Codigo unidad 3

viernes, 11 de diciembre de 2020 18:14

Ejemplo01

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Ejemplo01
{
   class Program
   {
      static void Main(string[] args)
      {
         //Creamos los hilos
         //Hilo 1 [Utilizando un pardon para crear Hilos]
         Task.Factory.StartNew(()=>ProcesoHilo1());
         //Hilo 2 [Forma Tradicional de crear un hilo]
         var t = new Task(() => ProcesoHilo2());
         t.Start();

         //Imprimir mensaje
         Console.WriteLine("Programa Principal!!");
         Console.ReadKey();
      }

      private static void Escribir(char v)
      {
         int ciclo = 1000;
         for (int i = 0; i < ciclo; i++)
         {
            Console.Write(v);
         }
      }

      private static void ProcesoHilo1() {
         int ciclo = 1000;
         for (int i = 0; i < ciclo; i++)
         {
            Console.Write('-');
         }
      }
      private static void ProcesoHilo2()
      {
         int ciclo = 1000;
         for (int i = 0; i < ciclo; i++)
         {
            Console.Write('+');
         }
      }
   }
```

```
}
```

Ejemplo02

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Ejemplo02
{
   class Program
   {
      static void Main(string[] args)
      {
         String text1 = "Prueba 1", text2 = "Prueba con mayor información 2";

         //Hilos 1
         var tarea1 = new Task<int>(LengthText, text1);
         tarea1.Start();
         //Hilos 2
         Task<int> tarea2 = Task.Factory.StartNew(LengthText, text2);
         //Hilo 3
         Task.Factory.StartNew(() => {
            Console.WriteLine($"Tamaño del texto {text1} es {tarea1.Result}");
            Console.WriteLine($"Tamaño del texto {text2} es {tarea2.Result}");
         });

         //Imprimimos los resultados
         Console.WriteLine("Programa Pricnipal");
         //Establecemos la paus
         Console.ReadKey();
      }

      /// <summary>
      /// Función que permite obtener el tamaño de un texto
      /// </summary>
      /// <param name="obj"></param>
      /// <returns></returns>
      private static int LengthText(Object obj)
      {
         Console.WriteLine($"Tarea con Id {Task.CurrentId} Procesando {obj}");
         return obj.ToString().Length;
      }
   }
}
```

Ejemplo03

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
using System.Threading;
using System.Threading.Tasks;

namespace Ejemplo03
{
   class Program
   {
      static void Main(string[] args)
      {
         //Declaracion de variables para cancelar procesos
         var cts = new CancellationTokenSource();
         var token = cts.Token;

         //Evento de cancelación
         token.Register(()=> {
            Console.WriteLine($"Proceso Cancelado!!! {Task.CurrentId}");
         });

         //Hilo con cancelación de token
         var hilo1 = new Task(()=>BucleInfinito(ref token), token);
         hilo1.Start();

         Console.ReadKey();
         cts.Cancel();

         //Mostramos el mensaje de finalización del programa
         Console.WriteLine("Programa Principal finalizado!!");
         Console.ReadKey();
      }

