Files and Streams

Today you will learn

- Files and Web Applications
- File System Information
- Reading and Writing with Streams

Files and Web Applications

- **Limitations of Files:**
  - File-naming limitations
  - Multiuser limitations
  - Scalability problems
  - Security risks

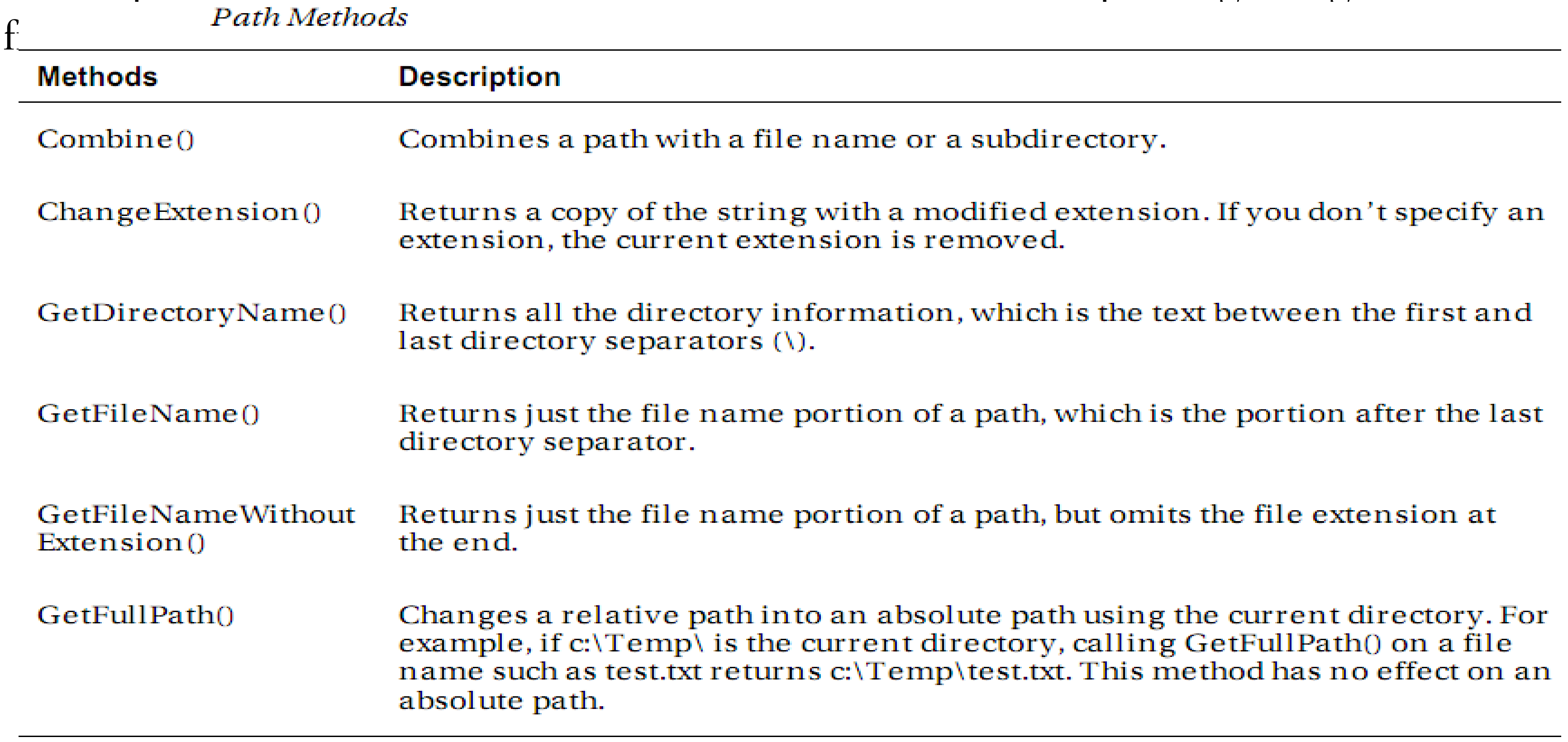
File System Information

- Typical File System Operations are: Copying Files and Creating Directories.
- .NET uses five classes for this, which are in System.IO Namespace.
- The Directory and File classes, which provide static methods that allow you to retrieve information about any files and directories visible from your server.
- The DirectoryInfo and FileInfo classes, which use similar instance methods and properties to retrieve the same sort of information.
- The DriveInfo class, which provides static methods that allow you to retrieve information about a drive and the amount of free space it provides

File System Information

Path Class

- The Path class doesn’t include any real file management functionality.
- Which provides a few static methods that are useful when manipulating strings that contain f



Path Methods

Methods	Description
Combine()	Combines a path with a file name or a subdirectory.
ChangeExtension()	Returns a copy of the string with a modified extension. If you don't specify an extension, the current extension is removed.
GetDirectoryName()	Returns all the directory information, which is the text between the first and last directory separators (\).
GetFileName()	Returns just the file name portion of a path, which is the portion after the last directory separator.
GetFileNameWithout Extension()	Returns just the file name portion of a path, but omits the file extension at the end.
GetFullPath()	Changes a relative path into an absolute path using the current directory. For example, if c:\Temp\ is the current directory, calling GetFullPath() on a file name such as test.txt returns c:\Temp\test.txt. This method has no effect on an absolute path.

