Tutorial: Manipular archivos y directorios en C#

Este tutorial ofrece una introducción a los fundamentos de la E/S de archivos en C#. Para ilustrar las características, crea una pequeña aplicación, denominada FileExplorer, que examina archivos de texto en un directorio y proporciona información tal como los atributos, hora del último acceso. También incluye una opción que escribe la información en un archivo de registro.

Crear la aplicación

Para iniciar el proyecto, cree un formulario en el que los usuarios puedan seleccionar un directorio, un archivo del directorio y, a continuación, la información sobre el archivo que desean recuperar.

Para crear el proyecto

1. En el menú Archivo, haga clic en Nuevoproyecto.

   Aparecerá el cuadro de diálogo Nuevo proyecto.

2. En el panel Tipos de proyecto, haga clic en Proyectos de C# y, a continuación, elija Aplicación para Windows en el panel Plantillas.
3. En el cuadro Nombre, escriba `Tut_NombreAlumno` como nombre del proyecto.

   Visual Studio agregará el proyecto al Explorador de soluciones y se abrirá el Diseñador de Windows Forms.

4. Agregue los controles de la siguiente tabla al formulario y establezca los correspondientes valores para sus propiedades.

Objeto	Propiedades	Valor
TextBox	Name Text	**txtDirectory** **Directorio**
Button	Name Text	**btnSubmit** **Enviar**
Button	Name Text	**btnExamine** **Examinar**
ComboBox	Name Text	**lstFilePick** **Seleccione un archivo**
CheckBox	Name Text Checked	**chkFileLength** **Longitud del archivo** **True**
CheckBox	Name	**chkLastAccess**

	Text Checked	**Hora del último acceso** **True**
CheckBox	Name Text Checked	**chkSave** **Guardar resultados** **False**

Mostrar el directorio actual

La aplicación necesita un punto de inicio. En consecuencia, el control `txtDirectory` TextBox utiliza la función `Directory.GetCurrentDirectory()`para devolver y mostrar una cadena que representa la ruta de acceso actual.

Para devolver el directorio actual

1. Haga doble clic en el formulario para crear un controlador de eventos para `Form1_Load`.

   Se abrirá el Editor de código.

2. Agregue el código siguiente para que el control `txtDirectory` TextBox muestre la ubicación actual.

   ```
   txtDirectory.Text = Directory.GetCurrentDirectory()
   ```

3. Ejecute el programa para comprobar que se devuelve la ruta de acceso correcta.

   El control `txtDirectory`TextBox muestra el directorio actual.

Prueba la aplicación.

Cambiar directorios

Dado que es posible que un usuario desee seleccionar archivos de un directorio diferente, la aplicación utiliza la misma propiedad para cambiar de directorio. Para cambiar a un directorio diferente, el usuario escribe una nueva ruta de acceso en el control `txtDirectory` TextBox.

Para cambiar de directorio

1. Haga doble clic en el control del formulario para crear un controlador de eventos clic para `btnSubmit`.

   Se abrirá el Editor de código.

2. Agregue el código siguiente al controlador de eventos clic.

   ```
   String NewPath =" NewPath holds the path the user has entered.";

   NewPath = txtDirectory.Text;
   // Change the location to NewPath.
   ```

```
Directory.SetCurrentDirectory( NewPath);
```

Comprobar si se escribió una ruta de acceso válida

Utilice una instrucción **Try...Catch** para detectar excepciones que surgen de escribir una ruta de acceso en blanco o no válida.

Para garantizar rutas de acceso válidas

1. En el evento `btnSubmit_Click`, , agregue `string ErrorMessage=null;` en una nueva línea.
2. Antes de la línea de código `Directory.SetCurrentDirectory( NewPath);`, agregue una instrucción **Try**

```
try {
         // Change the location to NewPath.
         Directory.SetCurrentDirectory( NewPath);
}
```

3. Agregue lo siguiente:

```
           catch (DirectoryNotFoundException f){ //' This catches errors caused by a path
that is not valid.
               ErrorMessage = "You must enter a valid path. If trying to access a different
drive, remember to include the drive letter.";


            }
            catch {     //' This checks to make sure the path is not blank.

                ErrorMessage = "You must enter a path.";
            }
             finally{
            //' Display the error message only if one exists.
                    if (ErrorMessage != null)
                        MessageBox .Show(ErrorMessage);

              }
```

Prueba la aplicación.

