;

namespace ThreadExample1

{

class Priority

{

static void Main(string[] args)

{

MyThread5 ob = new MyThread5();

Thread t1 = new Thread(new ThreadStart(ob.fun));

Thread t2 = new Thread(new ThreadStart(ob.fun));

Thread t3 = new Thread(new ThreadStart(ob.fun));

t1.Name = "One";

t2.Name = "Two";

t3.Name = "Three";

t2.Priority = ThreadPriority.Highest;

t1.Priority = ThreadPriority.Lowest;

t3.Priority = ThreadPriority.Normal;

t1.Start();

t2.Start();

t3.Start();

}

}

}

REFLECTION

using System;

using System.Reflection;

namespace ThreadExample1

{

class Relection1

{

static void Main(string[] args)

{

int a = 10;

Type type = a.GetType();

Console.WriteLine(type);

Type t1 = typeof(System.String);

Console.WriteLine(t1.Assembly);

Type t = typeof(System.String);

Console.WriteLine(t.FullName);

Console.WriteLine(t.BaseType);

Console.WriteLine(t.IsClass);

Console.WriteLine(t.IsEnum);

Console.WriteLine(t.IsInterface);

Type t2 = typeof(System.String);

Console.WriteLine("Constructors of {0} type...", t2);

ConstructorInfo[] ci = t2.GetConstructors(BindingFlags.Public | BindingFlags.Instance);

foreach (ConstructorInfo c in ci)

{

Console.WriteLine(c);

}

string path = @"C:\Users\User\source\repos\DatabaseFirstDemo\DatabaseFirstDemo\bin\Debug\EntityFramework.dll";

Assembly ass = Assembly.LoadFile(path);

Type[] t3 = ass.GetTypes();

foreach(var ty in t3)

{

Console.WriteLine( "CLASS = " + ty.Name);

MethodInfo[] meth = ty.GetMethods();

foreach(var m in meth)

{

Console.WriteLine("Method names = "+m.Name);

}

}

}

}

}

REFLECTION MAIN

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ThreadExample1

{

class Employee

{

public int Eid { get; set; }

public string Ename { get; set; }

public float Sal { get; set; }

public Employee()

{

Eid = 1;

Ename = "Joseph";

Sal = 2222;

}

public Employee(int x,string n , float s)

{

Eid = x;

Ename = n;

Sal = s;

}

public void getdata()

{

Console.WriteLine("Enter id,name and sal");

Eid = int.Parse(Console.ReadLine());

Ename = Console.ReadLine();

Sal = float.Parse(Console.ReadLine());

}

}

}

using System;

using System.Reflection;

namespace ThreadExample1

{

class Reflection2

{

static void Main(string[] args)

{

Type t = Type.GetType("ThreadExample1.Employee");

Console.WriteLine("FULL NAME = " + t.FullName);

Console.WriteLine("NAME = " + t.Name);

Console.WriteLine("NAMESPACE = " + t.Namespace);

Console.WriteLine("\n\n");

Console.WriteLine("properties = " );

PropertyInfo[] prop = t.GetProperties();

foreach(PropertyInfo p in prop)

{

Console.WriteLine(p.Name + " - " + p.PropertyType.Name);

}

Console.WriteLine("\n\n");

Console.WriteLine("Methods = ");

MethodInfo[] meth = t.GetMethods();

foreach (MethodInfo m in meth)

{

Console.WriteLine(m.Name + " - " + m.ReturnType.Name);

}

Console.WriteLine("\n\n");

Console.WriteLine("Methods = ");

ConstructorInfo[] methCon = t.GetConstructors();

foreach (ConstructorInfo c in methCon)

{

Console.WriteLine(c.ToString() );

}

}

}

}