File System Information

Methods	Description
GetPathRoot()	Retrieves a string with the root drive (for example, "c:\"), provided that information is in the string. For a relative path, it returns a null reference.
HasExtension()	Returns True if the path ends with an extension.
IsPathRooted()	Returns True if the path is an absolute path and False if it's a relative path.

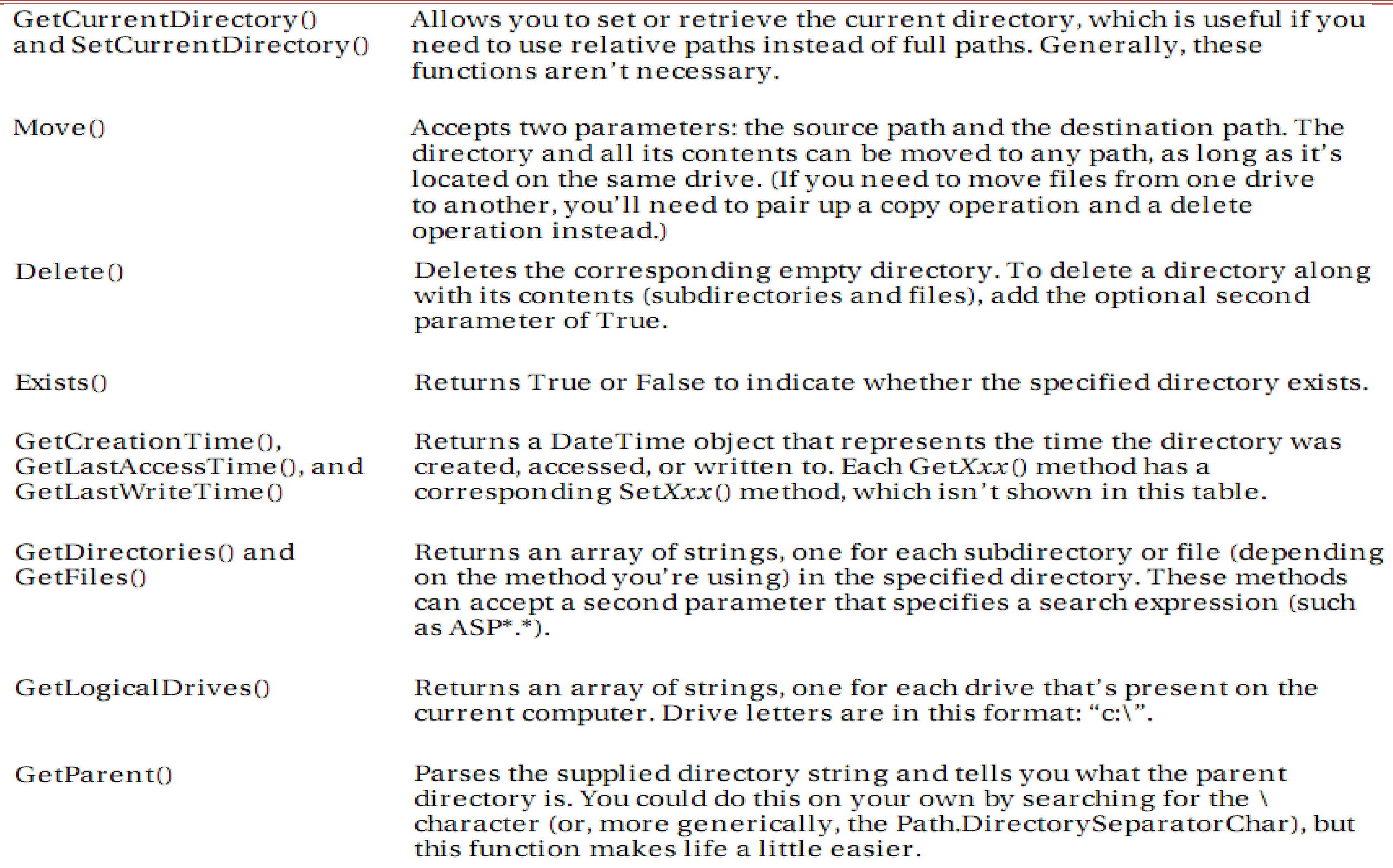
```
string file= Path.GetFileName( @"c:\Documents\Upload\Users\JamesX\resume.doc")
//file now contains "resume.doc"

string absolutePath = @"c:\Users\MyDocuments"
string subPath = @"Sarah\worksheet.xls"
string combined = Path.Combine(absolutePath, subPath)
// combined now contains "c:\Users\MyDocuments\Sarah\worksheet.xls"
```