      private static void BucleInfinito(ref CancellationToken token)
      {
         int contador = 0;
         bool flag = true;
         while (flag)
         {
            try
            {
               //Verificamos si existe una solicitud de cancelación
               token.ThrowIfCancellationRequested();
            }
            catch (Exception)
            {
               flag = false;
            }
            Console.WriteLine($"{contador++}\t");
         }
      }
   }
}
```

Ejemplo04

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Ejemplo04
{
   static class Program
   {
      /// <summary>
      /// Punto de entrada principal para la aplicación.
      /// </summary>
      [STAThread]
      static void Main()
      {
         Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);
         Application.Run(new Form1());
      }
   }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Ejemplo04
{
   public partial class Form1 : Form
   {
      int opc = 0;
      bool hilo1Pausado = false;
      bool hilo2Pausado = false;

      bool hilo1Activo = false;
      bool hilo2Activo = false;

      public Form1()
      {
         InitializeComponent();
      }

      private void rdbtnSuma_CheckedChanged(object sender, EventArgs e)
      {
         if (this.rdbtnSuma.Checked) {
            this.opc = 1;
         }
      }
```

```
private void rdbtnResta_CheckedChanged(object sender, EventArgs e)
{
   if (this.rdbtnResta.Checked) {
      this.opc = 2;
   }
}

private void rdbtnMulti_CheckedChanged(object sender, EventArgs e)
{
   if (this.rdbtnMulti.Checked) {
      this.opc = 3;
   }
}

private void btnProcesar_Click(object sender, EventArgs e)
{
   Operar();
}

private void Operar()
{
   var numero1 = Int16.Parse(this.txtNumero1.Text);
   var numero2 = Int16.Parse(this.txtNumero2.Text);
   var resultado = 0;
   var operacion = "";
   switch (this.opc)
   {
      case 1:
         //Suma
         operacion = "Suma";
         resultado = numero1 + numero2;
         break;
      case 2:
         //Resta
         operacion = "Resta";
         resultado = numero1 - numero2;
         break;
      case 3:
         //Multiplicación
         operacion = "Multiplicación";
         resultado = numero1 * numero2;
         break;
      default:
         break;
   }

   //MessageBox.Show($"El resultado de {operacion} es: {resultado.ToString()}");
   this.labelResultado.Text = $"El resultado de {operacion} es: {resultado.ToString()}";
}


#region "METODOS PARA EL HILO 1"
private void btnHilo1_Click(object sender, EventArgs e)
{
   if (this.hilo1Activo)
   {
      this.hilo1Activo = false;
```

```
        this.Hilo1.CancelAsync();
        this.btnHilo1.Text = "Hilo 1";
    }
    else
    {
        this.hilo1Activo = true;
        this.Hilo1.RunWorkerAsync();
        this.btnHilo1.Text = "Cancelar";
    }

}

private void Hilo1_DoWork(object sender, DoWorkEventArgs e)
{
    //Trabajo pesado
    for (int i = 0; i < 100; i++)
    {
        //this.pgrBarHilo1.Value = i;
        this.Hilo1.ReportProgress(i);

        pausarHilo1();

        //Verificamos si existe una cancelación pendiente
        if (this.Hilo1.CancellationPending)
        {
            e.Cancel = true;
            break;
        }
        Thread.Sleep(500);
    }
}

private void pausarHilo1()
{
    //Verificamos si existe una pausa
    if (this.hilo1Pausado)
    {
        while (this.hilo1Pausado) ;
    }
}

private void Hilo1_ProgressChanged(object sender, ProgressChangedEventArgs e)
{
    //Mostramos el avance atraves del delegado en el progressbar
    DelegateProgressBar1(e.ProgressPercentage);
}
private void Hilo1_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e)
{
    if (e.Cancelled)
    {
        MessageBox.Show("Proceso 1 cancelado!");
    }
    else if (e.Error != null){
        MessageBox.Show("Ocurrio un error en el Proceso 1");
    }
    else {
        MessageBox.Show("Proceso 1 Terminado con exito");
```

```
    }
}

/// <summary>
/// Delegado que permite la actualización de la interfaz del ProgressBar 1
/// </summary>
/// <param name="avance"></param>
private void DelegateProgressBar1(int avance) {
    //Validamos que el progress bar no se encuentre en uso por el hilo principal
    if (InvokeRequired) {
        //Llamada para modificación de la interfaz
        Invoke(new Action<int>(DelegateProgressBar1), avance);
    }else {
        //Modificamos la interfaz
        this.pgrBarHilo1.Value = avance;
    }
}
#endregion

#region "METODOS PARA EL HILO 2"

private void btnHilo2_Click(object sender, EventArgs e)
{
    if (this.hilo2Activo)
    {
        this.hilo2Activo = false;
        this.Hilo2.CancelAsync();
        this.btnHilo2.Text = "Hilo 2";
    }
    else
    {
        this.hilo2Activo = true;
        this.Hilo2.RunWorkerAsync();
        this.btnHilo2.Text = "Cancelar";
    }

}

private void Hilo2_DoWork(object sender, DoWorkEventArgs e)
{
    //Trabajo pesado
    for (int i = 0; i < 100; i++)
    {
        //this.pgrBarHilo2.Value = i;
        this.Hilo2.ReportProgress(i);
        pausarHilo2();
        //Verificamos si existe una cancelación pendiente
        if (this.Hilo2.CancellationPending) {
            e.Cancel = true;
            break;
        }
        Thread.Sleep(500);
    }
}

private void pausarHilo2()
{
```

```
    //Verificamos si existe una pausa
    if (this.hilo2Pausado)
    {
        while (this.hilo2Pausado) ;
    }
}

private void Hilo2_ProgressChanged(object sender, ProgressChangedEventArgs e)
{
    //Mostramos el avance a traves del delegado del progressbar
    DelegateProgressBar2(e.ProgressPercentage);
}
/// <summary>
/// Delegado que permite la actualización de la interfaz del ProgressBar 1
/// </summary>
/// <param name="avance"></param>
private void DelegateProgressBar2(int avance)
{
    //Validamos que el progress bar no se encuentre en uso por el hilo principal
    if (InvokeRequired)
    {
        //Llamada para modificación de la interfaz
        Invoke(new Action<int>(DelegateProgressBar2), avance);
    }
    else
    {
        //Modificamos la interfaz
        this.pgrBarHilo2.Value = avance;
    }
}

#endregion


private void Hilo2_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e)
{
    if (e.Cancelled)
    {
        MessageBox.Show("Proceso 2 cancelado!");
    }
    else if (e.Error != null)
    {
        MessageBox.Show("Ocurrio un error en el Proceso 1");
    }
    else
    {
        MessageBox.Show("Proceso 2 Terminado con exito");
    }
}

private void btnPausa1_Click(object sender, EventArgs e)
{
    if (this.hilo1Pausado)
    {
        this.hilo1Pausado = false;
    }
    else {
```

```
            this.hilo1Pausado = true;
        }
    }

    private void btnPausa2_Click(object sender, EventArgs e)
    {
        if (this.hilo2Pausado)
        {
            this.hilo2Pausado = false;
        }
        else
        {
            this.hilo2Pausado = true;
        }
    }
  }
}
```