**DAY 59 Hands-on \_ C# Additional Topics (Async , Multithreading)**

**Hands-on1 : Named and Optional Parameters**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace D3Handson1

{

class NamedParameter

{

static void GetCohortDetails(string Cohortname, int Genccount, string Mode, string Track, string CurrentModule)

{

Console.WriteLine("It is {0} with {1} GenCs undergoing training for {2} thru {3}. The current module of training is {4}", Cohortname, Genccount, Track, Mode, CurrentModule);

}

// Optional Prameters

static void OrderDetails(string Productname, string Sellername, int Orderquantity = 1, bool returnable = true)

{

Console.WriteLine("Here is the order detail – {0} number of {1} by {2} is ordered. It’s returnable status is {3}", Orderquantity, Productname, Sellername, returnable);

}

static void Main(string[] args)

{

Console.WriteLine("Cohort Details");

// Named parameters

GetCohortDetails(Cohortname: "CDE", Genccount: 20, Track: "Java ", Mode: "OBL", CurrentModule: "Stage3");

GetCohortDetails(Cohortname: "CDE", Genccount: 20, Mode: "PARC", Track: ".Net", CurrentModule: "Stage3");

Console.WriteLine("-------------------------------------");

// Optional parameters

Console.WriteLine("Order Details");

OrderDetails(Sellername: "abc", Productname: "xyz", Orderquantity: 10, returnable: false);

OrderDetails(Sellername: "abc", Productname: "xyz");

Console.WriteLine("-------------------------------------");

Console.ReadLine();

}

}

}

**--------------------------------------------------------------------------------------**

**Handson1: async1**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace AsyncHandsOn

{

class Program

{

public static async Task Method1()

{

await Task.Run(() =>

{

string a = Method2();

Console.WriteLine(a);

});

}

public static string Method2()

{

Console.WriteLine("wait for string to return");

Thread.Sleep(5000);

return "this is method 2";

}

static void Main(string[] args)

{

Method1();

Console.ReadKey();

}

}

}

**--------------------------------------------------------------------------------------**

**Hands-on1 : async2**

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsHandsOn

{

static class Program

{

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

Form1.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsHandsOn

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

public int CountChars()

{

int count = 0;

using (StreamReader sr = new StreamReader("C:\\Users\\badrinath somisetti\\Desktop\\Badri.txt"))

{

string content = sr.ReadToEnd();

count=content.Length;

Thread.Sleep(2000);

}

return count;

}

private async void button1\_Click(object sender, EventArgs e)

{

Task<int> task = new Task<int>(CountChars);

task.Start();

label1.Text = "File is processing";

int count = await task;

label1.Text = count.ToString()+" characters";

}

private void label1\_Click(object sender, EventArgs e)

{

}

}

}

**---------------------------------------------------**

**Handson 2 : multithreading 1**

Printer.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace ThreadingHandsOn

{

class Printer

{

public void PrintNumber()

{

Console.WriteLine("->{0} is executing PrintNumbers()", Thread.CurrentThread.Name);

//Print out numbers.

Console.WriteLine("Your numbers: ");

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

{

Console.Write("{0}, ", i);

Thread.Sleep(2000);

}

Console.WriteLine();

}

}

}

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace ThreadingHandsOn

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("\*\*\*\*\*ThreadStart Delegate App\*\*\*");

Console.WriteLine("Do You Want [1] or [2] threads?");

string threadCount = Console.ReadLine();

//Name the current Thread

Thread primaryThread = Thread.CurrentThread;

primaryThread.Name = "Primary";

//Display Thread Info

Console.WriteLine("-> {0} is executing Main()", Thread.CurrentThread.Name);

//Make Worker Class

Printer p = new Printer();

switch (threadCount)

{

case "2":

//now make the thread.

//give only method name

Thread backgroundThread = new Thread(new ThreadStart(p.PrintNumber))

{

Name = "Secondary"

};

backgroundThread.Start();

//changes the state of current instance to ThreadState.Running

break;

case "1":

p.PrintNumber();

break;

default:

Console.WriteLine("I don't know what you want... you get 1 Thread");

goto case "1";

}

Thread.Sleep(1000);

//Do some additional work.

Console.WriteLine("Hello this from main!");

}

}

}

**--------------------------------------------------------------------------------------**

**Hands-on2 : multithreading 2**

Printer.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading;

namespace multithreading

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("\*\*\*\*\*\*\*Multithreading Program\*\*\*\*\*\*\*\*\n");

Console.WriteLine("Main thread started. ThreadID = {0}", Thread.CurrentThread.ManagedThreadId);

Printer p = new Printer();

WaitCallback workItem = new WaitCallback(PrintTheNumbers);

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

{

ThreadPool.QueueUserWorkItem(workItem, p);

}

Console.WriteLine("All task Queued");

Console.ReadLine();

}

static void PrintTheNumbers(object state)

{

Printer task = (Printer)state;

task.PrintNumbers();

}

}

}

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading;

namespace multithreading

{

public class Printer

{

private object lockToken = new object();

public void PrintNumbers()

{

lock (lockToken)

{

Console.WriteLine("Thread-> {0} started @{1} and executing PrintNumbers() method", Thread.CurrentThread.ManagedThreadId, DateTime.Now.ToLongTimeString());

Console.Write("your numbers");

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

{

Console.Write("{0}, ", i); Thread.Sleep(500);

}

Console.WriteLine();

}

}

}

}

---------------------------------------------------------------------------------------------------------