Chương 5: File Operations

Tổng quan

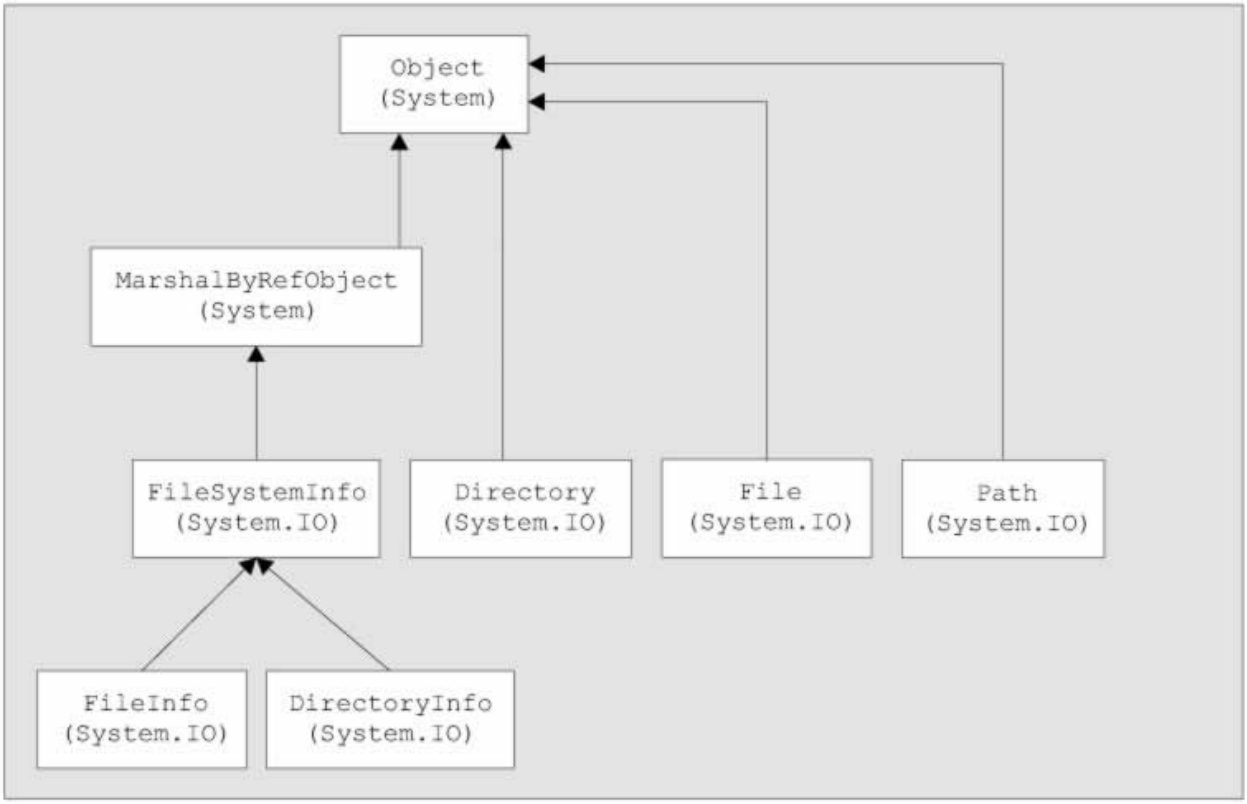
Trong chương này chúng ta sẽ khảo sát làm thế nào để thực hiện các nhiệm vụ như đọc từ file và viết ra file và hệ thống đăng ký (Registry) trong C#. Cụ thể chúng ta sẽ được học.

- Cấu trúc thư mục, tìm kiếm file và folder hiện diện và kiểm tra thuộc tính của chúng.
- Di chuyển, sao chép, huỷ các file và folder
- Đọc và ghi văn bản trong các file
- Đọc và ghi các khoá trong Registry

Microsoft cung cấp rất nhiều mô hình đối tượng trực giác mà chúng ta sẽ được khảo sát trong chương này, chúng ta cũng biết cách sử dụng các lớp cơ bản của .NET để thực hiện các nhiệm vụ được đề cập ở trên.

5.1 Quản lý tập tin hệ thống

Các lớp dưới đây được sử dụng để duyệt qua các file hệ thống và cách thức thực hiện như di chuyển, sao chép, huỷ các file được hiển thị trên sơ đồ. Namespace của mỗi lớp đặt trong ngoặc trong sơ đồ.



Mục đích của các lớp trình bày dưới đây:

- System.MarshalByRefObject – Lớp đối tượng cơ sở cho các lớp của .NET nó điều khiển từ xa; cho phép điều hành dữ liệu giữa các vùng ứng dụng.
- FileSystemInfo – Lớp đối tượng cơ sở biểu diễn các file đối tượng hệ thống
- FileInfo and File – Các lớp này thể hiện một file trên file hệ thống
- DirectoryInfo and Directory – Các lớp này thể hiện một folder trên tập tin hệ thống
- Path – Lớp này chứa các bộ phận tĩnh dùng chế tác các pathnames

5.1.1 Các lớp .NET thể hiện các File và Folder

Trước khi xem làm thế nào bạn có thể lấy dữ liệu từ các tập tin hoặc viết dữ liệu lên tập tin, .NET hỗ trợ thế nào những thao tác liên quan đến tập tin và thư mục. Trên

.NET Framework, namespace System.IO là vùng của các thư viện lớp dành cho những dịch vụ liên quan đến xuất nhập dữ liệu dựa trên tập tin.

Ở hình trên ta thấy System.IO cung cấp cho bạn 4 lớp (Directory, File, DirectoryInfo, FileInfo) cho phép bạn thao tác với tập tin riêng lẽ cũng như tương tác với cấu trúc thư mục của máy. Hai lớp đầu Directory và File cho phép những thao tác tạo, gỡ bỏ và thao tác khác nhau trên các tập tin và thư mục. Hai lớp có liên hệ mật thiết DirectoryInfo và FileInfo cũng có những chức năng tương tự nhưng các thành viên không phải là static, nên muốn triệu gọi các thành viên trước tiên bạn phải tạo một thể hiện đối tượng lớp.Hai lớp đầu Directory và File dẫn xuất từ lớp System.Object trong khi DirectoryInfo và FileInfo dẫn xuất từ lớp FileSystemInfo. Lớp FileSystemInfo là lớp cơ bản abstract có một số thuộc tính và phương thức cung cấp những thông tin liên quan đến một tập tin và thư mục.

5.1.1.1 Các thuộc tính của lớp cơ bản FileSystemInfo:

Name	**Description**
CreationTime	Thời gian file, folder được tạo
DirectoryName (FileInfo), Parent (DirectoryInfo)	Tên đường dẫn của folder chức đựng
Exists	Xác định file ,folder hiện hữu
Extension	Tên mở rộng của file, trả về khoảng trắng nếu là folder
FullName	Tên đường dẫn của file ,folder
LastAccessTime	Thời gian file, folder truy xuất làn cuối

Name	Description
LastWriteTime	Time file or folder was last modified
Name	Name of the file or folder
Root	Đường dẫn gốc
Length	Kích thước file tính bằng bytes (chỉ FileInfo)

Các phương thức bạn có thể thực hiện như sau:

Name	Purpose
Create()	Tạo một folder hoặc một file rỗng
Delete()	Huỷ file, folder
MoveTo()	Di chuyển hoặc sửa tên file, folder.
CopyTo()	(FileInfo only) Sao chép file, không sao chép fương thức cho folders.
GetDirectories()	(DirectoryInfo only) Trả về một mảng các đối tượng của DirectoryInfo đại diện tát cả folders được chứa trong folder này.
GetFiles()	(DirectoryInfo only) Trả về một mảng các đối tượng của FileInfo đại diện tát cả folders được chứa trong folder này

Name	Purpose
GetFileSystemObjects()	(DirectoryInfo only) Trả về đối trương FileInfo và DirectoryInfo như mảng của tham khảoFileSystemInfo .

