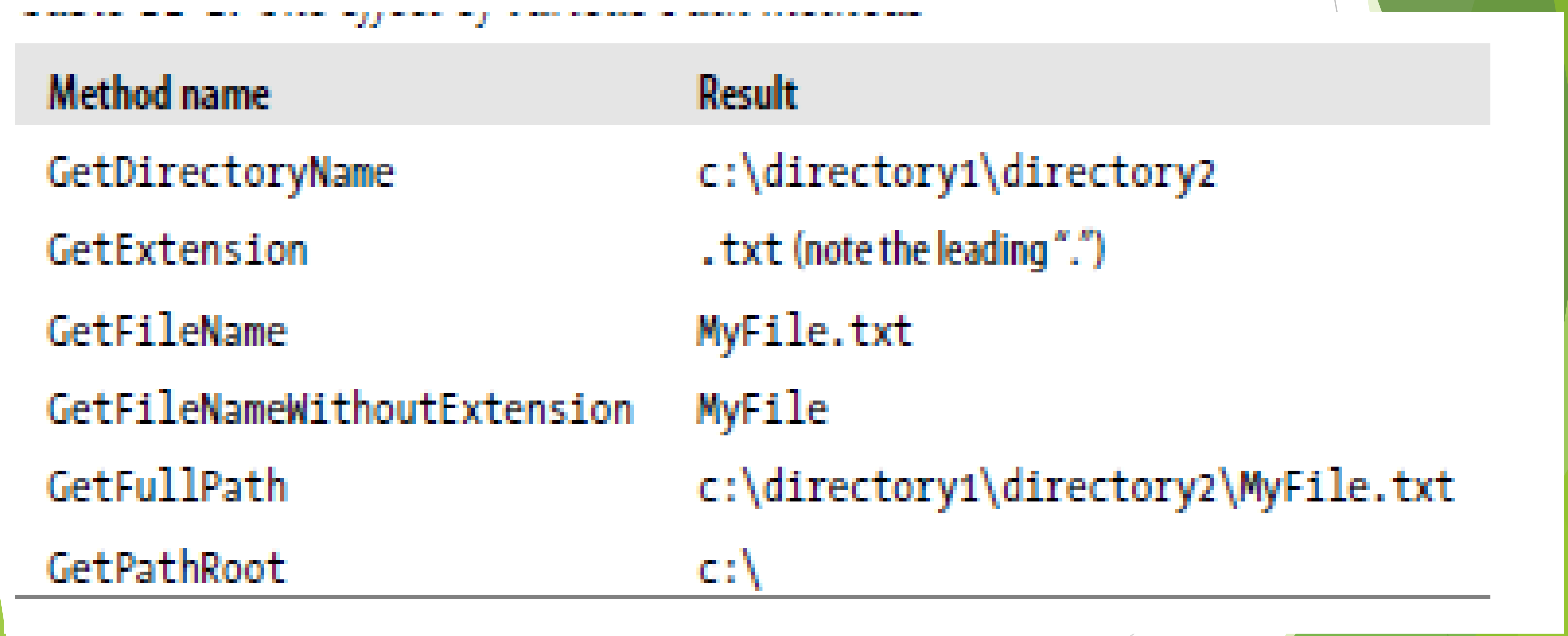
