**实验5.2异步编程**

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**

**5.4 取消或终止任务的执行**

实验步骤：

（1）新建一个WPF工程，在MainWindow.xaml中添加以下代码：

<DockPanel>

<Label DockPanel.Dock="Top" Content="获取任务执行进度的基本用法"/>

<Border DockPanel.Dock="Bottom" >

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

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

<Button Name="btnCancel" Margin="20 0 0 0" Width="90" Content="取消任务执行" Click="btnCancel\_Click"/>

</StackPanel>

</Border>

<StackPanel>

<Label Margin="0 10 0 0" Content="任务1（可取消）："/>

<Grid Background="White">

<ProgressBar Name="progressBar1" Background="AliceBlue" Height="20" Margin="10" />

<TextBlock x:Name="textBlock1" HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Grid>

<TextBlock Text="任务2（不可取消）："/>

<Grid Background="White">

<ProgressBar Name="progressBar2" Background="AliceBlue" Height="20" Margin="10" />

<TextBlock x:Name="textBlock2" HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Grid>

</StackPanel>

</DockPanel>

（2）在主类MainWindow 类中创建一个取消源对象

public partial class MainWindow : Window

{

CancellationTokenSource cts;

……

}

在主类MainWindow 类中添加

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

（3）将btnStart的Click事件处理程序改为异步事件处理程序，即添加async关键字：

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

{

图形用户界面, 文本, 应用程序

描述已自动生成

文本, 应用程序

中度可信度描述已自动生成

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图形用户界面

中度可信度描述已自动生成

（4）实现btnCancel按钮的Click事件

图片包含 图形用户界面

描述已自动生成

程序截图：

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 sy5\_4

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

CancellationTokenSource cts;

public MainWindow()

{

InitializeComponent();

}

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

{

cts = new CancellationTokenSource();

CancellationToken ct = cts.Token;

textBlock1.Text = "";

var p1 = new Progress<int>();

p1.ProgressChanged += (s, n) =>

{

progressBar1.Value = n;

textBlock1.Text = " " + n.ToString() + "%";

};

var p2 = new Progress<int>();

p2.ProgressChanged += (s, n) =>

{

progressBar2.Value = n;

textBlock2.Text = " " + n.ToString() + "%";

};

//传递cts.Token表示该任务接收取消通知

var t1 = Task.Run(() => MyMethod(p1, cts.Token, 500), cts.Token);

//传递CancellationToken.None表示该任务不可取消

var t2 = Task.Run(() => MyMethod(p2, CancellationToken.None, 800));

try

{

await t1;

if (t1.Exception == null)

{

textBlock1.Text += " 任务完成";

}

}

catch(Exception ex)

{

textBlock1.Text += " " + ex.Message;

}

try

{

await t2;

if (t2.Exception == null)

{

textBlock2.Text += " 任务完成";

}

}

catch (Exception ex)

{

textBlock2.Text += " " + ex.Message;

}

}

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

{

cts.Cancel();

}

private void MyMethod(IProgress<int> progress,CancellationToken ct,int delay)

{

int p = 0;

while(p<100 && ct.IsCancellationRequested == false)

{

System.Threading.Thread.Sleep(delay);

p += 10;

progress.Report(p);