The Directory and File Classes

Directory Class Members

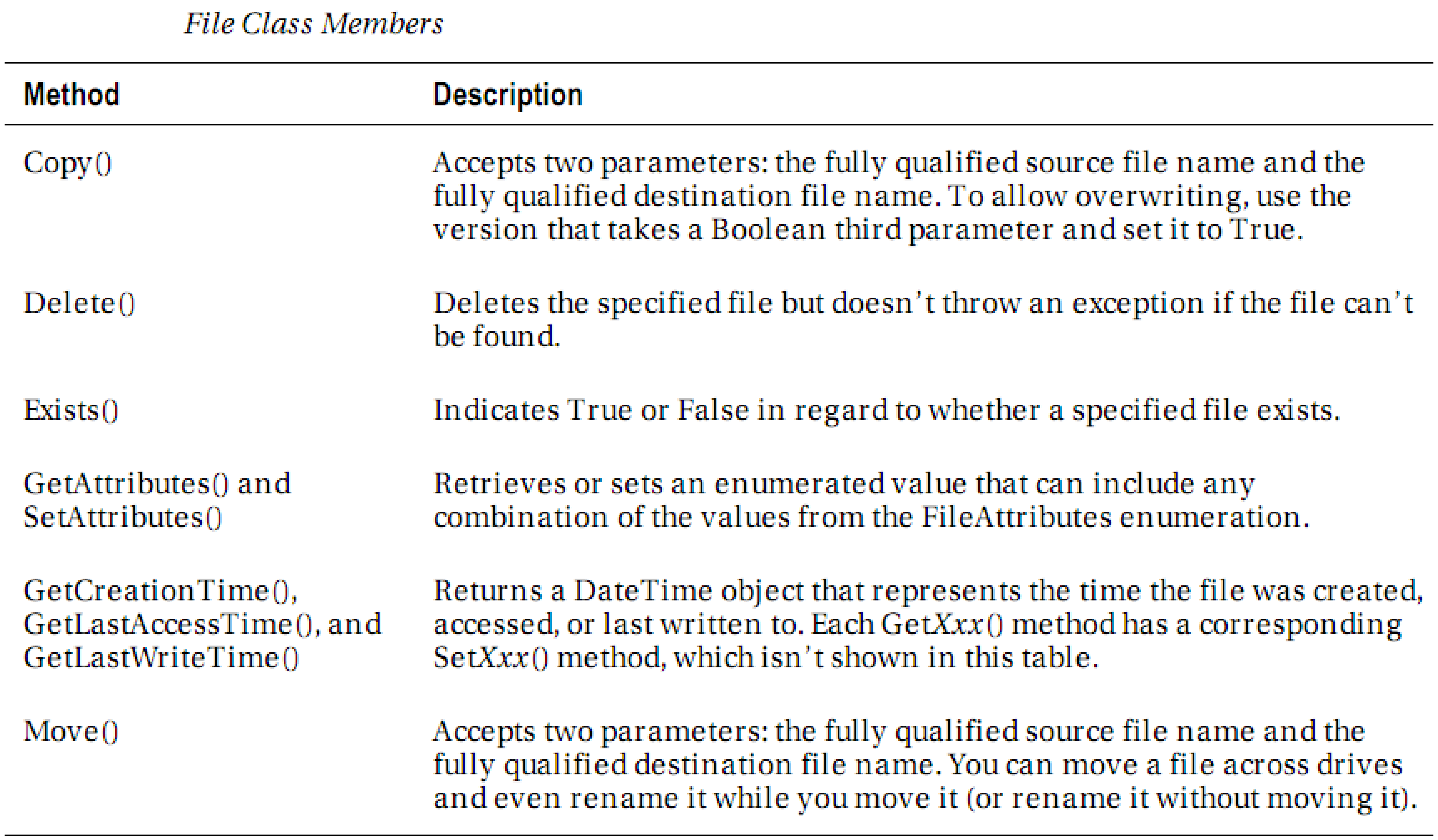
Method	Description
CreateDirectory()	Creates a new directory. If you specify a directory inside another nonexistent directory, ASP.NET will thoughtfully create *all* the required directories.