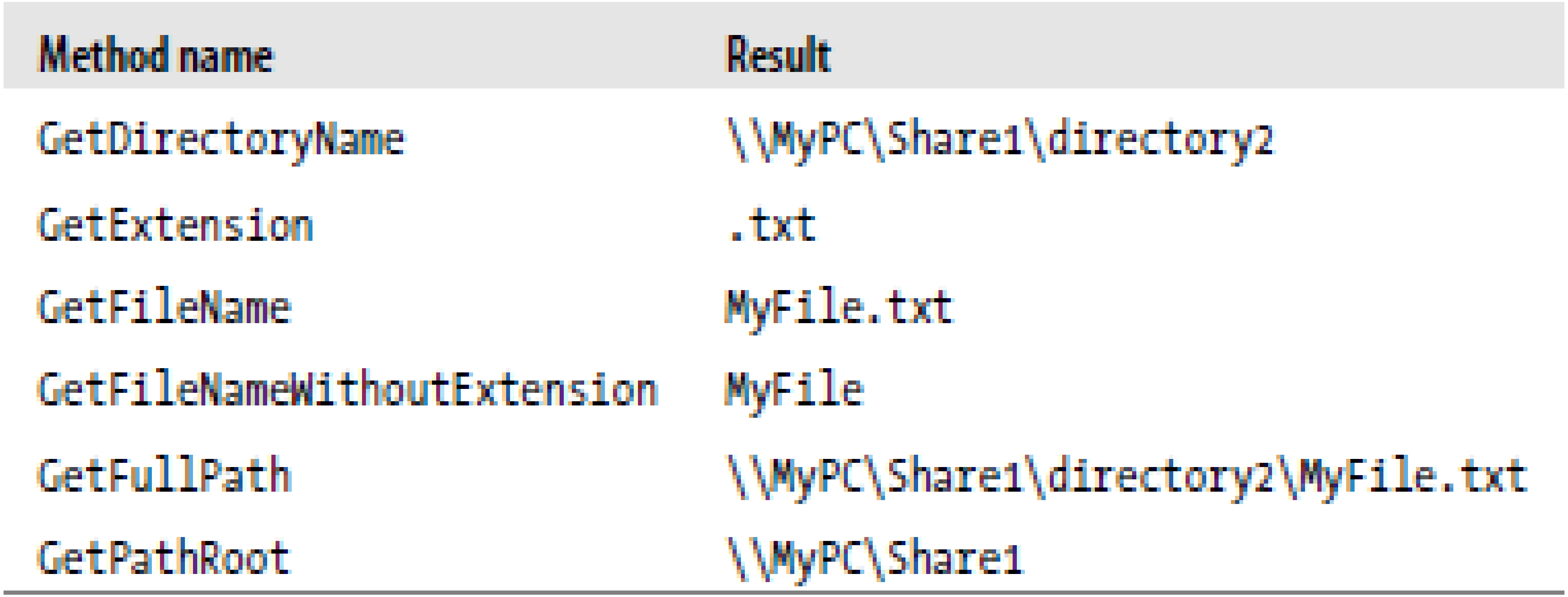
Session 20