5.1.1.2 Tạo một đối tượng DirectoryInfo

Bạn bắt đầu làm việc với lớp DirectoryInfo bằng cách khai báo một đường dẫn cụ thể ví dụ : "C:\", "D:\WINNT", . . .nếu bạn muốn truy cập thư mục của ứng dụng đang thi hành bạn dùng ký hiệu "." thí dụ :

```
//Tạo một thư mục mới từ thư mục hiện hành trở đi
DirectoryInfo dirl = new DirectoryInfo(".");

//Tạo mộtthư mục mới từ thư mục C:\Foo\Bar trở đi
DirectoryInfo dir2 = new DirectoryInfo(@"C:\Foo\Bar");
```

5.1.2 Lớp Path

Lớp Path không là một lớp mà bạn khai báo cụ thể, Đúng ra, nó trình bày các phương thức tỉnh để thực hiện các phép toán trên tên đường dẫn dễ dàng hơn. ví dụ bạn muốn hiển thị tên đường dẫn cho một file ReadMe.txt trong folder C:\My Documents. Bạn có thể tìm đường dẫn đến file như sau:

```
Console.WriteLine(Path.Combine(@"C:\My Documents", "ReadMe.txt"));
```

Sử dụng lớp Path dễ dàng hơn nhiều so với khi bạn thực hiện các ký hiệu bằng tay nhất là bởi vì lớp Path nhận biết được các định dạng khác nhau của đường dẫn trên các hệ điều dành khác nhau. Tại lúc viết, Windows chỉ hỗ trợ hệ điều hành được hỗ trợ bởi

.NET nhưng nếu .NET sử dụng trên Unix, Path sẽ dùng / thay cho \ làm dấu ngăn cách tên đường dẫn. Path.Combine() là phương thức mà lớp này thường hay sử dụng nhưng Path cũng thực hiện những phương thức khác để cung cấp thông tin về đường dẫn hoặc yêu cầu định dạng của nó.

5.1.3 Thí dụ : A File Browser

Phần này trình bày ví dụ hướng dẫn làm thế nào để trình duyệt thư mục và hiển thị thuộc tính của file:

Một thí dụ ứng dụng C# gọi FileProperties, nó trình bày một giao diện người dùng đơn giản cho phép bạn trình duyệt các file hệ thống và hiển thị thời gian tạo thành, thời gian truy xuất làn cuối, kích thước file.

Ứng dụng FileProperties nhìn như sau. Bạn gõ tên của folder hoặc tên file trong textbox chính ở phía trên của cửa sổ và nhấn vào nút Display. Nội dung của folder được tạo trên listbox. Màn hình sẽ hiển thị thuộc tính file đang được dùng xem xét một folder:

Nếu bạn chỉ muốn hiển thị creation time, last access time, and last modification time cho folders – DirectoryInfo thực thi các thuộc tính thích hợp, chúng ta chỉ chọn vào thuộc tính của chúng.

Ta tạo một dự án như sau standard C# Windows application trong Visual Studio.NET, và thêm vào các textboxes và listbox từ Windows Forms area của toolbox. Chúng ta cũng đổi tên các điều khiển một cách trực giác như textBoxInput, textBoxFolder, buttonDisplay, buttonUp, listBoxFiles, listBoxFolders, textBoxFileName, txtBoxCreationTime, textBoxLastAccessTime, textBoxLastWriteTime, và txtBoxFileSize.

Sau đó chúng ta thêm vào đoạn code. Đầu tiên ta cần khai báo sử dụng System.IO namespace:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.IO;
```

Chúng ta thêm trường thành viên vào main form:

```
public class Form1 : System.Windows.Forms.Form
{
   private string currentFolderPath;
```

currentFolderPath sẽ giữ dường dẫn của thư mục và nội dung sẽ hiển thị trên listboxes.

Chúng ta cần thêm một số sự kiện ngõ ra như sau:

- User nhấn nút Display: Trong trường hợp này ta cần có nơi ngõ ra folder. Nếu là một folder ta đưa ra danh sách file và thư mục con, Nếu nó là file, chúng ta cho hiển thị thuộc tính ra textboxes.
- User nhấn lên file trên listbox: chúng ta chỉ cho hiển thị thuộc tính của file trên textboxes.
- User nhấn trên một folder trên Folders listbox: chúng ta xoá tất cả các điều khiển và hiển thị nội dung của thư mục con trên listboxes.
- User nhấn trên nút Up : chúng ta xoá tất cả điều khiển và hiển thị thư mục cha trên listboxes.

Trước tiên ta lên danh sách các sự kiện, và làm rỗng nội dung của các điều khiển:

```
protected void ClearAllFields()
{
  listBoxFolders.Items.Clear();
  listBoxFiles.Items.Clear();
  textBoxFolder.Text = "";
  textBoxFileName.Text = "";
  textBoxCreationTime.Text = "";
  textBoxLastAccessTime.Text = "";
  textBoxLastWriteTime.Text = "";
  textBoxFileSize.Text = "";
}
```

thứ hai ta định nghĩa một fương thức, DisplayFileInfo(), nó thực hiện tiến trình hiển thị thông tin lên textboxes. Phương thức này lấy một thông số, tên đường dẫn file, và làm việc bởi một đối tượng FileInfo dựa trên path này:

```
protected void DisplayFileInfo(string fileFullName)
{
  FileInfo theFile = new FileInfo(fileFullName);
  if (!theFile.Exists)
     throw new FileNotFoundException("File not found: " + fileFullName);
  textBoxFileName.Text = theFile.Name;
  textBoxCreationTime.Text = theFile.CreationTime.ToLongTimeString();
```

```
  textBoxLastAccessTime.Text = theFile.LastAccessTime.ToLongDateString();
  textBoxLastWriteTime.Text = theFile.LastWriteTime.ToLongDateString();
  textBoxFileSize.Text = theFile.Length.ToString() + " bytes";
}
```

Ta tạo phương thức DisplayFolderList(), Nó hiển thị nội dung được cho bởi folder trong hai listboxes.

```
protected void DisplayFolderList(string folderFullName)
{
  DirectoryInfo theFolder = new DirectoryInfo(folderFullName);
  if (!theFolder.Exists)
    throw new DirectoryNotFoundException("Folder not found: "
                                + folderFullName);
  ClearAllFields();
  textBoxFolder.Text = theFolder.FullName;
  currentFolderPath = theFolder.FullName;

  // list all subfolders in folder
  foreach(DirectoryInfo nextFolder in theFolder.GetDirectories())
    listBoxFolders.Items.Add(nextFolder.Name);

  // list all files in folder
  foreach(FileInfo nextFile in theFolder.GetFiles())
```

```
    listBoxFiles.Items.Add(nextFile.Name);
}
protected void OnDisplayButtonClick(object sender, EventArgs e)
{
  try
  {
    string folderPath = textBoxInput.Text;
    DirectoryInfo theFolder = new DirectoryInfo(folderPath);
    if (theFolder.Exists)
    {
      DisplayFolderList(theFolder.FullName);
      return;
    }
    FileInfo theFile = new FileInfo(folderPath);
    if (theFile.Exists)
    {
      DisplayFolderList(theFile.Directory.FullName);
      int index = listBoxFiles.Items.IndexOf(theFile.Name);
      listBoxFiles.SetSelected(index, true);
      return;
    }
    throw new FileNotFoundException("There is no file or folder with "
                                    + "this name: " + textBoxInput.Text);
```

```
  }
  catch(Exception ex)
  {
    MessageBox.Show(ex.Message);
  }
}
```

Trong đoạn code trên chúng ta thành lập nếu văn bản cung cấp trình bày một folder hoặc một file bởi thể hiện DirectoryInfo và FileInfo instances and khảo sát thuộc tính có sẵn của mỗi đối tượng Nếu không có sẵn chúng ta đưa ra ngoại lệ.

