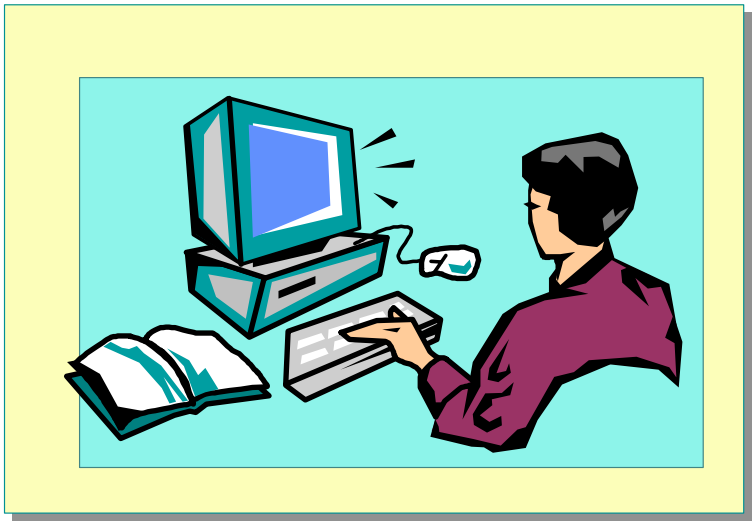
Práctica A: Archivos

Objetivo
Presentar el laboratorio.

Presentación
En este laboratorio, creará una aplicación que leerá y escribirá caracteres a y desde archivos. También creará una aplicación que puede utilizar los objetos StringReader y StreamReader para leer datos de caracteres de archivos o cadenas.



Objetivos

En este laboratorio, estudiaremos cómo:

- Crear una aplicación que lee y escribe caracteres a/desde archivos.
- Crear una aplicación que utiliza los objetos **StringReader** y **StreamReader** para leer datos de caracteres de archivos o cadenas.

Configuración del laboratorio

Hay archivos de inicio y de solución asociados a este laboratorio. Los archivos de inicio se encuentran en la carpeta Starter y los archivos de solución se encuentran en la carpeta Solution. Estas dos carpetas se pueden encontrar dentro del fichero labs09.zip

Escenario

En este laboratorio, se proporciona como punto de partida una aplicación de consola de Visual Studio® .NET. La aplicación, denominada *Files*, abre uno o más archivos especificados en la línea de comandos y cuenta los bytes, caracteres, palabras y líneas de cada archivo. El resultado para cada archivo y los totales para todos los archivos se muestran en la consola.

La aplicación soporta un parámetro **–f** para permitir que la salida por pantalla sea redirigida a un archivo, y un parámetro **–t** para ejecutar un modo de prueba especial de operación donde se obtiene la entrada de una cadena de prueba previamente codificada, en lugar de archivos especificados por el usuario. La aplicación está basada en una versión ligeramente modificada del ejemplo *Word Count* del SDK del .NET Framework.

Tiempo estimado para completar este laboratorio: 45 minutos

Ejercicio 1
Leer y escribir archivos y cadenas

En este ejercicio, modificará la aplicación *Files* para extraer un archivo específico.

↙ Examinar la aplicación

1. En Visual Studio .NET, abra el proyecto **Files**, que se encuentra en la carpeta Starter\Files dentro del fichero labs09.zip.
2. Abra el archivo WordCount.vb, y examine el código. Preste especial atención a los dos métodos que modificará: el método **Main** de la clase **App**, y el método **CountStats** de la clase **WordCounter**.
3. Genere la aplicación **Files**.
4. Establezca los argumentos de la línea de comandos de la aplicación mediante los siguientes pasos:
   a. Si el panel del Explorador de soluciones de la ventana Visual Studio .NET no está visible, en el menú **Ver**, haga clic en **Explorador de soluciones**.
   b. En el panel del Explorador de soluciones, haga clic con el botón derecho en **Files** para mostrar el menú de acceso directo, y haga clic en **Propiedades**.
   c. En el cuadro de diálogo **Páginas de propiedades de Files**, haga clic en la carpeta **Propiedades de configuración** y clic en **Depuración**.
   d. En el panel derecho debajo de **Opciones de inicio**, establezca los argumentos de la línea de comandos como sigue:

      ```
      -a -o -t -foutput.txt test.txt
      ```

   e. Haga clic en **Aceptar** para cerrar el cuadro de diálogo **Páginas de propiedades de Files**.
5. En el menú **Depurar**, haga clic en **Iniciar sin depurar**. En la ventana de la consola, debería ver el siguiente texto:

   ```
   Replace this Console.WriteLine in App's Main↪
    method with code as per the lab
   Lines   Words   Chars   Bytes   Pathname
   Replace this Console.WriteLine in WordCounter's↪
    CountStats method
       0       0       0       0   Test String
   -----   -----   -----   -----   --------------------
       0       0       0       0   Total in all files
   Word usage sorted alphabetically (0 unique words)
   Word usage sorted by occurrence (0 unique words)
   ```

6. En la ventana de la consola, pulse cualquier tecla para detener la depuración.
7. En el código del método **Main** de la clase **App**, localice la siguiente línea de código:

```
If Not(ap.OutputFile Is Nothing) Then
```

Si ap.OutputFile no contiene **Nothing**, contiene el nombre del archivo de salida especificado en el parámetro de línea de comandos **–f**.