Mostrar el contenido del directorio en un control ComboBox

Para permitir que la aplicación muestre el contenido del directorio actual, puede utilizar el método `Directory.EnumerateFiles(sourceDirectory); y EnumerateDirectories (sourceDirectory)`, que devuelve una colección de cadenas que representan los nombres de los archivos en el directorio.

Para mostrar el contenido del directorio

1. En el evento `btnSubmit_Click`, inserte lo siguiente.

```
                string sourceDirectory = NewPath;
```

```
            var txtFiles = Directory.EnumerateFiles(sourceDirectory);

            foreach (string currentFile in txtFiles)
            {
                string fileName = currentFile.Substring(sourceDirectory.Length + 1);
                lstFilePick.Items.Add(currentFile);

            }
```

La información recopilada aparece en el control `lstFilePick`ComboBox, en el que podrá seleccionar un archivo específico para examinarlo.

`Añáde también los directorios al combo`

Prueba la aplicación

Permitir que el usuario seleccione un archivo para examinarlo

Aunque el control `lstFilePick` ComboBox muestra todos los archivos de un directorio, es probable que el usuario desee seleccionar y examinar un archivo específico.

Para habilitar la selección de un archivo concreto

- Cree un controlador de evento clic para `btnExamine_Click` y agregue el código siguiente para confirmar la selección de un archivo.

  ```
  System.IO.FileInfo thisFile=new FileInfo (lstFilePick.SelectedItem.ToString() ) ;
  ```

  En thisFile tendremos el fichero que queremos examinar.

Permitir que el usuario determine la información que desea recopilar

Una vez que se muestran los archivos en el control `lstFilePick` ComboBox, el código adicional permite que el usuario especifique la información de la que se informa. Por ejemplo, es posible que un usuario sólo desee saber la fecha en la que se tuvo acceso al archivo por última vez. Es posible que otro usuario desee conocer también el tamaño de un archivo. Los usuarios pueden activar o desactivar las casillas de verificación (`chkLastAccess`, `chkFileLength`) para personalizar los resultados.

Para mostrar información específica

1. Declare estas variables al principio del evento `btnExamine_Click` después de `(lstFilePick.SelectedItem)`:

```
            string texto = "Los atributos del fichero "+lstFilePick.SelectedItem.ToString() +
            " son : ";
              if (chkFileLength.Checked)
              {
                  texto = texto + " La Longitud del fichero es : " +
thisFile.Length.ToString();
              }

              if (chkLastAccess.Checked)
              {
```

```
                texto = texto + " última modificación fue : " +
thisFile.LastAccessTime.ToString();
            }
```

El método `thisFile.Length.ToString()` devuelve el atributo longitud
`thisFile.LastAccessTime.ToString()`y de vuelve la última vez que se modificó el fichero .

Mostrar los resultados

Para completar la funcionalidad de la aplicación, un **MsgBox** informa de la información recopilada.

```
MessageBox.Show(texto);
```

Guardar los resultados

Es posible que el usuario desee guardar los resultados de examinar un archivo. En consecuencia, debe agregar código que compruebe si existe un archivo de registro, cree uno si es necesario y, a continuación, escriba los resultados en el archivo de registro.

Para crear un archivo de registro:

- Agregue lo siguiente al final del evento `btnExamine_Click`.

```
if (chkSave.Checked)
            {
                 // si no existe el fichero lo crea y si existe sobreescribe el contenido

                System.IO.File.WriteAllText(NewPath + "/log.txt", texto);

            }
Prueba la aplicación
```