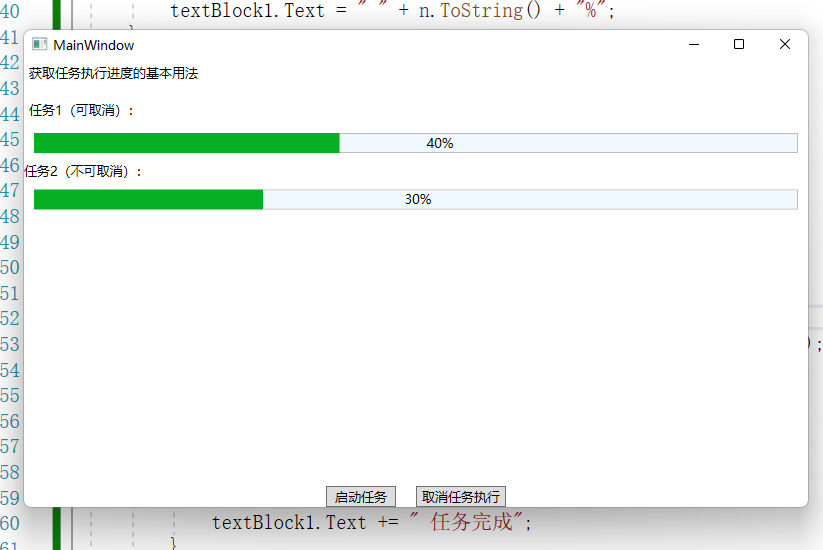
}

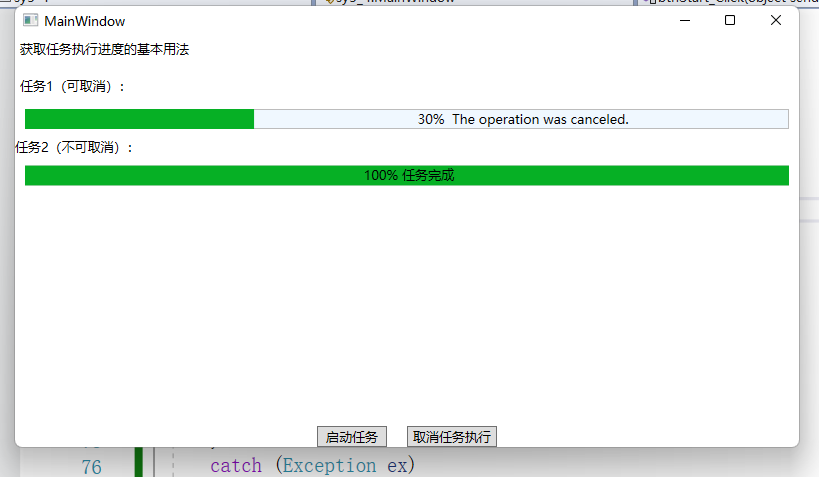
ct.ThrowIfCancellationRequested();

}

}

}





**5.5 获取任务执行的状态**

实验步骤：

（1）在mainWindow.xaml文件中添加以下代码：

<DockPanel>

<Label DockPanel.Dock="Top" Content="获取任务执行状态的基本用法" />

<Border DockPanel.Dock="Bottom">

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

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

<Button Name="btnCancel" Margin="20 0 0 0" Width="90" Content="取消任务执行" Click="btnCancel\_Click"/>

</StackPanel>

</Border>

<ScrollViewer>

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

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

</StackPanel>

</ScrollViewer>

</DockPanel>

（2）在主类MainWindow 类中创建一个取消源对象

图形用户界面, 文本

描述已自动生成

在主类MainWindow 类中添加

图形用户界面, 文本, 应用程序

描述已自动生成

（3）将btnStart的Click事件处理程序改为异步事件处理程序，即添加async关键字：

图形用户界面, 文本, 电子邮件

描述已自动生成

文本

描述已自动生成

（4）实现btnCancel按钮的Click事件

图片包含 图形用户界面

描述已自动生成

程序截图：

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 sy5\_5

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

CancellationTokenSource cts;

public MainWindow()

{

InitializeComponent();

}

private void MyMethod(CancellationTokenSource cts)

{

var ct = cts.Token;

while (ct.IsCancellationRequested == false)

{

System.Threading.Thread.Sleep(100);

}

ct.ThrowIfCancellationRequested();

}

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

{

textBlock1.Text = "开始执行任务...";

cts = new CancellationTokenSource();

var t1 = Task.Run(() => MyMethod(cts), cts.Token);

textBlock1.Text += "\n任务状态(每秒获取1次):";

while (t1.IsCanceled == false)

{

textBlock1.Text += t1.Status + "----";

await Task.Delay(TimeSpan.FromSeconds(1));

}

try

{

await t1;

} catch(Exception ex)

{

textBlock1.Text += "\n" + ex.Message;

textBlock1.Text += String.Format("\nStatus:{0},IsCompleted:{1},IsFaulted:{2},IsCanceled:{3}",t1.Status,t1.IsCompleted,t1.IsFaulted,t1.IsCanceled);

}

}

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

{

cts.Cancel();

}

}

}