The Directory and File Classes



GetCurrentDirectory() and SetCurrentDirectory()	Allows you to set or retrieve the current directory, which is useful if you need to use relative paths instead of full paths. Generally, these functions aren't necessary.
Move()	Accepts two parameters: the source path and the destination path. The directory and all its contents can be moved to any path, as long as it's located on the same drive. (If you need to move files from one drive to another, you'll need to pair up a copy operation and a delete operation instead.)
Delete()	Deletes the corresponding empty directory. To delete a directory along with its contents (subdirectories and files), add the optional second parameter of True.
Exists()	Returns True or False to indicate whether the specified directory exists.
GetCreationTime(), GetLastAccessTime(), and GetLastWriteTime()	Returns a DateTime object that represents the time the directory was created, accessed, or written to. Each Get*Xxx*() method has a corresponding Set*Xxx*() method, which isn't shown in this table.
GetDirectories() and GetFiles()	Returns an array of strings, one for each subdirectory or file (depending on the method you're using) in the specified directory. These methods can accept a second parameter that specifies a search expression (such as ASP*.*).
GetLogicalDrives()	Returns an array of strings, one for each drive that's present on the current computer. Drive letters are in this format: "c:\".
GetParent()	Parses the supplied directory string and tells you what the parent directory is. You could do this on your own by searching for the \ character (or, more generically, the Path.DirectorySeparatorChar), but this function makes life a little easier.

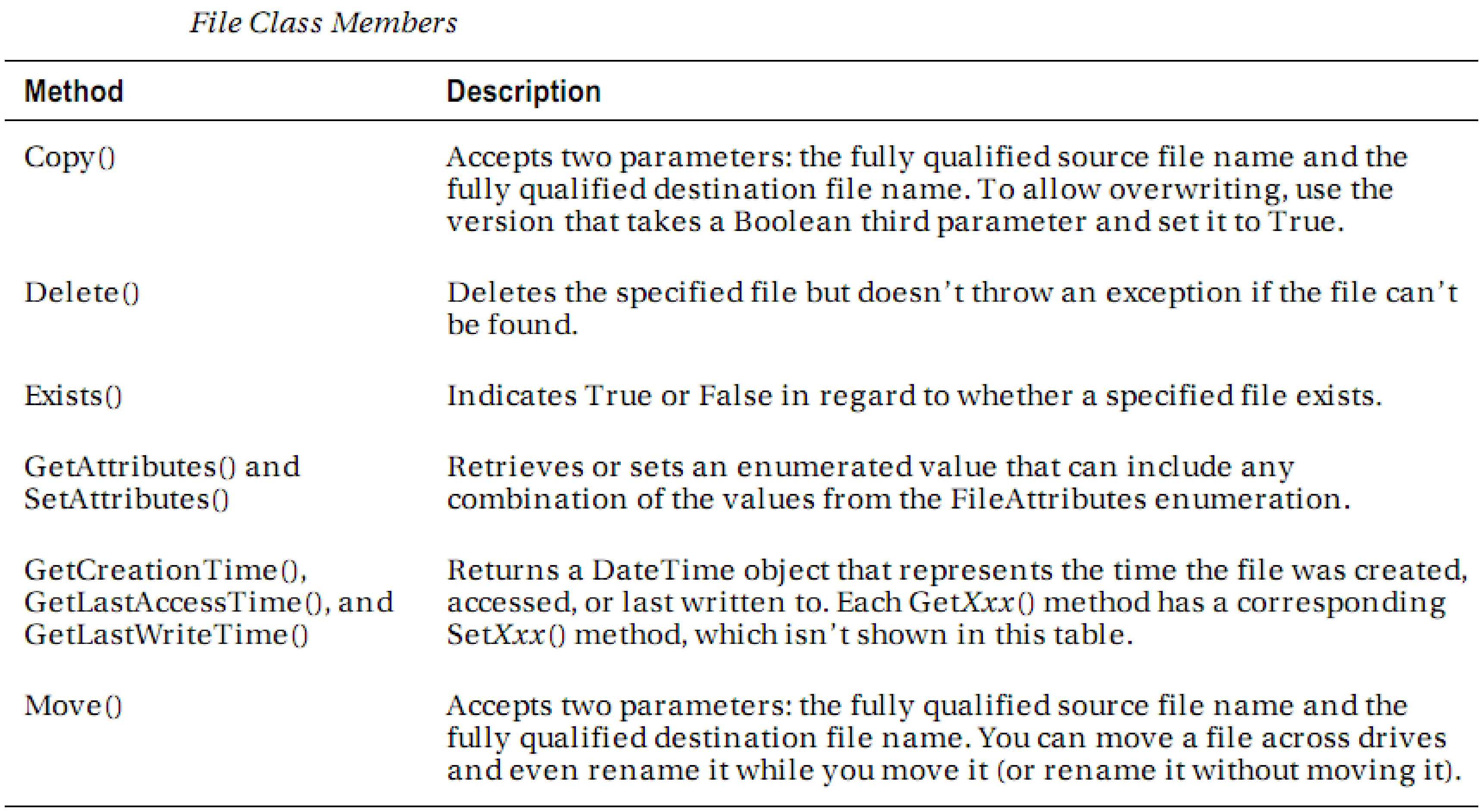
The Directory and File Classes



File Class Members

Method	Description
Copy()	Accepts two parameters: the fully qualified source file name and the fully qualified destination file name. To allow overwriting, use the version that takes a Boolean third parameter and set it to True.
Delete()	Deletes the specified file but doesn't throw an exception if the file can't be found.
Exists()	Indicates True or False in regard to whether a specified file exists.
GetAttributes() and SetAttributes()	Retrieves or sets an enumerated value that can include any combination of the values from the FileAttributes enumeration.
GetCreationTime(), GetLastAccessTime(), and GetLastWriteTime()	Returns a DateTime object that represents the time the file was created, accessed, or last written to. Each Get*Xxx*() method has a corresponding Set*Xxx*() method, which isn't shown in this table.
Move()	Accepts two parameters: the fully qualified source file name and the fully qualified destination file name. You can move a file across drives and even rename it while you move it (or rename it without moving it).