```
protected void OnListBoxFilesSelected(object sender, EventArgs e)
{
  try
  {
    string selectedString = listBoxFiles.SelectedItem.ToString();
    string fullFileName = Path.Combine(currentFolderPath, selectedString);
    DisplayFileInfo(fullFileName);
  }
  catch(Exception ex)
  {
    MessageBox.Show(ex.Message);
  }
}
```

Bộ quản lý sự kiện cho chọn lựa của folder trong Folders listbox được thi hành trong cách tương tự ngoại trừ trường hợp chúng ta gọi phương thức DisplayFolderList() để cập nhật nội dung của listboxes:

```
protected void OnListBoxFoldersSelected(object sender, EventArgs e)
{
  try
  {
    string selectedString = listBoxFolders.SelectedItem.ToString();
    string fullPathName = Path.Combine(currentFolderPath, selectedString);
    DisplayFolderList(fullPathName);
  }
  catch(Exception ex)
  {
    MessageBox.Show(ex.Message);
  }
}
```

Cuối cùng khi nút Up được nhấn , DisplayFolderList() phải cũng được gọi ngoại trừ lúc này chúng ta cần nhận được đường dẫn của cha thư mục hiện hành được hiển thị.

```
protected void OnUpButtonClick(object sender, EventArgs e)
{
  try
  {
    string folderPath = new FileInfo(currentFolderPath).DirectoryName;
```

```
    DisplayFolderList(folderPath);
  }
  catch(Exception ex)
  {
    MessageBox.Show(ex.Message);
  }
}
```

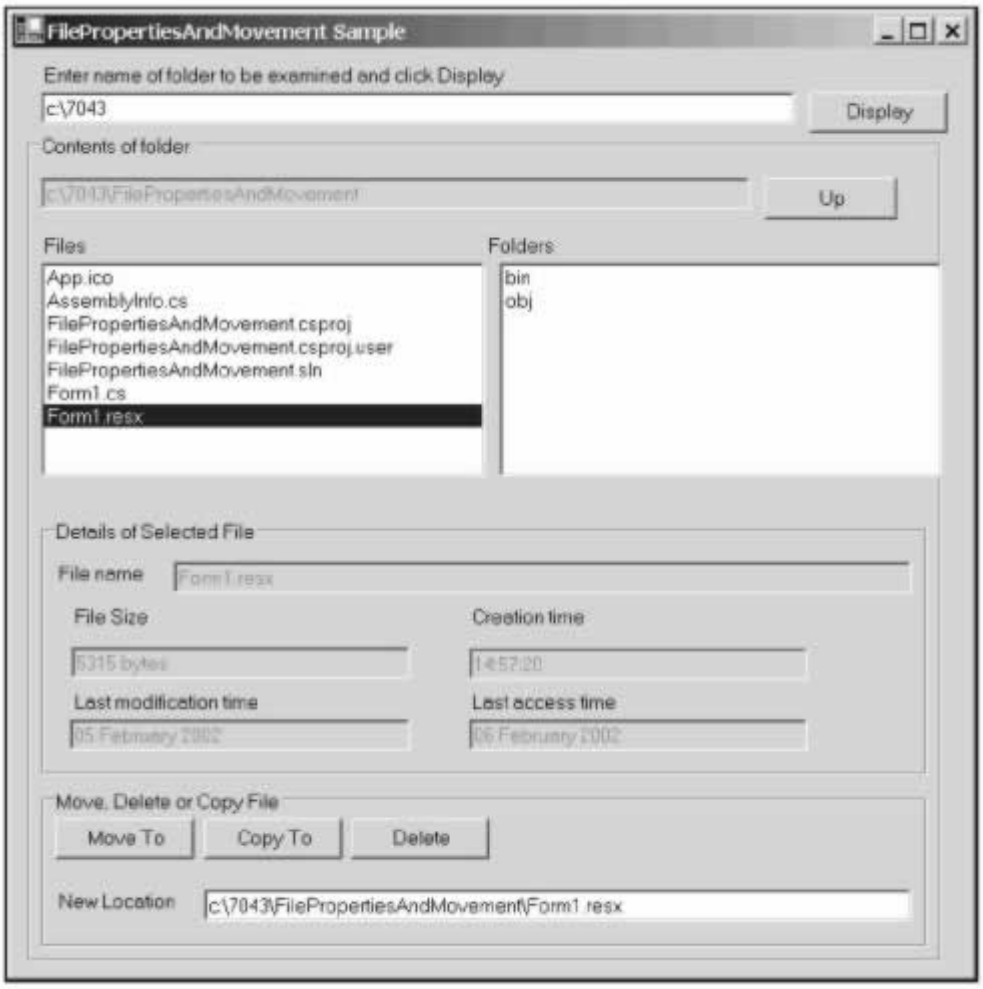
5.2 Di chuyển, Sao chép, Huỷ File

Chúng ta vừa mới đế cập di chuyển và huỷ files hoặc folders bằng phương thức MoveTo() và Delete() của lớp FileInfo và DirectoryInfo. Các phương thức tương đương nhau trên các lớp File và Directory là Move() và Delete(). Lớp FileInfo và File cũng có cách thức thực hiện tương tự, CopyTo() and Copy(). Không có fương thức nào copy các folder, tuy nhiên bạn có thể copy từng file trong folder.

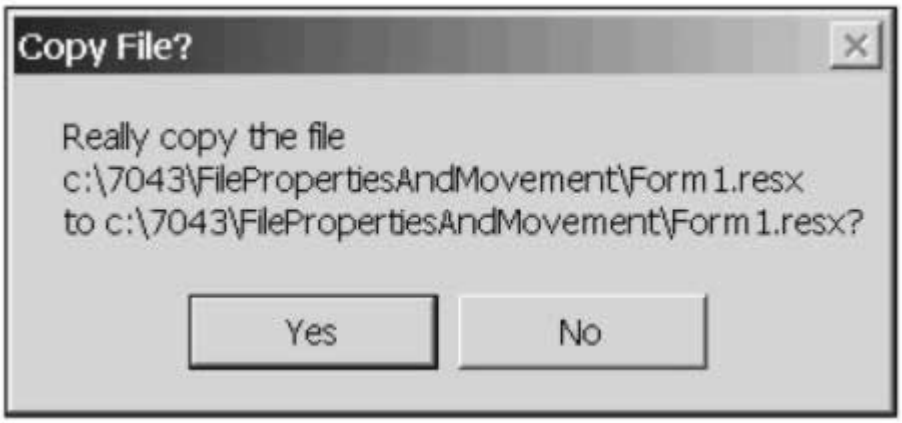
Tất cả các phương thức này đều hoàn toàn trực giác, bạn có thể tìm kiếm chi tiết trong help MSDN. Trong phần này chúng ta sẽ học cách gọi các phương thức tỉnh Move(), Copy(), và Delete() trong lớp File.Để thực hiện chúng ta dùng bài học phần trước FileProperties làm ví dụ, FilePropertiesAndMovement. Ví dụ này ta sẽ thêm tính năng mỗi lần thuộc tính của một file được hiển thị, ứng dụng này sẽ cho ta chọn lựa thêm xoá file hoặc di chuyển hoặc sao chép nó đến vị trí khác

5.2.1 Ví dụ: FilePropertiesAndMovement

Ví dụ mới như sau:



Từ hình trên chúng ta có thể thấy nó rất giống nhau trong lần xuất hiện ở ví dụ FileProperties, Ngoại trừ chúng có thêm một nhóm gồm ba nút button và một textbox tại phía dưới cửa xổ. Các điều khiển này chỉ hiện khi ví dụ hiển thị thuộc tính của một file- Lần khác chúng bị ẩn, Khi thuộc tính của file được hiển thị , FilePropertiesAndMovement tự động đặt tên đầy đủ đường dẫn của file đó ở cuối của xổ trong textbox. User có thể nhấn bất kỳ buttons để thực hiện phép toán thích hợp. Khi chương trình chạy một message box tương ứng được hiển thị xác nhận hành động.



Để mã hoá chúng ta cần thêm các điều khiển thích hợp, giống như thêm các sự kiện điều khiển cho ví dụ FileProperties . Chúng ta tạo các controls mới với tên buttonDelete, buttonCopyTo, buttonMoveTo, và textBoxNewPath.

Chúng ta sẽ thấy sự kiện điều kiện nhận được khi user nhấn vào Delete button;

```
protected void OnDeleteButtonClick(object sender, EventArgs e)
{
  try
  {
    string filePath = Path.Combine(currentFolderPath,
                          textBoxFileName.Text);
    string query = "Really delete the file\n" + filePath + "?";
    if (MessageBox.Show(query,
      "Delete File?", MessageBoxButtons.YesNo) == DialogResult.Yes)
    {
      File.Delete(filePath);
      DisplayFolderList(currentFolderPath);
```