Files and Streams



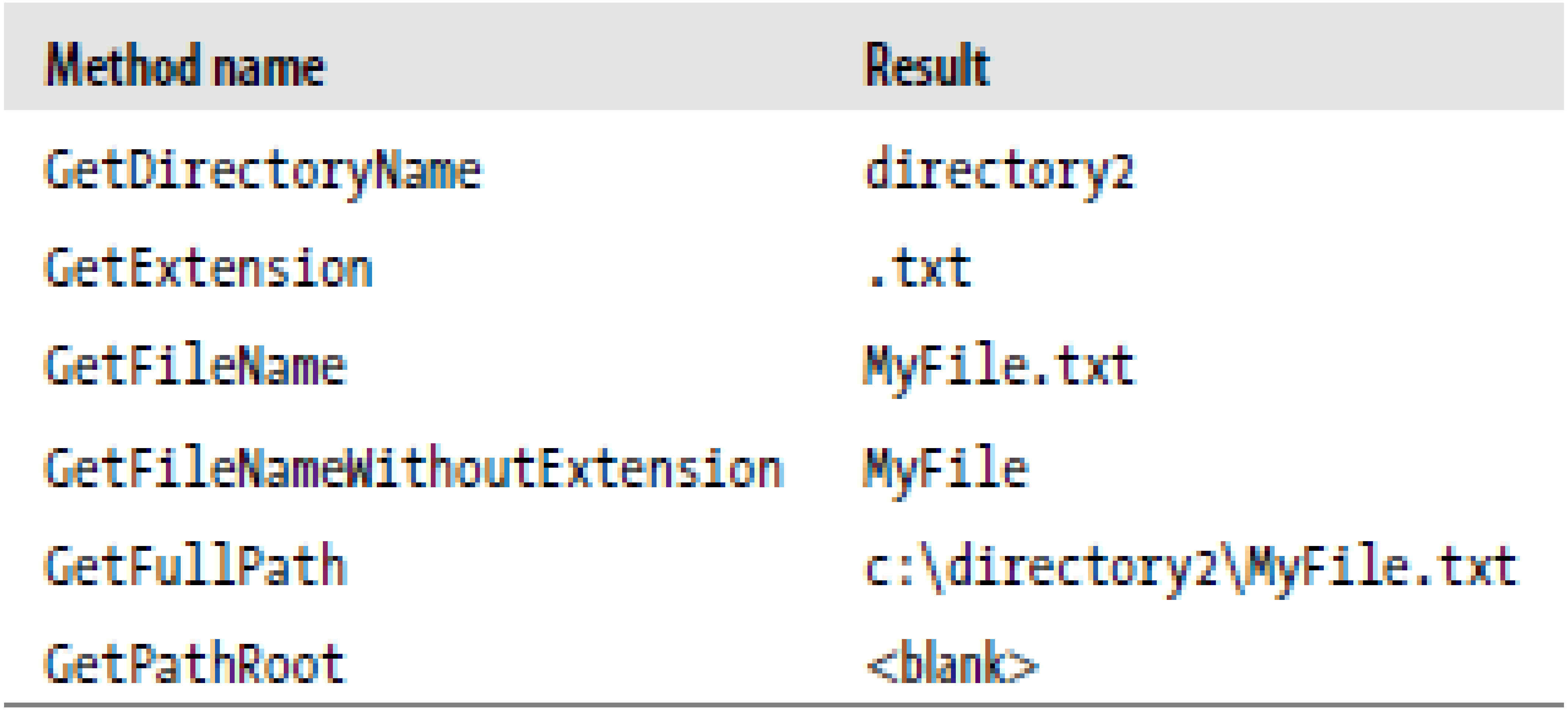
Method name	Result
GetDirectoryName	c:\directory1\directory2
GetExtension	.txt (note the leading ".")
GetFileName	MyFile.txt
GetFileNameWithoutExtension	MyFile
GetFullPath	c:\directory1\directory2\MyFile.txt
GetPathRoot	c:\

Files and Streams



Method name	Result
GetDirectoryName	\\MyPC\Share1\directory2
GetExtension	.txt
GetFileName	MyFile.txt
GetFileNameWithoutExtension	MyFile
GetFullPath	\\MyPC\Share1\directory2\MyFile.txt
GetPathRoot	\\MyPC\Share1

Files and Streams



Method name	Result
GetDirectoryName	directory2
GetExtension	.txt
GetFileName	MyFile.txt
GetFileNameWithoutExtension	MyFile
GetFullPath	c:\directory2\MyFile.txt
GetPathRoot	<blank>