The Directory and File Classes



File Class Members

Method	Description
Copy()	Accepts two parameters: the fully qualified source file name and the fully qualified destination file name. To allow overwriting, use the version that takes a Boolean third parameter and set it to True.
Delete()	Deletes the specified file but doesn't throw an exception if the file can't be found.
Exists()	Indicates True or False in regard to whether a specified file exists.
GetAttributes() and SetAttributes()	Retrieves or sets an enumerated value that can include any combination of the values from the FileAttributes enumeration.
GetCreationTime(), GetLastAccessTime(), and GetLastWriteTime()	Returns a DateTime object that represents the time the file was created, accessed, or last written to. Each Get*Xxx*() method has a corresponding Set*Xxx*() method, which isn't shown in this table.
Move()	Accepts two parameters: the fully qualified source file name and the fully qualified destination file name. You can move a file across drives and even rename it while you move it (or rename it without moving it).

(Demo)

The DirectoryInfo and FileInfo Classes

DirectoryInfo and FileInfo Members

Member	Description
Attributes	Allows you to retrieve or set attributes using a combination of values from the FileAttributes enumeration.
CreationTime, LastAccessTime, and LastWriteTime	Allows you to set or retrieve the creation time, last-access time, and last-write time using a DateTime object.
Exists	Returns True or False depending on whether the file or directory exists. In other words, you can create FileInfo and DirectoryInfo objects that don't actually correspond to current physical directories, although you obviously won't be able to use properties such as CreationTime and methods such as MoveTo().
FullName, Name, and Extension	Returns a string that represents the fully qualified name, the directory or file name (with extension), or the extension on its own, depending on which property you use.
Delete()	Removes the file or directory, if it exists. When deleting a directory, it must be empty, or you must specify an optional parameter set to True.
Refresh()	Updates the object so it's synchronized with any file system changes that have happened in the meantime (for example, if an attribute was changed manually using Windows Explorer).
Create()	Creates the specified directory or file.
MoveTo()	Copies the directory and its contents or the file. For a DirectoryInfo object, you need to specify the new path; for a FileInfo object, you specify a path and file name.

The DirectoryInfo and FileInfo Classes

Unique DirectoryInfo Members

Member	Description
Parent and Root	Returns a DirectoryInfo object that represents the parent or root directory. For a directory like c:\temp\myfiles, the parent is c:\temp, and the root is c:\.
CreateSubdirectory()	Creates a directory with the specified name in the directory represented by the DirectoryInfo object. It also returns a new DirectoryInfo object that represents the subdirectory.
GetDirectories()	Returns an array of DirectoryInfo objects that represent all the subdirectories contained in this directory.
GetFiles()	Returns an array of FileInfo objects that represent all the files contained in this directory.

Unique FileInfo Members

Member	Description
Directory	Returns a DirectoryInfo object that represents the parent directory.
DirectoryName	Returns a string that identifies the name of the parent directory.
Length	Returns a Long (64-bit integer) with the file size in bytes.
CopyTo()	Copies a file to the new path and file name specified as a parameter. It also returns a new FileInfo object that represents the new (copied) file. You can supply an optional additional parameter of True to allow overwriting.

The DirectoryInfo and FileInfo Classes

- Create a DirectoryInfo or FileInfo object to specify the full path in the constructor:

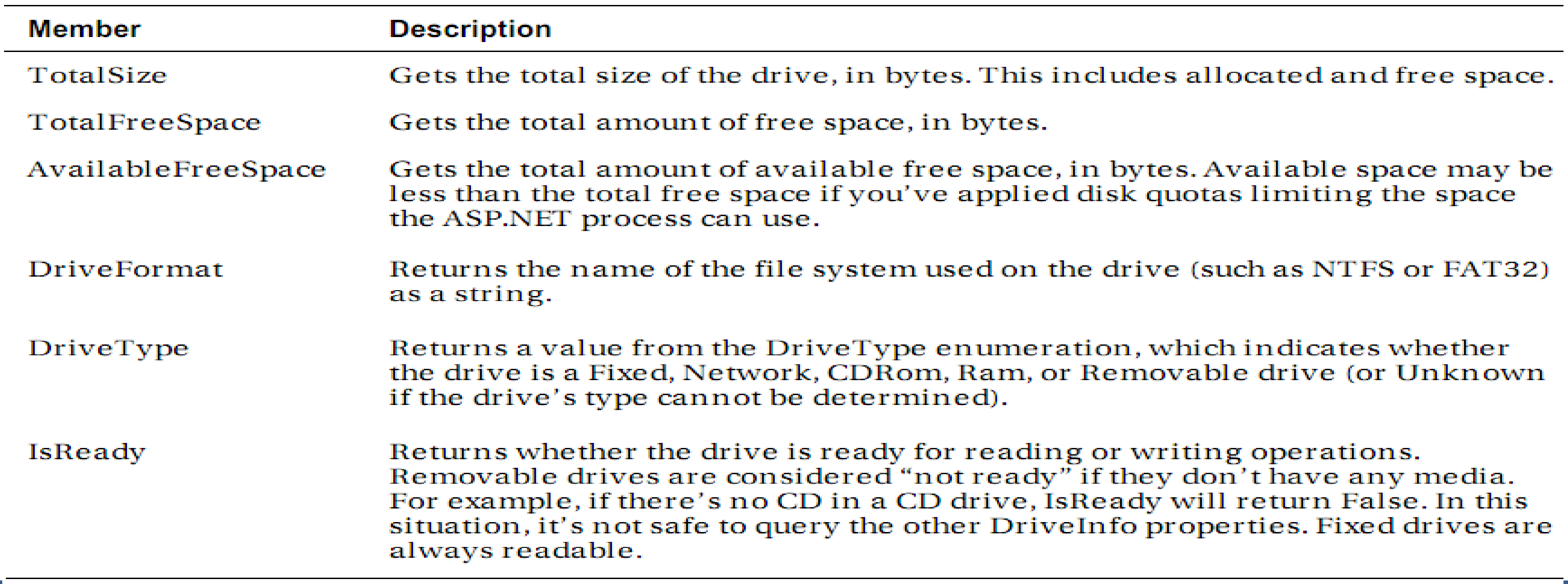
```
DirectoryInfo myDirectory = new DirectoryInfo(@”c:\Temp”);
FileInfo myfile = new FileInfo(@”c:\Temp\readme.txt”);
```

- If this does not locate the physical path use Create() method.

```
myDirectory.Create();
myFile.Create();
```