```
    }
  }
  catch(Exception ex)
  {
    MessageBox.Show("Unable to delete file. The following exception"
                + " occurred:\n" + ex.Message, "Failed");
  }
}
```

Đoạn code thực hiện phương thức này sẽ có phần nắm bắt lỗi, thông báo sẽ lỗi không thể xoá được nếu file xoá đang thực hiện trên một tiến trình khác. Chúng ta xây dựng đường dẫn của file của file bị xoá từ trường CurrentParentPath , Nơi nó chứa dường dẫn thư mục cha, và tên file trong textBoxFileName textbox:

Phương thức di chuyển và sao chép file được cấu trúc :

```
protected void OnMoveButtonClick(object sender, EventArgs e)
{
  try
  {
    string filePath = Path.Combine(currentFolderPath,
                        textBoxFileName.Text);
    string query = "Really move the file\n" + filePath + "\nto "
                + textBoxNewPath.Text + "?";
    if (MessageBox.Show(query,
      "Move File?", MessageBoxButtons.YesNo) == DialogResult.Yes)
```

```
    {
      File.Move(filePath, textBoxNewPath.Text);
      DisplayFolderList(currentFolderPath);
    }
  }
  catch(Exception ex)
  {
    MessageBox.Show("Unable to move file. The following exception"
                + " occurred:\n" + ex.Message, "Failed");
  }
}

protected void OnCopyButtonClick(object sender, EventArgs e)
{
  try
  {
    string filePath = Path.Combine(currentFolderPath,
                          textBoxFileName.Text);
    string query = "Really copy the file\n" + filePath + "\nto "
                + textBoxNewPath.Text + "?";
    if (MessageBox.Show(query,
      "Copy File?", MessageBoxButtons.YesNo) == DialogResult.Yes)
    {
```

```
        File.Copy(filePath, textBoxNewPath.Text);
        DisplayFolderList(currentFolderPath);
      }
    }
    catch(Exception ex)
    {
      MessageBox.Show("Unable to copy file. The following exception"
                + " occurred:\n" + ex.Message, "Failed");
    }
}
```

Chúng ta cũng tạo buttons và textbox mới được đánh dấu enabled và disabled ở thời điểm thích hợp để enable chúng khi chúng ta hiển thị nội dung của file, chúng ta cần thêm đoạn code sau:

```
protected void DisplayFileInfo(string fileFullName)
{
  FileInfo theFile = new FileInfo(fileFullName);
  if (!theFile.Exists)
    throw new FileNotFoundException("File not found: " + fileFullName);

  textBoxFileName.Text = theFile.Name;
  textBoxCreationTime.Text = theFile.CreationTime.ToLongTimeString();
  textBoxLastAccessTime.Text = theFile.LastAccessTime.ToLongDateString();
  textBoxLastWriteTime.Text = theFile.LastWriteTime.ToLongDateString();
```

```
  textBoxFileSize.Text = theFile.Length.ToString() + " bytes";


  // enable move, copy, delete buttons
  textBoxNewPath.Text = theFile.FullName;
  textBoxNewPath.Enabled = true;
  buttonCopyTo.Enabled = true;
  buttonDelete.Enabled = true;
  buttonMoveTo.Enabled = true;
}
```

Chúng ta cũng cần thay đổi DisplayFolderList:

```
protected void DisplayFolderList(string folderFullName)
{
  DirectoryInfo theFolder = new DirectoryInfo(folderFullName);
  if (!theFolder.Exists)
    throw new DirectoryNotFoundException("Folder not found: " + folderFullName);


  ClearAllFields();
  DisableMoveFeatures();
  textBoxFolder.Text = theFolder.FullName;
  currentFolderPath = theFolder.FullName;


  // list all subfolders in folder
```

```
  foreach(DirectoryInfo nextFolder in theFolder.GetDirectories())
    listBoxFolders.Items.Add(NextFolder.Name);


  // list all files in folder
  foreach(FileInfo nextFile in theFolder.GetFiles())
    listBoxFiles.Items.Add(NextFile.Name);
}
```

DisableMoveFeatures là một hàm tiện ích nhỏ nó disables các controls mới:

```
    void DisableMoveFeatures()
    {
      textBoxNewPath.Text = "";
      textBoxNewPath.Enabled = false;
      buttonCopyTo.Enabled = false;
      buttonDelete.Enabled = false;
      buttonMoveTo.Enabled = false;
    }
```

Chúng ta cần thêm phương thức ClearAllFields() để xoá các textbox thêm vào:

```
    protected void ClearAllFields()
    {
      listBoxFolders.Items.Clear();
      listBoxFiles.Items.Clear();
      textBoxFolder.Text = "";
```

```
        textBoxFileName.Text = "";
        textBoxCreationTime.Text = "";
        textBoxLastAccessTime.Text = "";
        textBoxLastWriteTime.Text = "";
        textBoxFileSize.Text = "";
        textBoxNewPath.Text = "";
    }
```

5.3 Đọc và viết vào File

Đọc và viết vào files nói chung rất đơn giản; tuy nhiên , Điều này không phải bắt buộc biết các đối tượng DirectoryInfo hoặc FileInfo mà chúng ta vừa khảo sát.Thay vào đó chúng ta phải biết một số lớp trình bày nội dung chung gọi là **stream**, Điều này chúng ta sẽ khảo sát sau đây.

5.3.1 Streams

Đọc và viết dữ liệu sẽ được thực hiện thông qua lớp stream. Stream là dòng dữ liệu chảy đi. Đây là một thực thể (entity) có khả năng nhận được hoặc tạo ra một "nhúm" dữ liệu. System.IO.Stream là một lớp abstract định nghĩa một số thành viên chịu hỗ trợ việc đọc/viết đồng bộ (synchronus) hoặc không đồng bộ (asynchronous) đối với khối trữ tin (nghĩa là một tập tin trên đĩa hoặc tập tin trên ký ức).

Vì Stream là một lớp abstract, nên bạn chỉ có thể làm việc với những lớp được dẫn xuất từ Stream. Các hậu duệ của Stream tượng trưng dữ liệu như là một dòng dữ liệu thô dạng bytes (thay vì dữ liệu dạng văn bản). Ngoài ra, các lớp được dẫn xuất từ Stream hỗ trợ việc truy tìm (seek) nghĩa là một tiến trình nhận lấy và điều chỉnh vị trí trên một dòng chảy. Trước khi tìm hiểu những chức năng mà lớp Stream cung cấp, bạn nên xem qua các thành viên của lớp Stream.

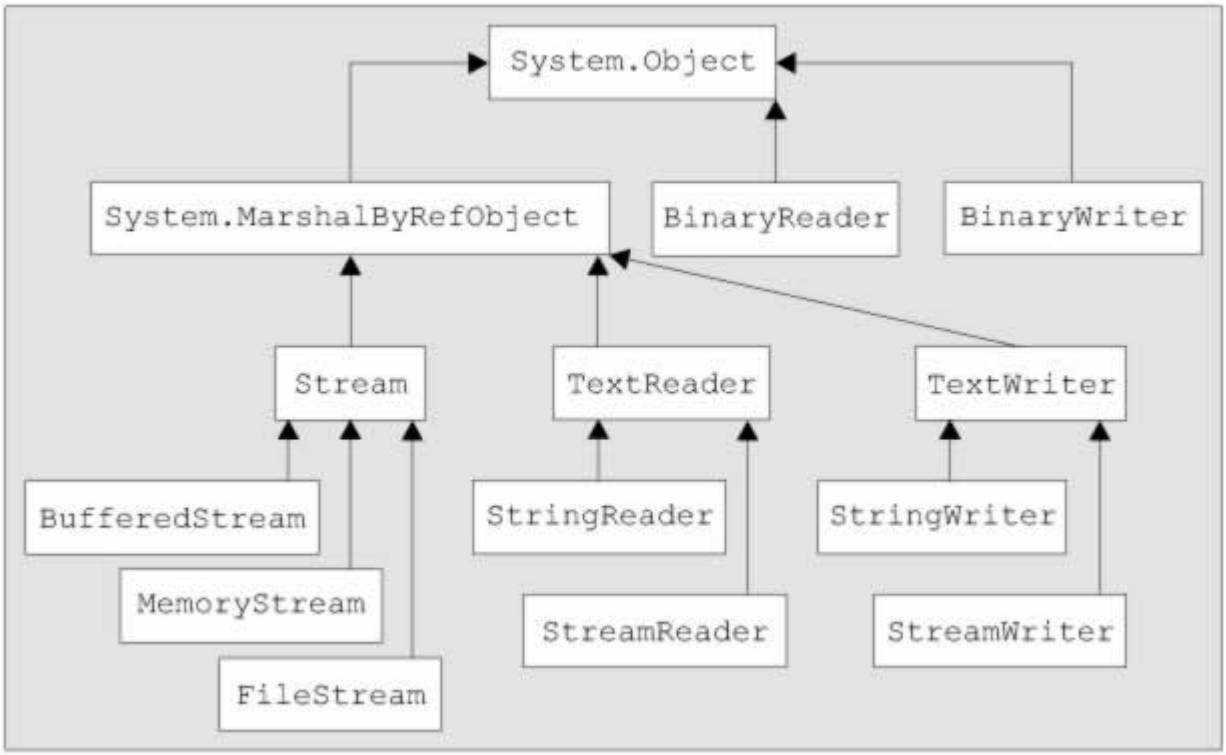
Ý tưởng của stream đã có từ lâu. Một stream là một đối tượng dùng để chuyển dữ liệu. Dữ liệu có thể được truyền theo hai hướng:

- Nếu dữ liệu được truyền từ nguồn bên ngoài vào trong chương trình của bạn, ta gọi là đọc dữ liệu
- Nếu dữ liệu được truyền từ chương trình của bạn ra nguồn bên ngoài , ta gọi là viết dữ liệu

Thường thì nguồn bên ngoài sẽ là một file, ngoài ra nó còn bao gồm cả trường hợp sau:

- Đọc hoặc ghi dữ liệu trên mạng dùng giao thức mạng
- Đọc hoặc ghi đến một đường ống chỉ định
- Đọc hoặc ghi đến một vùng của bộ nhớ

Các lớp có mối liên hệ trong namespace System.IO như hình sau:



5.3.2 Làm việc với Binary Files

Reading and writing to binary files thường được làm việc với lớp FileStream .

5.3.2.1 Làm việc với FileStream

Lớp FileStream đem lại việc thi công cho những thành viên của lớp abstract Stream theo một thể thức thích hợp đối với các file-base streaming giống như các lớp DirectoryInfo và FileInfo, lớp FileStream cho phép mở những tập tin hiện hữu cũng như tạo mới file. Khi tạo tập tin , lớp FileStream thường dùng những enum FileMode, FileAccess và FileShare

