**实验3.2 线程和线程池的基本用法**

实验相关材料位于：<ftp://192.168.40.100/>下的“**--- 梁其洋**”目录下（可以通过搜索名字找到）

作业提交到：[ftp://192.168.40.14/梁其洋/网络编程\*\*班/实验\*](ftp://192.168.40.14/梁其洋/网络编程**班/实验*)

要求：

1. 提交时**将本文档重新命名**，文档命名规则：学号+姓名+实验\*，例如，**517300614400 张三 实验\*.docx （一定要学号在前姓名在后，方便自动排序！）**

2. 你们没有FTP文件的删除权限，**如果需要提交新版本，在姓名后加序号后提交即可，例如，517300614400张三（1）实验\*.docx**

**实验目的：**

掌握线程和线程池的基本用法

**实验步骤：**

（1）新建一个名为example2的WPF项目（用其他工程名时注意修改xaml和cs代码中的命名空间）。输入下列代码，其中window标签的部分属性已给出。要求用自己的中文名字命名Window窗体的Title属性。

<Window x:Class="example2.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

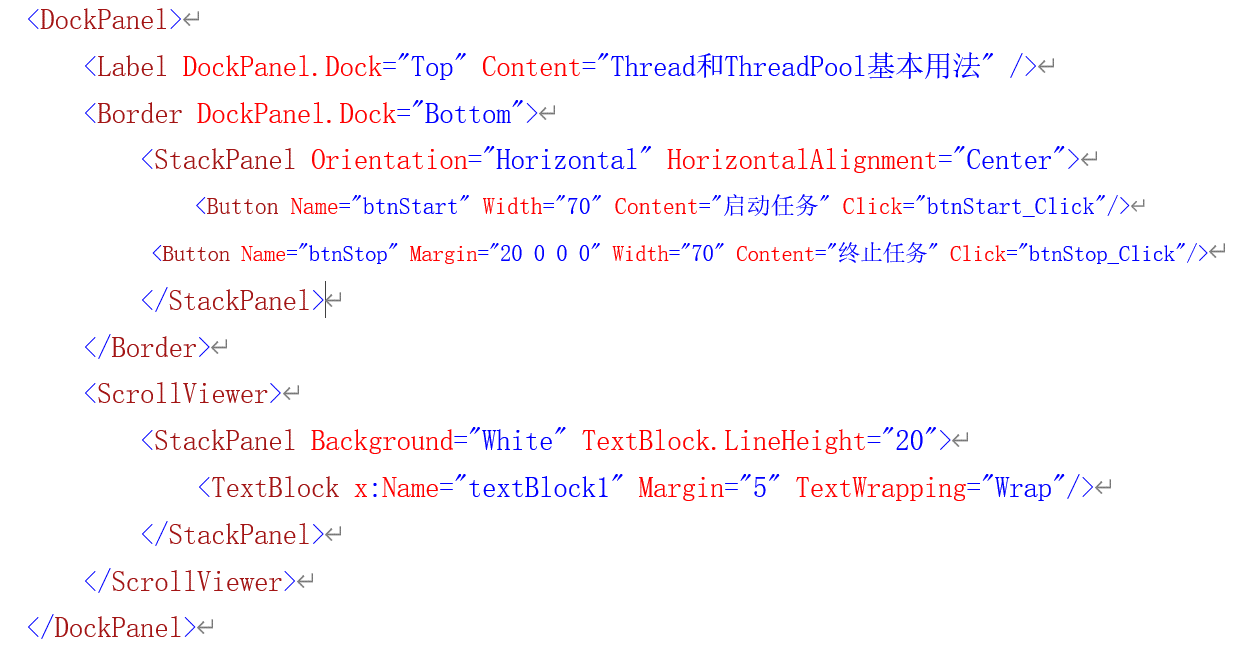
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:example2"

mc:Ignorable="d"

Title="张三\_MainWindow" Height="350" Width="525"> 

</Window>

（2）切换到MainWindow.xaml.cs中，添加using System.Threading;

添加启动任务按钮的功能：

private void btnStart\_Click(object sender, RoutedEventArgs e)

{

MyClass.IsStop = false;

textBlock1.Text = "";

MyClass c = new MyClass(textBlock1);

MyData state = new MyData { Message = "a", Info = "\n线程1已终止" };//或者采用下面两行注释来实例化Mydata

//MyData state = new MyData();

//state.Message = "a"; state.Info = "\n线程1已终止";

Thread thread1 = new Thread(c.MyMethod); //线程是通过委托执行相应的方法

thread1.IsBackground = true;

thread1.Start(state); //带参数的线程

state = new MyData { Message = "b", Info = "\n线程2已终止" };

