Asynchronous File I/O

.NET Framework (current version)

Asynchronous operations enable you to perform resource-intensive I/O operations without blocking the main thread. This performance consideration is particularly important in a Windows 8.x Store app or desktop app where a time-consuming stream operation can block the UI thread and make your app appear as if it is not working.

Starting with the .NET Framework 4.5, the I/O types include async methods to simplify asynchronous operations. An async method contains `Async` in its name, such as ReadAsync, WriteAsync, CopyToAsync, FlushAsync, ReadLineAsync, and ReadToEndAsync. These async methods are implemented on stream classes, such as Stream, FileStream, and MemoryStream, and on classes that are used for reading from or writing to streams, such TextReader and TextWriter.

In the .NET Framework 4 and earlier versions, you have to use methods such as BeginRead and EndRead to implement asynchronous I/O operations. These methods are still available in the .NET Framework 4.5 to support legacy code; however, the async methods help you implement asynchronous I/O operations more easily.

Starting with Visual Studio 2012, Visual Studio provides two keywords for asynchronous programming:

`Async` (Visual Basic) or `async` (C#) modifier, which is used to mark a method that contains an asynchronous operation.

`Await` (Visual Basic) or `await` (C#) operator, which is applied to the result of an async method.

To implement asynchronous I/O operations, use these keywords in conjunction with the async methods, as shown in the following examples. For more information, see Asynchronous Programming with Async and Await.

The following example demonstrates how to use two FileStream objects to copy files asynchronously from one directory to another. Notice that the Click event handler for the Button control is marked with the `async` modifier because it calls an asynchronous method.

C#

```
using System;
using System.Threading.Tasks;
using System.Windows;
using System.IO;

namespace WpfApplication
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }

        private async void Button_Click(object sender, RoutedEventArgs e)
        {
            string StartDirectory = @"c:\Users\exampleuser\start";
            string EndDirectory = @"c:\Users\exampleuser\end";

            foreach (string filename in Directory.EnumerateFiles(StartDirectory))
            {
                using (FileStream SourceStream = File.Open(filename, FileMode.Open))
                {
```

```
                    using (FileStream DestinationStream = File.Create(EndDirectory +
filename.Substring(filename.LastIndexOf('\\'))))
                    {
                        await SourceStream.CopyToAsync(DestinationStream);
                    }
                }
            }
        }
    }
}
```

The next example is similar to the previous one but uses StreamReader and StreamWriter objects to read and write the contents of a text file asynchronously.

C#

```
        private async void Button_Click(object sender, RoutedEventArgs e)
        {
            string UserDirectory = @"c:\Users\exampleuser\";

            using (StreamReader SourceReader = File.OpenText(UserDirectory + "BigFile.txt"))
            {
                using (StreamWriter DestinationWriter = File.CreateText(UserDirectory +
"CopiedFile.txt"))
                {
                    await CopyFilesAsync(SourceReader, DestinationWriter);
                }
            }
        }

        public async Task CopyFilesAsync(StreamReader Source, StreamWriter Destination)
        {
            char[] buffer = new char[0x1000];
            int numRead;
            while ((numRead = await Source.ReadAsync(buffer, 0, buffer.Length)) != 0)
            {
                await Destination.WriteAsync(buffer, 0, numRead);
            }
        }
```

The next example shows the code-behind file and the XAML file that are used to open a file as a Stream in a Windows 8.x Store app, and read its contents by using an instance of the StreamReader class. It uses asynchronous methods to open the file as a stream and to read its contents.

C#

```
using System;
using System.IO;
using System.Text;
using Windows.Storage.Pickers;
using Windows.Storage;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;

namespace ExampleApplication
```

```
{
    public sealed partial class BlankPage : Page
    {
        public BlankPage()
        {
            this.InitializeComponent();
        }

        private async void Button_Click_1(object sender, RoutedEventArgs e)
        {
            StringBuilder contents = new StringBuilder();
            string nextLine;
            int lineCounter = 1;

            var openPicker = new FileOpenPicker();
            openPicker.SuggestedStartLocation = PickerLocationId.DocumentsLibrary;
            openPicker.FileTypeFilter.Add(".txt");
            StorageFile selectedFile = await openPicker.PickSingleFileAsync();

            using (StreamReader reader = new StreamReader(await
selectedFile.OpenStreamForReadAsync()))
            {
                while ((nextLine = await reader.ReadLineAsync()) != null)
                {
                    contents.AppendFormat("{0}. ", lineCounter);
                    contents.Append(nextLine);
                    contents.AppendLine();
                    lineCounter++;
                    if (lineCounter > 3)
                    {
                        contents.AppendLine("Only first 3 lines shown.");
                        break;
                    }
                }
            }
            DisplayContentsBlock.Text = contents.ToString();
        }
    }
}
```

XAML

```
<Page
    x:Class="ExampleApplication.BlankPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:ExampleApplication"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">

    <StackPanel Background="{StaticResource ApplicationPageBackgroundBrush}"
VerticalAlignment="Center" HorizontalAlignment="Center">
        <TextBlock Text="Display lines from a file."></TextBlock>
        <Button Content="Load File" Click="Button_Click_1"></Button>
        <TextBlock Name="DisplayContentsBlock"></TextBlock>
    </StackPanel>
</Page>
```

See Also

Stream
File and Stream I-O
Asynchronous Programming with Async and Await