```
// tạo một tập tin mới trên thư mục làm việc
FileStream myFStream = new FileStream("test.dat",FileMode.OpenOrCreate,
FileAccess.ReadWrite);
```

5.3.2.2 The FileStream Class

FileStream được sử dụng đọc và viết dữ liệu vào hoặc từ một file. Để khởi tạo một FileStream, bạn cần 4 phần sau:

- **file** bạn muốn truy xuất .
- **mode**, cho biết bạn muốn mở file như thế nào.
- **access**, cho biết bạn muốn truy xuất file như thế nào – bạn định đọc hoặc viết file hoặc cả hai.
- **share** access – khả năng truy xuất file.

Enumeration	**Values**
FileMode	Append, Create, CreateNew, Open, OpenOrCreate, or Truncate

Enumeration	Values
FileAccess	Read, ReadWrite, or Write
FileShare	Inheritable, None, Read, ReadWrite, or Write

5.3.3 Làm việc với BufferedStream

Khi bạn triệu gọi hàm Read() thì một công tác đọc dữ liệu cho đầy buffer từ đĩa được tiến hành. Tuy nhiên, để cho có hiệu năng, hệ điều hành thường phải đọc trong một lúc một khối lượng lớn dữ liệu tạm thời trữ trên bufer. Buffer hoạt động như mọt kho hàng.

Một đối tượng Bufered stream cho phép hệ điều hành tạo buffer riêng cho mình dùng, rồi đọc dữ liệu vào hoặc viết dữ liệu lên ổ đĩa theo một khối lượng dữ liệu nào đó mà hệ điều hành thấy là có hiệu năng. Tuy nhiên, bạn xũng có thể ấn định chiều dài khối dữ liệu. Nhưng bạn nhớ cho là buffer sẽ chiêmd chỗ trong ký ức chứ không phải trên đĩa từ. Hiệu quả sử dụng đến buffer là ciệc xuất nhập dữ liệu chạy nhanh hơn.

Một đối tượng BufferedStream được hình thành xung quanh một đối tượng Stream mà bạn đã tạo ra trước đó. Muốn sử dụng đến một BufferedStream bạn bắt đầu tạo một đối tượng Stream thông thường như trong thí dụ :

```
stream inputstream = File.OpenRead(@"C;\test\source\folder3.cs ");
stream outputstream = File.Openwrite(@"C:test\source\folder3.bak");
```

Một khi bạn đã có stream bình thường, bạn trao đối tượng này cho hàm constructor của buffere stream:

```
BufferedStrream bufInput = new BufferedStream(inputstream);
```

```
BufferedStream bufOutput =new BufferedStream(outputstream);
```

Sau đó, bạn sử dụng BufferedStream như là một stream bình thường, bạn triệu gọi hàm Read() hoặc Write() như bạn đã làm trước kia. Hệ điều hành lo việc quản lý vùng đệm:

```
while ((bytesRead = bufInput.Read(buffer, 0, SIZE_BUFF))>0)
{  bufOutput.Write(buffer, 0, bytesRead);
}
```

Chỉ có một khác biệt mà bạn phải nhớ cho là phải tuôn ghi (flush) nội dung của buffer khi bạn muốn bảo đảm là dữ liệu được ghi lên đĩa.