Files and Streams

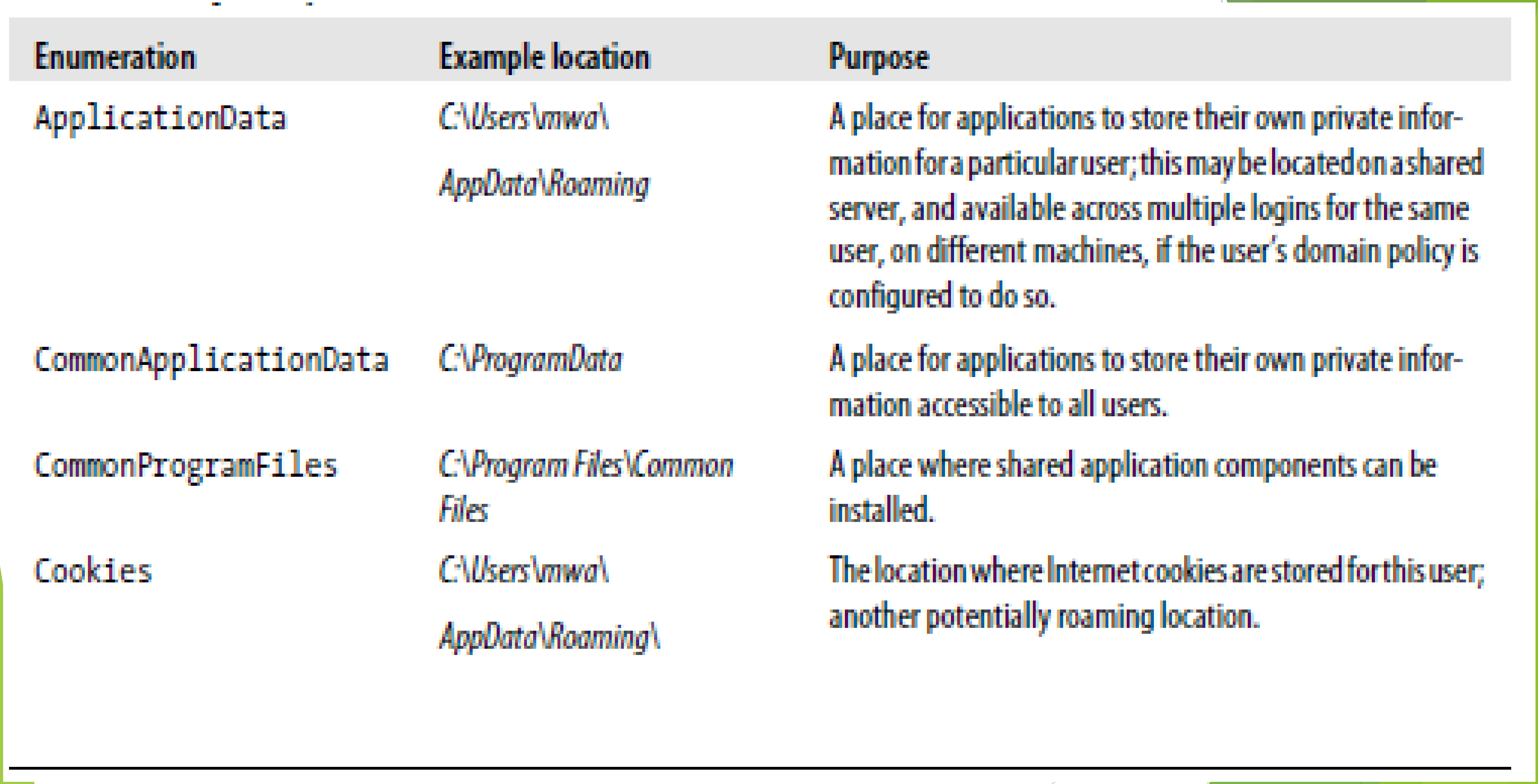
- *Getting the file size*
- Creating Temporary Files
- Deleting Files
- *Configuring access control on new directories*
- *Denying permissions*
- Deleting a Directory
- Writing a Whole Text File at Once
- *Creating a StreamWriter*
- *Writing a string with StreamWriter*