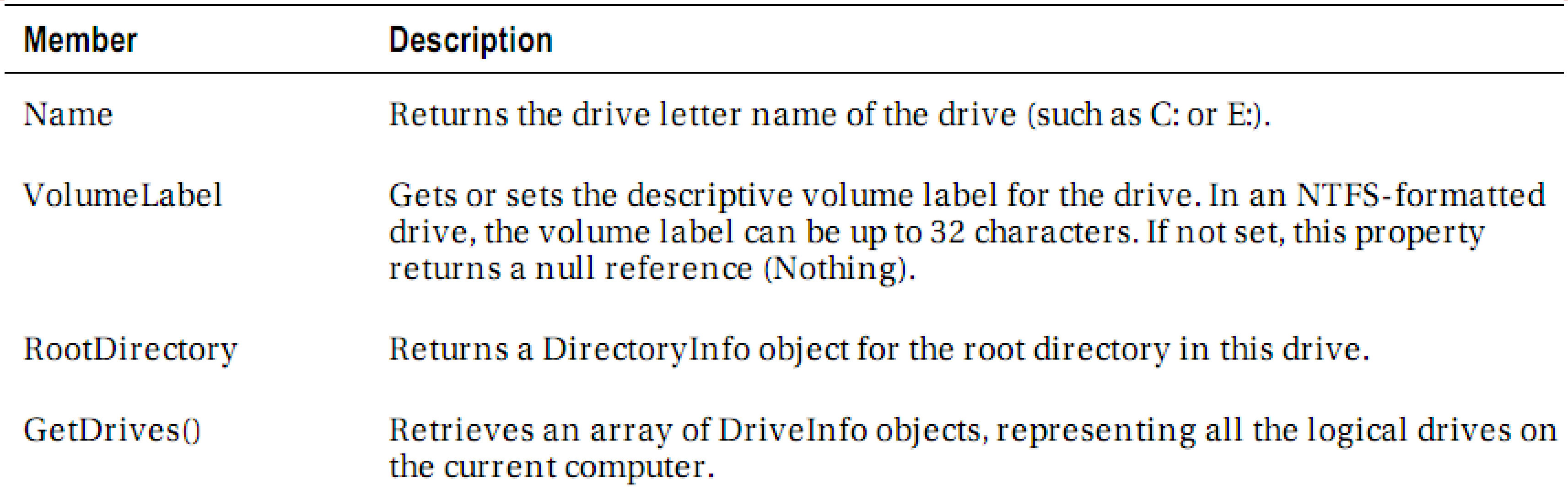
The DriveInfo Class

DriveInfo Members



Member	Description
TotalSize	Gets the total size of the drive, in bytes. This includes allocated and free space.
TotalFreeSpace	Gets the total amount of free space, in bytes.
AvailableFreeSpace	Gets the total amount of available free space, in bytes. Available space may be less than the total free space if you've applied disk quotas limiting the space the ASP.NET process can use.
DriveFormat	Returns the name of the file system used on the drive (such as NTFS or FAT32) as a string.
DriveType	Returns a value from the DriveType enumeration, which indicates whether the drive is a Fixed, Network, CDRom, Ram, or Removable drive (or Unknown if the drive's type cannot be determined).
IsReady	Returns whether the drive is ready for reading or writing operations. Removable drives are considered “not ready” if they don't have any media. For example, if there's no CD in a CD drive, IsReady will return False. In this situation, it's not safe to query the other DriveInfo properties. Fixed drives are always readable.

The DriveInfo Class



Member	Description
Name	Returns the drive letter name of the drive (such as C: or E:).
VolumeLabel	Gets or sets the descriptive volume label for the drive. In an NTFS-formatted drive, the volume label can be up to 32 characters. If not set, this property returns a null reference (Nothing).
RootDirectory	Returns a DirectoryInfo object for the root directory in this drive.
GetDrives()	Retrieves an array of DriveInfo objects, representing all the logical drives on the current computer.

A Sample File Browser (Demo)

File Browser Example

```
<asp:DropDownList ID="ddlDrives" runat="server"
    OnSelectedIndexChanged="ddlDrives_SelectedIndexChanged"
    AutoPostBack="true"></asp:DropDownList><br />
<asp:ListBox ID="lstDirectories" runat="server"
    OnSelectedIndexChanged="lstDirectories_SelectedIndexChanged"
    AutoPostBack="true"></asp:ListBox><br />
<asp:ListBox ID="lstFiles" runat="server"
    OnSelectedIndexChanged="lstFiles_SelectedIndexChanged"
    AutoPostBack="true"></asp:ListBox><br />
<asp:Label ID="lblFileInfo" runat="server" Text="Label"></asp:Label>
```

```
using System.IO;
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        foreach (var d in DriveInfo.GetDrives())
        {
            ddlDrives.Items.Add(d.Name);
        }
    }
}
protected void ddlDrives_SelectedIndexChanged(object sender, EventArgs e)
{
    foreach(var d in Directory.GetDirectories(ddlDrives.SelectedItem.Text))
    {
        lstDirectories.Items.Add(d);
    }
}
protected void lstDirectories_SelectedIndexChanged(object sender, EventArgs e)
{
    foreach (var f in Directory.GetFiles(lstDirectories.SelectedItem.Text))
    {
        lstFiles.Items.Add(f);
    }
    //Another way
    //DirectoryInfo d = new DirectoryInfo(lstDirectories.SelectedItem.Text);
    //foreach (var f in d.GetFiles("*.pdf"))
    //{
    //    lstFiles.Items.Add(f);
    //}
}
```

```
protected void lstFiles_SelectedIndexChanged(object sender, EventArgs e)
{
    //Creating a File
    StreamWriter w = File.CreateText(@"D:\test.txt");
    w.Write("abc");
    w.Write("xyz");
    w.Close();
    //Another way of writing to a File
    File.WriteAllText(@"D:\test.txt", "abc xyz abc");
    //Deleting a File
    File.Delete(@"D:\test.txt");
    //Using Path Class
    //Path.Combine("D:", "folder", "folder2");
    lblFileInfo.Text = Path.GetExtension(lstFiles.SelectedItem.Text);
    //Get Info in the File
    FileInfo f = new FileInfo(lstFiles.SelectedItem.Text);
    lblFileInfo.Text =  f.CreationTime + " " + f.LastAccessTime;
    //lblFileInfo.Text=
             File.GetCreationTime(lstFiles.SelectedItem.Text).ToString();
    //lblFileInfo.Text +=
             File.GetLastAccessTime(lstFiles.SelectedItem.Text).ToString();
}
```

Reading and Writing with Streams

- .NET framework support for creation of Flat files like text and binary files.

Text Files

- Write into a file using StreamWriter
- Read from a file using StreamReader
- File class support methods called CreateText() or OpenText().

```
' Define a StreamWriter (which is designed for writing text files).
          StreamWriter  w;
' Create the file, and get a StreamWriter for it.
          w = File.CreateText("c:\myfile.txt")
w.WriteLine("This file generated by ASP.NET")        ' Write a string.
w.WriteLine(42)                                      ' Write a number.
```

- Close the File using Close() or Dispose() method

```
          w.Close()
```

Demo

Read from Text File

- ReadLine() returns a null reference when there is no more data in the file.