```
bufOutput.Flush();
```

Lệnh trên bảo hệ điều hành lấy toàn bộ dữ liệu trên buffer cho tuôn ra ghi lên tập tin trên đĩa.

5.3.4 Làm việc với file văn bản

Nếu bạn biết file bạn đang làm việc (đọc/viết) thuộc loại văn bản nghĩa là dữ liệu kiểu string, thì bạn nên nghĩ đến việc sử dụng đến các lớp StreamReader và StreamWriter. Cả hai lớp theo mặc nhiên làm việc với ký tự Unicode. Tuy nhiên bạn có thể thay đổi điều này bằng cách cung cấp một đối tượng quy chiếu được cấu hình một cách thích hợp theo System.Text.Reference. Nói tóm lại hai lớp này được thiết kế để thao tác dễ dàng các tập tin loại văn bản.

Lớp StreamReader được dẫn xuất từ một lớp abstract mang tên TextReader cũng giống như String Reader. Lớp cơ bản TextReader cung cấp một số chức năng hạn chế cho mỗi hậu duệ, đặc biệt khả năng đọc và "liếc nhìn" (peek) lên một dòng ký tự (character stream).

Lớp StreamWriter và StringWriter cũng được dẫn xuất từ một lớp abstract mang tên TextWriter; lớp này định nghĩa những thành viên cho phép các lớp dẫn xuất viết những dữ liệu văn bản lên một dòng văn bản nào đó.

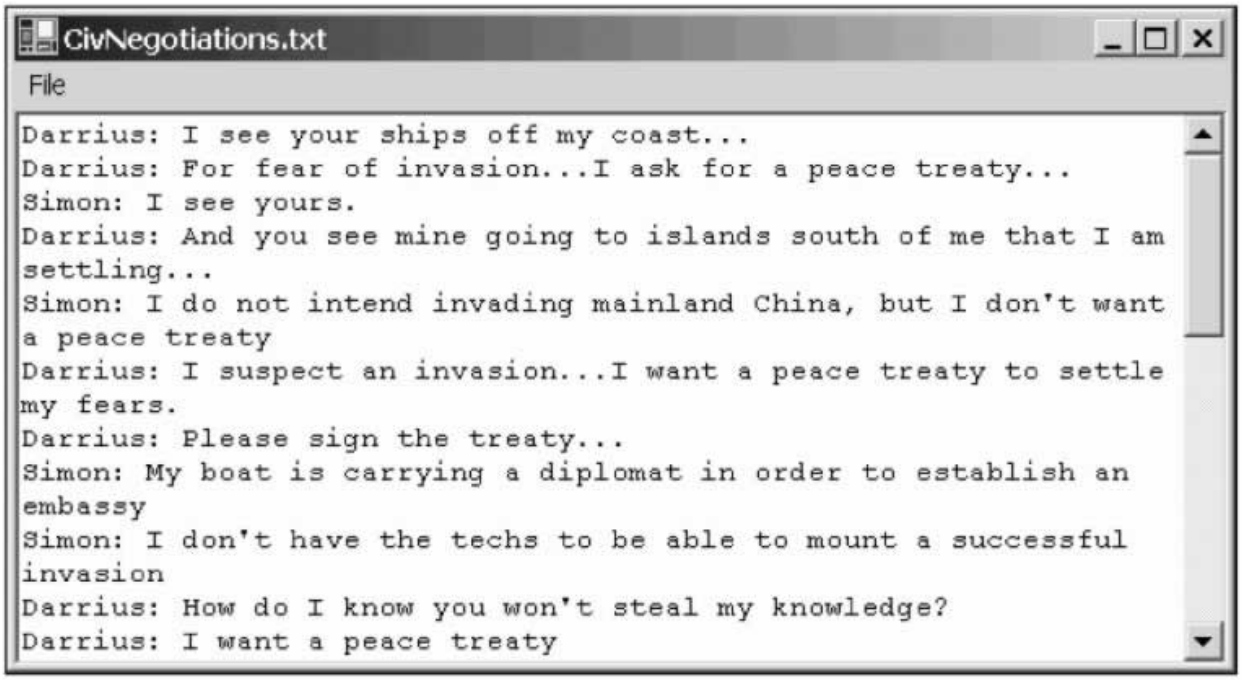
5.3.4.1 Các thành viên của lớp TextWriter

Tên phương thức	Ý nghĩa
Close()	Cho đóng lại các writer và giải phóng mọi nguồn lực chiếm dụng
Flush()	Cho xoá sạch tất cả các buffer đối với writer hiện hành
NewLine	Thuộc tính này dùng làm hằng sang hằng
Write()	Viết một hằng lên text stream không có newline constant
WriteLine()	Viết một hằng lên text stream có newline constant

5.3.4.2 Ví dụ đọc, viết một tập tin văn bản:

Ví dụ ReadWriteText trình bày cách sử dụng của lớp StreamReader và StreamWriter. Nó trình bày file được đọc vào và hiển thị Nó cũng có thể lưu file. Nó sẽ lưu bất kỳ file ở định dạng Unicode .

Màn hình trình bày ReadWriteText được dùng hiển thị file CivNegotiations. Chúng ta có thể đọc được ở nhiều định dạng file khác.



Chúng ta nhìn vào đoạn mã sau. Trước tiên ta thêm câu lệnh using , Từ đây bên cạnh System.IO, chúng ta sử dụng lớp StringBuilder từ System.Text namespace để xây dựng chuỗi trong textbox:

```
using System.IO;
using System.Text;
```

Tiếp theo chúng ta thêm các trường cho lớp main form

```
public class Form1 : System.Windows.Forms.Form
{
   private OpenFileDialog chooseOpenFileDialog = new OpenFileDialog();
   private string chosenFile;
```

Chúng ta cũng cần thêm vài chuẩn mã Windows Forms để thực hiện điều khiển cho menu và hộp thoại:

```
public Form1()
{
  InitializeComponent();
  menuFileOpen.Click += new EventHandler(OnFileOpen);
  chooseOpenFileDialog.FileOk += new
    CancelEventHandler(OnOpenFileDialogOK);
}
void OnFileOpen(object Sender, EventArgs e)
{
  chooseOpenFileDialog.ShowDialog();
}
void OnOpenFileDialogOK(object Sender, CancelEventArgs e)
{
  chosenFile = chooseOpenFileDialog.FileName;
  this.Text = Path.GetFileName(chosenFile);
  DisplayFile();
}
```

Từ đây chúng ta thấy mỗi khi người sử dụng nhấn OK để chọn một file trong hộp thoại, chúng ta gọi phương thức DisplayFile(), dùng để đọc file.

```
void DisplayFile()
{
  int nCols = 16;
```

```
FileStream inStream = new FileStream(chosenFile, FileMode.Open,
                                     FileAccess.Read);
long nBytesToRead = inStream.Length;
if (nBytesToRead > 65536/4)
  nBytesToRead = 65536/4;
int nLines = (int)(nBytesToRead/nCols) + 1;
string [] lines = new string[nLines];
int nBytesRead = 0;
for (int i=0 ; i<nLines ; i++)
{
  StringBuilder nextLine = new StringBuilder();
  nextLine.Capacity = 4*nCols;
  for (int j = 0 ; j<nCols ; j++)
  {
    int nextByte = inStream.ReadByte();
    nBytesRead++;
    if (nextByte < 0 || nBytesRead > 65536)
      break;
    char nextChar = (char)nextByte;
    if (nextChar < 16)
      nextLine.Append(" x0" + string.Format("{0,1:X}",
                                    (int)nextChar));
    else if
```

```
            (char.IsLetterOrDigit(nextChar) ||
                                char.IsPunctuation(nextChar))
            nextLine.Append("  " + nextChar + " ");
          else
            nextLine.Append(" x" + string.Format("{0,2:X}",
                                (int)nextChar));
        }
        lines[i] = nextLine.ToString();
      }
      inStream.Close();
      this.textBoxContents.Lines = lines;
    }
```

Như vậy chúng ta đã mở được file nhờ phương thức DisplayFile(). bây giờ chúng ta xử lý cách để lưu file chúng ta thêm đoạn mã SaveFile(). Bạn nhìn vào phương thức SaveFile() chúng ta viết mỗi dòng ra textbox, bằng stream StreamWriter

```
    void SaveFile()
    {
      StreamWriter sw = new StreamWriter(chosenFile, false,
                          Encoding.Unicode);
      foreach (string line in textBoxContents.Lines)
        sw.WriteLine(line);
      sw.Close();
    }
```

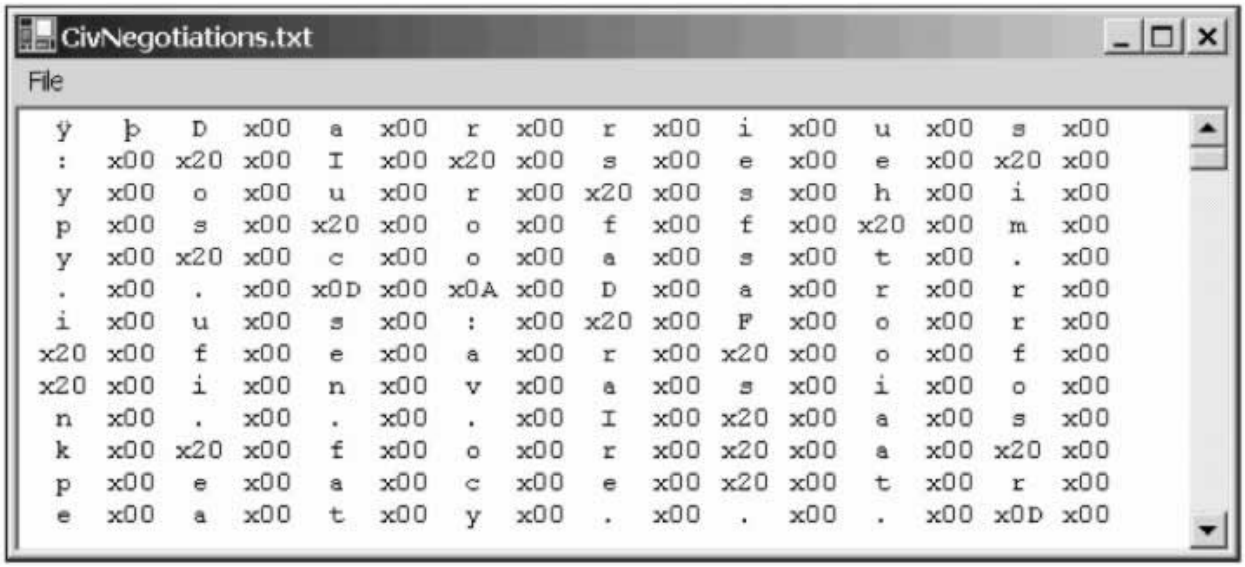
Bây giờ ta xem xét làm thế nào file được đọc vào. Trong quá trình xử lý thực sự chúng ta không biết có bao nhiêu dòng sẽ được chứa (cũng có nghĩa là có bao nhiêu ký tự (char)13 (char)10 tuần tự trong file đến khi nào kết thúc file) Chúng ta giải quyết vấn đề này bằng cách ban đầu đọc file vào trong lớp đại diện StringCollection, được nằm trong System.Collections.Specialized namespace. Lớp này được thiết kế để giữ một bộ của chuỗi có thể được mở rộng một cách linh hoạt. Nó thực thi hai phương thức : Add(), nó thêm một chuỗi vào bộ chọn lựa (collection) , và CopyTo(), nó sao chép string collection vào trong một mảng. Mỗi thành phần của đối tượng StringCollection object sẽ giữ 1 hàng của file.

Bây giờ chúng ta sẽ xem xét phương thức ReadFileIntoStringCollection() . Chúng ta sử dụng StreamReader để đọc trong mỗi hàng. Khó khăn chính là cần đếm ký tự đọc để chắc chúng ta không vượt quá khả năng chứa đựng của textbox:

```
StringCollection ReadFileIntoStringCollection()
{
  const int MaxBytes = 65536;
  StreamReader sr = new StreamReader(chosenFile);
  StringCollection result = new StringCollection();
  int nBytesRead = 0;
  string nextLine;
  while ( (nextLine = sr.ReadLine()) != null)
  {
    nBytesRead += nextLine.Length;
    if (nBytesRead > MaxBytes)
      break;
```

```
        result.Add(nextLine);
    }
    sr.Close();
    return result;
}
```

Đến đây đoạn mã được hoàn thành.



5.4 Đọc và viết vào Registry

Trong các của Windows từ Windows 95 trở đi Registry là trung tâm lưu trữ tất cả các thông tin cấu hình liên quan đến cài đặt Windows, sở thích người dùng, phần mền cài đặt, thiết bị. Hầu hết tất cả các phần mềm thương mại sử dụng Registry để chứa thông tin của chính nó, và các thành phần COM phải được đặt thông tin của chúng trong Registry để mà được gọi bởi các ứng dụng khách..NET Framework đã giảm sự quan trọng của Registry đối với ứng dụng, vì assembly đã trở thành "tự cung tự cấp" do đó không cần thông tin đặc điết để trữ trên Registry. Registry giờ đây chỉ là nơi tiện lợi để bạn trữ thông tin về sở thích của người sử dụng (user preference). Namespace

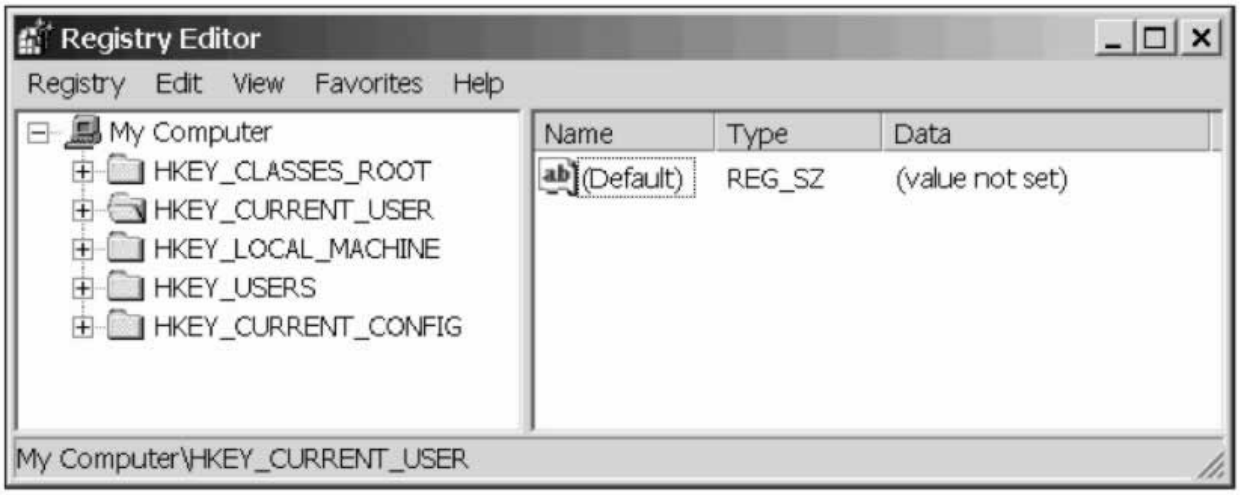
Microsoft.Win32 định nghĩa một vài lớp cho phép đọc hoặc viết system registry một cách dễ dàng.

Trước tiên chúng ta cùng xem lại cấu trúc của Registry

5.4.1 The Registry

Registry có một cấu trúc đẳng cấp giống như hệ thống các tập tin (file system). Cách thông thường để nhìn xem hoặc thay đổi nội dung của Registry là với một trong hai tiên ích: **regedit.exe** hoặ**c regedt32.exe** hiện diện trong tất cả các phiên bản Windows, từ khi Window 95 trở thành chuẩn. Còn Regedt32.exe thì chỉ hiện diện trong Windows NT và Windows 2000, ít thân thiện so với regedit.exe nhưng cho phép truy cập vào thông tin an ninh mà regedit không có khả năng nhìn xem. Trong phần này chúng ta sử dụng regedit.exe tại khung đối thoại Run hoặc command promt

Khi bạn khởi chạy regedit đầu tiên bạn sẽ thấy hình sau đây:



Regedit có giao diện mang dáng dấp treeview/listview giống như Windows Explorer, khớp với cấu trúc đẳng cấp của bản thân Registry . Tuy nhiên chúng ta sẽ thấy có vài sự khác biệt.

Trong một file system, các mắt cấp chóp có thể được xem là những partitions trên ổ đĩa , C:\, D:\, ... Trong Registry,tương đương với partition là **registry hive**. Các khuôn này có định và không thể thay đổi và có cả thảy là bảy.

- HKEY_CLASSES_ROOT (HKCR) chứa những chi tiết và các loại tập tin(.txt, .doc, and so on), và những ứng dụng nào có khả năng mở các tập tin loại nào. Ngoài ra nó còn chứa thông tin đăng ký đối với tất cả các cấu kiện COM (chiếm phần lớn Registry, vì Windows mang theo vô số thành phần COM).
- HKEY_CURRENT_USER (HKCU) chứa chi tiết liên quan đến sở thích của người sử dụng hiện đang đang nhập trên máy tính
- HKEY_LOCAL_MACHINE (HKLM) là hive đồ sộ chứa chi tiết tất cả phần mềm và phấn cứng được cài đặt trên máy tính
- HKEY_USERS (HKUSR) chứa chi tiết liên quan đến sở thích của tất cả người sử dụng Như bạn có thể chờ đợi, nó cũng chứa hive HKCU đơn giản là một ánh xạ lên một trong những key trên HKEY_USERS.
- HKEY_CURRENT_CONFIG (HKCF) chứa đựng chi tiết liên quan đến phần cứng máy tính.

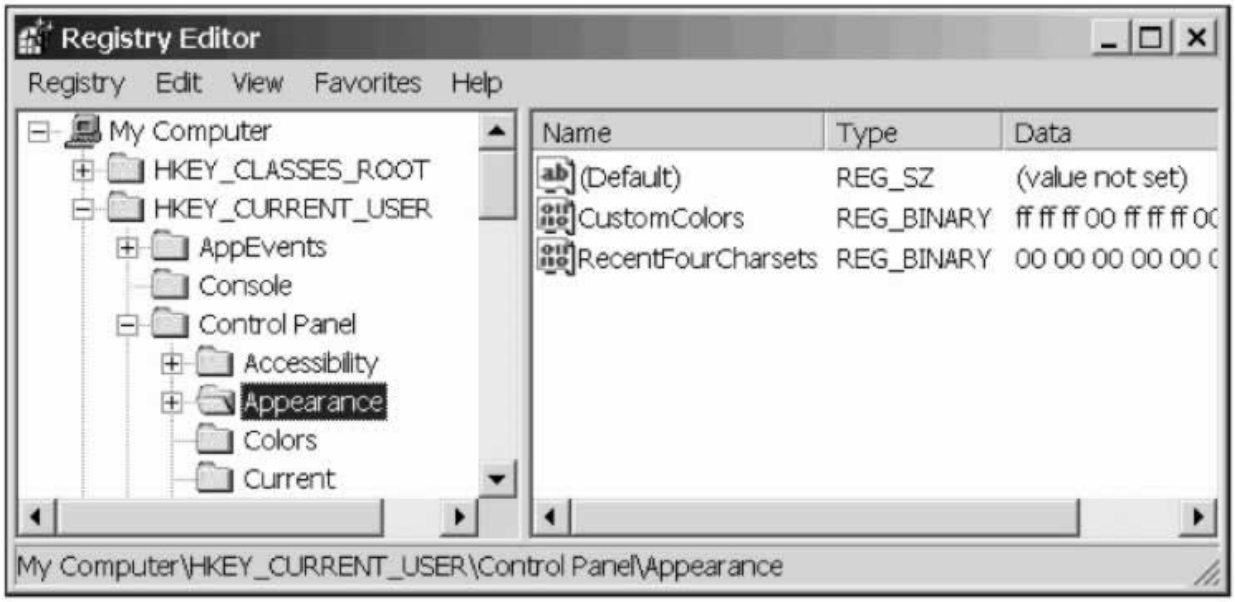
Phần còn lại là hai key chứa thông tin mang tình trạng tạm thời và thay đổi thường xuyên:

- HKEY_DYN_DATA là một container tổng quát đối với bất cứ dữ liệu volatile nào cần lưu trữ đâu đó trên Registry
- HKEY_PERFORMANCE_DATA chứa thông tin liên quan đến thành tích ứng dụng đang chạy.

Trong lòng các hive là một cấu trúc cây gồm các Registry key. Mỗi key (mục khoá) cũnggiống như một folder hoặc file trong một file system. Tuy nhiên có một khác

biệt rất quan trọng . File system phân biệt giữa các files và folders nhưng Registry hiện diện chỉ toàn là key. Một key có thể chứa cả dữ liệu và các key khác.

Nếu một key chứa dữ liệu thì lúc này nó sẽ hiện diện như là một loạt các trị Mỗi trị sẽ có một cái tên một kiểu dữ liệu và một trị Một key có thể có một trị mặc nhiên không được đặt tên



Key HKCU\Control Panel\Appearance có 3 bộ trị có mang tên mặc dù trị mặc nhiên không chứa bất cứ dữ liệu nào. Cột Type chi tiết hoá kiểu dữ liệu của mỗi trị. Các mục vào vào Registry có thể được định dạng theo một trong 3 kiểu dữ liệu:

- REG_SZ gần như tương đương với .NET string
- REG_DWORD gần như tương đương với .NET unit
- REG_BINARY bản dãy các byte

5.4.2 The .NET Registry Classes

Việc truy cập vào Registry trên .NET sẽ thông qua hai lớp Registry và RegistryKey thuộc amespace Microsoft.Win32. Một thể hiện của lớp RegistryKey tượng

trưng cho một registry key. Lớp RegistryKey cung cấp những thành viên cốt lõi cho phép bạn làm việc với registry key.

Lớp RegistryKey sẽ là lớp mà bạn sẽ dùng để làm việc với registry key. Ngược lại lớp Registry là lớp mà bạn chả bao giờ thể hiện. Vai trò của nó là cung cấp cho bạn những thể hiện RegistryKey tượng trưng cho key top-level những hive khác nhau để qua các thuộc tính static và có cả thảy 7 thuộc tính bao gồm ClassesRoot, CurrentConfig, CurrentUser, DynData, LocalMachine, PerformanceData, and Users. chắc chắn bạn đã biết thuộc tính nào chỉ Registry hive nào Do đó muốn có một thể hiện của một RegistryKey tượng trưng cho key HKLM, bạn viết