Files and Streams

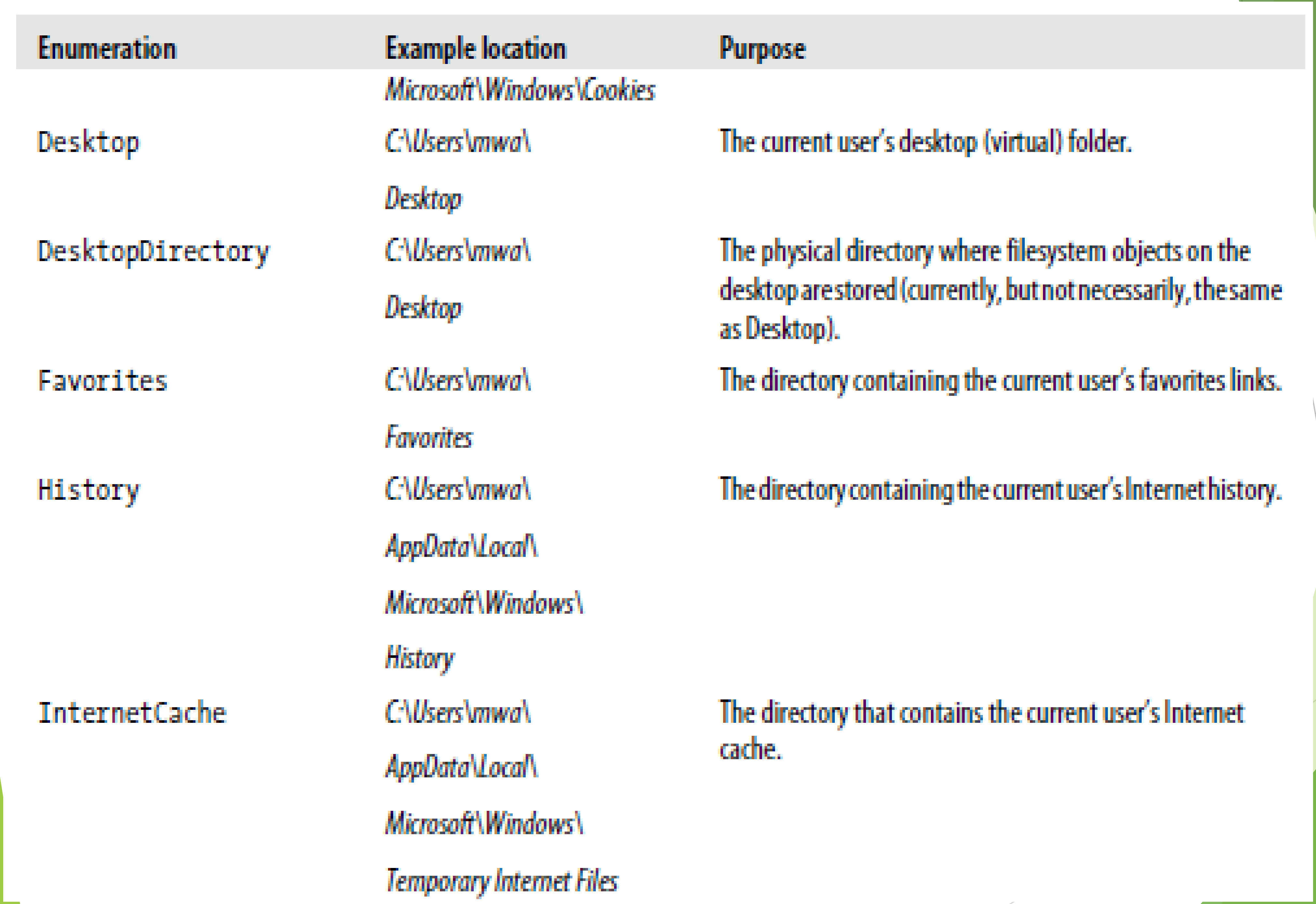
- Finding and Modifying Permissions
- *Putting exception handling in a helper method*
- *Handling exceptions from FileInfo*
- *Loading binary file content*
- *Reading from a stream*
- *Seeking within a stream*
- *Copying from one stream to another*
- *Creating folders and files in a store*
- *Using StreamReader and StreamWriter with isolated storage*