Thread thread2 = new Thread(c.MyMethod);

thread2.IsBackground = true;

thread2.Start(state);

state = new MyData { Message = "c", Info = "\n线程3已终止" };

ThreadPool.QueueUserWorkItem(new WaitCallback(c.MyMethod), state); //向线程池中添加工作项, 开启一个线程

state = new MyData { Message = "d", Info = "\n线程4已终止" };

ThreadPool.QueueUserWorkItem(new WaitCallback(c.MyMethod), state);

}

private void btnStop\_Click(object sender, RoutedEventArgs e)

{

MyClass.IsStop = true;//改变MyClass的状态参数，三个正在执行的线程会改变输出

}

在public partial class MainWindow : Window之外添加两个辅助类MyClass和MyData：

**public class MyClass**

{

public static volatile bool IsStop;

TextBlock textBlock1;

public MyClass(TextBlock textBlock1)

{

this.textBlock1 = textBlock1;

}

public void **MyMethod**(Object obj)

{

MyData state = obj as MyData;

while (IsStop == false)

{

AddMessage(state.Message);

Thread.Sleep(100); //当前线程休眠100ms

}

AddMessage(state.Info);

}

private void AddMessage(string s)

{

textBlock1.Dispatcher.Invoke(() =>

{

textBlock1.Text += s;

});

}

}

**public class MyData**

{

public string Info { get; set; }

public string Message { get; set; }

}

<Window x:Class="sy3.\_2.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:sy3.\_2"

mc:Ignorable="d"

Title="舒洪凡——MainWindow" Height="450" Width="800">

<DockPanel>

<Label DockPanel.Dock="Top" Content="Thread和ThreadPool基本语法"></Label>

<Border DockPanel.Dock="Bottom">

<StackPanel Orientation="Horizontal" HorizontalAlignment="Center">

<Button Name="btnStart" Width="70" Content="启动任务" Click="BtnStart\_OnClick"></Button>

<Button Name="btnStop" Margin="20 0 0 0 " Width="70" Content="终止任务" Click="BtnStop\_OnClick"/>

</StackPanel>

</Border>

<ScrollViewer>

<StackPanel Background="White" TextBlock.LineHeight="20">

<TextBlock x:Name="textBlock1" Margin="5" TextWrapping="Wrap"></TextBlock>

</StackPanel>

</ScrollViewer>

</DockPanel>

</Window>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace sy3.\_2

{

internal class MyData

{

public string Info { get; set; }

public string Message { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Controls;

namespace sy3.\_2

{

internal class MyClass

{

public static volatile bool IsStop;

TextBlock textBlock1;

public MyClass(TextBlock textBlock1)

{

this.textBlock1 = textBlock1;

}

public void MyMethod(Object obj)

{

MyData state = obj as MyData;

while (IsStop == false)

{

AddMessage(state.Message);

Thread.Sleep(100); //当前线程休眠100ms

}

AddMessage(state.Info);

}

private void AddMessage(string s)

{

textBlock1.Dispatcher.Invoke(() =>

{

textBlock1.Text += s;

});

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace sy3.\_2

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void BtnStart\_OnClick(object sender, RoutedEventArgs e)

{

MyClass.IsStop = false;

textBlock1.Text = "";

MyClass c = new MyClass(textBlock1);

MyData state = new MyData { Message = "a", Info = "\n线程1已终止" };//或者采用下面两行注释来实例化Mydata

//MyData state = new MyData();

//state.Message = "a"; state.Info = "\n线程1已终止";

Thread thread1 = new Thread(c.MyMethod); //线程是通过委托执行相应的方法

thread1.IsBackground = true;

thread1.Start(state); //带参数的线程

state = new MyData { Message = "b", Info = "\n线程2已终止" };

Thread thread2 = new Thread(c.MyMethod);

thread2.IsBackground = true;

thread2.Start(state);

state = new MyData { Message = "c", Info = "\n线程3已终止" };

ThreadPool.QueueUserWorkItem(new WaitCallback(c.MyMethod), state); //向线程池中添加工作项, 开启一个线程

state = new MyData { Message = "d", Info = "\n线程4已终止" };

ThreadPool.QueueUserWorkItem(new WaitCallback(c.MyMethod), state);

}

private void BtnStop\_OnClick(object sender, RoutedEventArgs e)

{

MyClass.IsStop = true;//改变MyClass的状态参数，三个正在执行的线程会改变输出

}

}

}

程序运行截图：