8. Reemplace la siguiente invocación **Console.WriteLine** de la línea con código que:

   a. Establezca **fsOut** a un objeto **FileStream** que cree un archivo con el nombre especificado con acceso de escritura y sin compartición de archivos.

   b. Establezca **sw** a un objeto **StreamWriter** que esté enlazado al objeto **FileStream** creado en el paso (a).

   c. Redirija la salida de la consola al archivo asociando el objeto **StreamWriter** a la consola mediante el siguiente comando:

   ```
   Console.SetOut(sw)
   ```

9. Regenere y ejecute la aplicación haciendo clic en **Iniciar** en el menú **Depurar**, y examine el archivo de salida en función de las opciones especificadas con los parámetros en la línea de comandos.

   El archivo debería encontrarse en el subdirectorio *bin*. Debería contener el siguiente texto:

```
Lines  Words   Chars   Bytes   Pathname
Replace this Console.WriteLine in WordCounter's➥
 CountStats method
    0      0       0       0   Test String
-----  -----   -----   -----   ---------------------
    0      0       0       0   Total in all files
Word usage sorted alphabetically (0 unique words)
Word usage sorted by occurrence (0 unique words)
```

↙ Agregar el proceso CountStats en modo de prueba y en modo normal

1. Localice las siguientes líneas en el código del método **CountStats** de la clase **WordCounter**:

```
Dim Ok As Boolean = True
numLines = 0 : numWords = 0 : numChars = 0 : numBytes = 0
Try
```

Si se ha establecido la opción **–t**, el llamador de **CountStats** tendrá establecido el parámetro de nombre de ruta a una cadena vacía, que se denomina **String.Empty**.

2. Sustituya la siguiente llamada **Console.WriteLine** por código que:

   a. Declare una variable de tipo **TextReader** y de nombre **tr**.

   Utilizamos el tipo **TextReader** porque es el tipo básico común para **StringReader** y **StreamReader**. El polimorfismo permitirá proveer un objeto **StringReader** o **StreamReader** al código que utilice un objeto **TextReader**.

   b. Si el nombre de ruta está vacío:

      i. Cree un **StringReader** denominado **sr**, que esté enlazado a un miembro de **WordCounter** denominado **testString**.

ii. Establezca el número de bytes de esta cadena en **numBytes**.

iii. Establezca **sr** en **tr**.

Esta asignación realiza un *cast* implícito de un **StringReader** a un **TextReader**.

c. Si el nombre de ruta no está vacío:

i. Cree un **FileStream** denominado **fsIn**, abriendo el archivo especificado en el nombre de ruta del parámetro. Abra con acceso a lectura y permita acceso a lectura compartido.

ii. Establezca el número de bytes de **fsIn** en **numBytes**.

iii. Cree un **StreamReader** denominado **sr** que esté enlazado a este archivo.

iv. Establezca **sr** en **tr**.

Esta asignación realiza un cast implícito de un **StreamReader** en un **TextReader**.

d. En este módulo, quite el comentario de la declaración de la cadena y el loop **while** a continuación del comentario:

```
' Process every line in the file
```

e. Siguiendo el bucle **while**, agregue código para cerrar el objeto **TextReader**.

3. Regenere y ejecute la aplicación, y examine el archivo de salida del subdirectorio *bin*. Debería contener el siguiente texto:

```
Lines  Words   Chars   Bytes   Pathname
    2      3      16      18   Test String
-----  -----   -----   -----   ----------------------
    2      3      16      18   Total in all files
Word usage sorted alphabetically (2 unique words)
    2: hello
    1: world
Word usage sorted by occurrence (2 unique words)
    1: world
    2: hello
```

4. Elimine los parámetros de prueba de las opciones de la línea de comandos que Visual Studio .NET utilizará para ejecutar la aplicación, de modo que las opciones de la línea de comando estén establecidas en:

```
-a -o -foutput.txt test.txt
```

5. Ejecute la aplicación y examine el archivo de salida del subdirectorio *bin*. Debería contener el siguiente texto:

```
Lines  Words   Chars   Bytes   Pathname
    5     16      65      73   ...\test.txt
-----  -----   -----   -----   --------------------
    5     16      65      73   Total in all files
Word usage sorted alphabetically (14 unique words)
    1: aid
    1: all
    1: come
    1: country
    1: for
    1: good
    1: is
    1: men
    1: now
    1: of
    2: the
    1: their
    1: time
    2: to
Word usage sorted by occurrence (14 unique words)
    1: aid
    1: all
    1: come
    1: country
    1: for
    1: good
    1: is
    1: men
    1: now
    1: of
    1: their
    1: time
    2: the
    2: to
```

6. Examine el archivo **Test.txt** del subdirectorio *bin* y verifique que la salida es la prevista.