Files and Streams

- *Using an encryption stream*

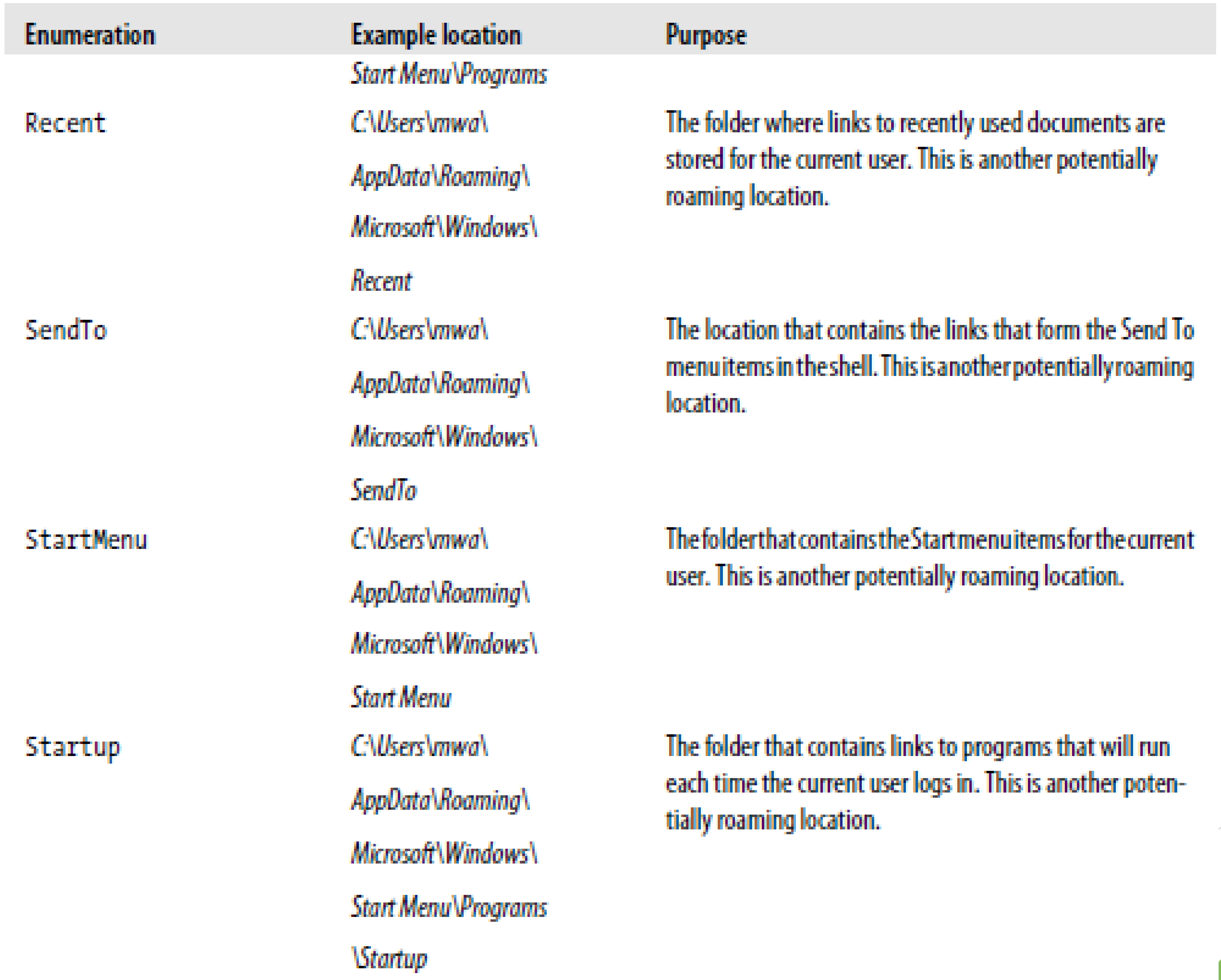


Enumeration	Example location	Purpose
ApplicationData	*C:\Users\mwa\AppData\Roaming*	A place for applications to store their own private information for a particular user; this may be located on a shared server, and available across multiple logins for the same user, on different machines, if the user's domain policy is configured to do so.
CommonApplicationData	*C:\ProgramData*	A place for applications to store their own private information accessible to all users.
CommonProgramFiles	*C:\Program Files\Common Files*	A place where shared application components can be installed.
Cookies	*C:\Users\mwa\AppData\Roaming\*	The location where Internet cookies are stored for this user; another potentially roaming location.