Demo

Reading and Writing with Streams

Multi-user friendly StreamReader

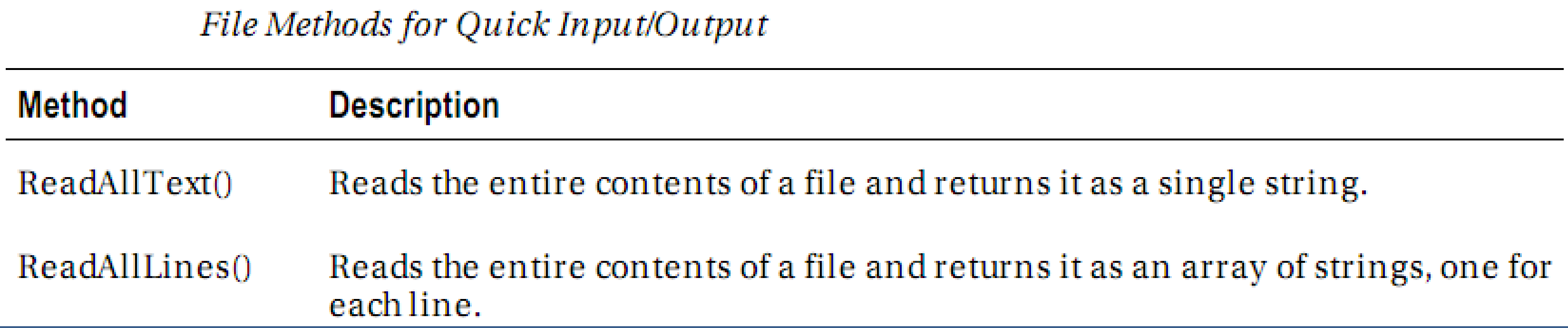
```
FileStream fs;
fs = File.Open("c:\myfile.txt", FileMode.Open, FileAccess.Read, FileShare.Read)
Dim r As New StreamReader(fs)
```

Binary Files

- To write in a Binary File create BinaryWriter object
- Create constructor of File.OpenWrite()
- To read from a Binary File create BinaryReader object
- Create constructor of File.OpenRead()

Shortcuts for Reading and Writing Files

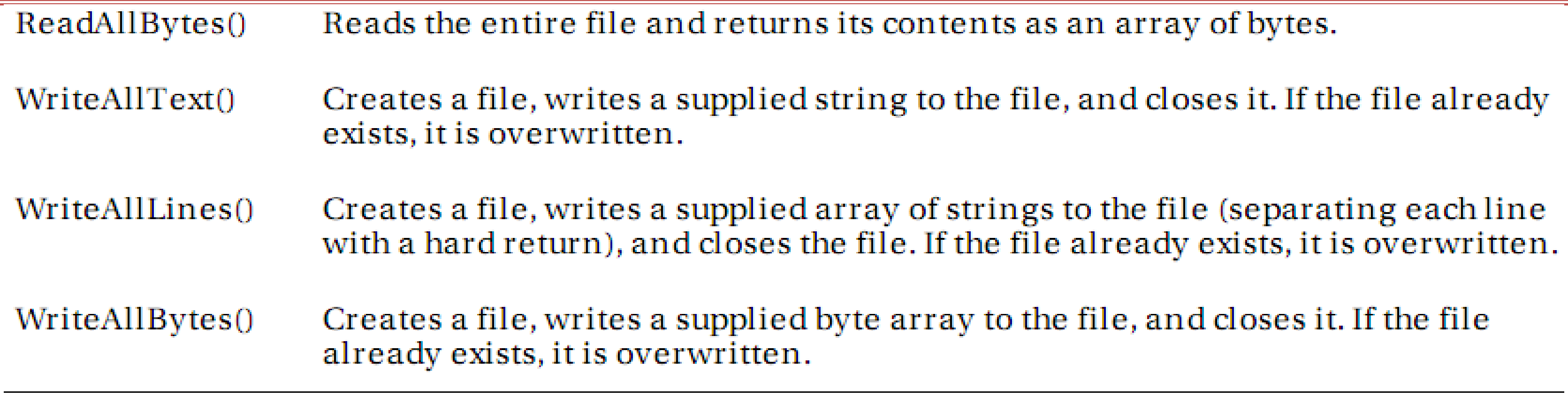
Demo



File Methods for Quick Input/Output

Method	Description
ReadAllText()	Reads the entire contents of a file and returns it as a single string.
ReadAllLines()	Reads the entire contents of a file and returns it as an array of strings, one for each line.

Reading and Writing with Streams



ReadAllBytes()	Reads the entire file and returns its contents as an array of bytes.
WriteAllText()	Creates a file, writes a supplied string to the file, and closes it. If the file already exists, it is overwritten.
WriteAllLines()	Creates a file, writes a supplied array of strings to the file (separating each line with a hard return), and closes the file. If the file already exists, it is overwritten.
WriteAllBytes()	Creates a file, writes a supplied byte array to the file, and closes it. If the file already exists, it is overwritten.

END OF LECTURE