```
RegistryKey hklm = Registry.LocalMachine;
```

Nếu bạn muốn đọc một vài dữ liệu trên key HKLM\Software\Microsoft, bạn phải đi lấy qui chiếu về key như sau :

```
RegistryKey hklm = Registry.LocalMachine;
RegistryKey hkSoftware = hklm.OpenSubKey("Software");
RegistryKey hkMicrosoft = hkSoftware.OpenSubKey("Microsoft");
```

Một registry key được truy cập theo kiểu này chi cho phép bạn đọc mà thôi, Nếu bạn muốn có khả năng viết lên key (bao gồm viết lên trị của key, tạo hoặc gỡ bỏ cây con cái thuộc quyền), bạn phải sử dụng một OpenSubkey nhận thêm một thông số thứ hai thuộc kiểu bool cho biết quyền read-write đối với key. Ví dụ bạn muốn có khả năng thay đổi key Microsoft.

```
RegistryKey hklm = Registry.LocalMachine;
RegistryKey hkSoftware = hklm.OpenSubKey("Software");
RegistryKey hkMicrosoft = hkSoftware.OpenSubKey("Microsoft", true);
```

Phương thức OpenSubKey() là một trong những hàm mà bạn triệu gọi nếu bạn chờ đợi key hiện hữu. Nếu nó không có thì nó sẽ trở về null preference. Còn nếu bạn muốn tạo một key mới bạn sẽ dùng CreateSubKey() (hàm này tự hoạt động cho quyền read write):

```
RegistryKey hklm = Registry.LocalMachine;
RegistryKey hkSoftware = hklm.OpenSubKey("Software");
RegistryKey hkMine = hkSoftware.CreateSubKey("MyOwnSoftware");
```

Một khi bạn đã có registry key bạn muốn đọc hoặc thay đổi, bạn có thể sử dụng các phương thức SetValue() hoặc GetValue() để đặt hoặc để lấy dữ liệu trên key. Thí dụ:

```
RegistryKey hkMine = HkSoftware.CreateSubKey("MyOwnSoftware");
hkMine.SetValue("MyStringValue", "Hello World");
hkMine.SetValue("MyIntValue", 20);
```

Đoạn mã trên sẽ đặt key về hai trị : MyStringValue sẽ mang kiểu dữ liệu REG_SZ, trong khi MyIntValue sẽ mang kiểu dữ liệu REG_DWORD.

RegistryKey.GetValue() cũng như vậy Nó được định nghĩa trả về một quy chiếu đối tượng, nghĩa là trả về một quy chiếu string nếu nó thấy có kiểu dữ liệu REG_SZ, và int nếu phát hiện kiểu dữ liệu REG_DWORD:

```
string stringValue = (string)hkMine.GetValue("MyStringValue");
int intValue = (int)hkMine.GetValue("MyIntValue");
```

Cuối cùng khi xong việc bạn phải cho đóng lại

```
hkMine.Close();
```

Các thành phần của RegistryKey bao gồm thuộc tính và phương thức sau:

5.4.2.1 Properties

Property Name	Description
Name	Tên của key (read-only)
SubKeyCount	số lượng sub key
ValueCount	Các trị trên key

5.4.2.2 Methods

Method Name	Purpose
Close()	Đóng lại key
CreateSubKey()	Tạo một subkey của một tên được cho
DeleteSubKey()	Bỏ một key được chỉ định
DeleteSubKeyTree()	Gỡ bỏ một cách đệ quy một subkey
DeleteValue()	Tháo bỏ một tên trị từ một key
GetSubKeyNames()	Trả về một dãy chuỗi chứa tên của subkeys
GetValue()	Trả về một tên trị
GetValueNames()	Trả về một dãy chuỗi chứa tên của tất cả các trị của key
OpenSubKey()	Trả về một tham khảo đến một RegistryKey,hàm này cho tìm lại subkey

Method Name	Purpose
SetValue()	Hàm này cho đặt một trị được chỉ định.