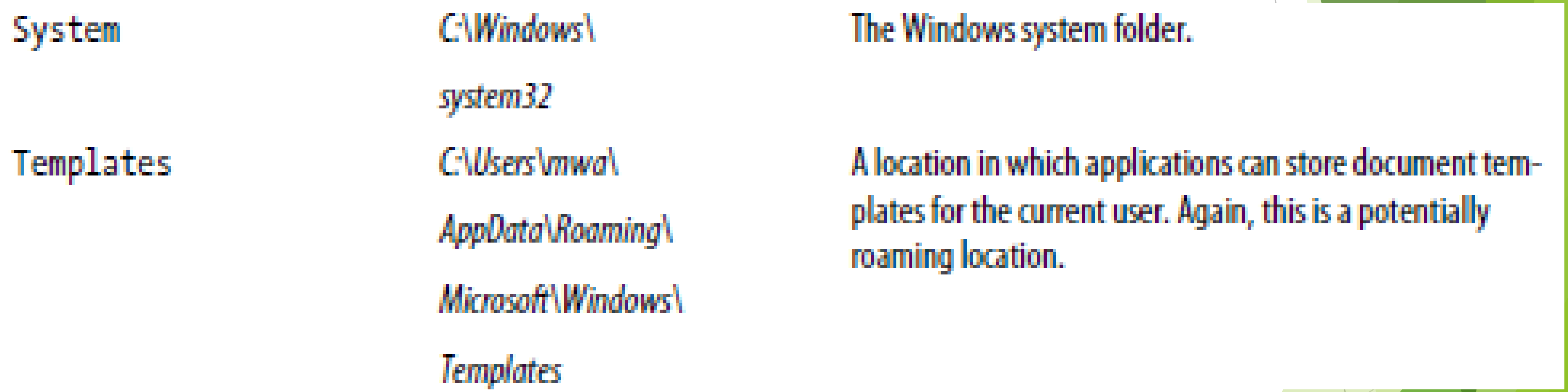


Enumeration	Example location	Purpose
	Microsoft\Windows\Cookies	
`Desktop`	*C:\Users\mwa\ Desktop*	The current user's desktop (virtual) folder.
`DesktopDirectory`	*C:\Users\mwa\ Desktop*	The physical directory where filesystem objects on the desktop are stored (currently, but not necessarily, the same as Desktop).
`Favorites`	*C:\Users\mwa\ Favorites*	The directory containing the current user's favorites links.
`History`	*C:\Users\mwa\ AppData\Local\ Microsoft\Windows\ History*	The directory containing the current user's Internet history.
`InternetCache`	*C:\Users\mwa\ AppData\Local\ Microsoft\Windows\ Temporary Internet Files*	The directory that contains the current user's Internet cache.

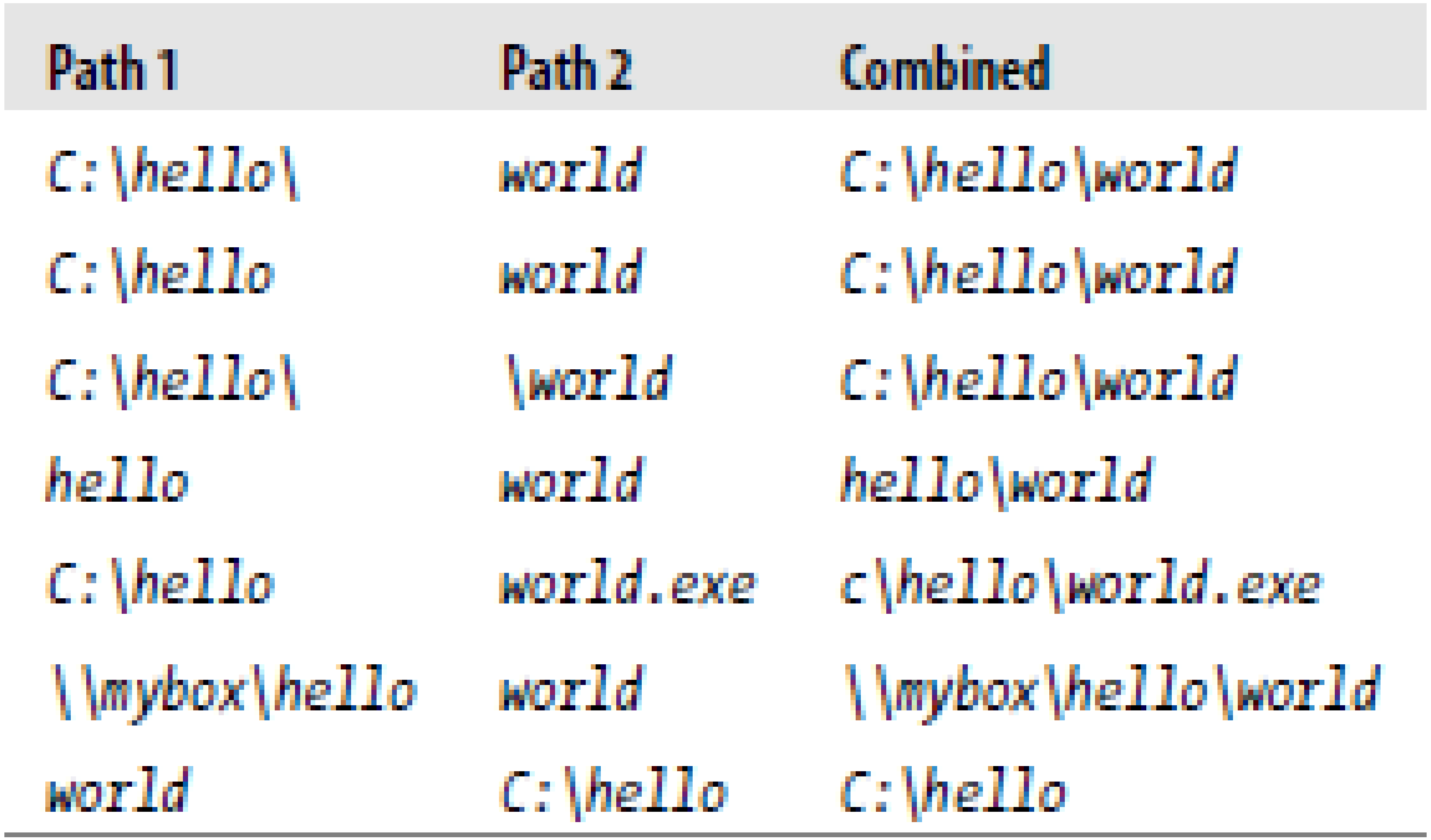
`LocalApplicationData`	*C:\Users\mwa\ AppData\Local*	A place for applications to store their private data associated with the current user. This is guaranteed to be on the local machine (as opposed to `ApplicationData` which may roam with the user).
`MyComputer`	<blank>	This is always an empty string because there is no real folder that corresponds to My Computer.
`MyDocuments`	*C:\Users\mwa\ Documents*	The folder in which the current user's documents (as opposed to private application datafiles) are stored.
`MyMusic`	*C:\Users\mwa\ Music*	The folder in which the current user's music files are stored.
`MyPictures`	*C:\Users\mwa\ Pictures*	The folder in which the current user's picture files are stored.
`Personal`	*C:\Users\mwa\ Documents*	The folder in which the current user's documents are stored (synonymous with `MyDocuments`).
`ProgramFiles`	*C:\Program Files*	The directory in which applications are installed. Note that there is no special folder enumeration for the 32-bit applications directory on 64-bit Windows.
`Programs`	*C:\Users\mwa\ AppData\Roaming\ Microsoft\Windows\*	The location where application shortcuts in the Start menu's Programs section are stored for the current user. This is another potentially roaming location.



Enumeration	Example location	Purpose
	Start Menu\Programs	
`Recent`	*C:\Users\mwa\AppData\Roaming\Microsoft\Windows\Recent*	The folder where links to recently used documents are stored for the current user. This is another potentially roaming location.
`SendTo`	*C:\Users\mwa\AppData\Roaming\Microsoft\Windows\SendTo*	The location that contains the links that form the Send To menu items in the shell. This is another potentially roaming location.
`StartMenu`	*C:\Users\mwa\AppData\Roaming\Microsoft\Windows\Start Menu*	The folder that contains the Start menu items for the current user. This is another potentially roaming location.
`Startup`	*C:\Users\mwa\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup*	The folder that contains links to programs that will run each time the current user logs in. This is another potentially roaming location.

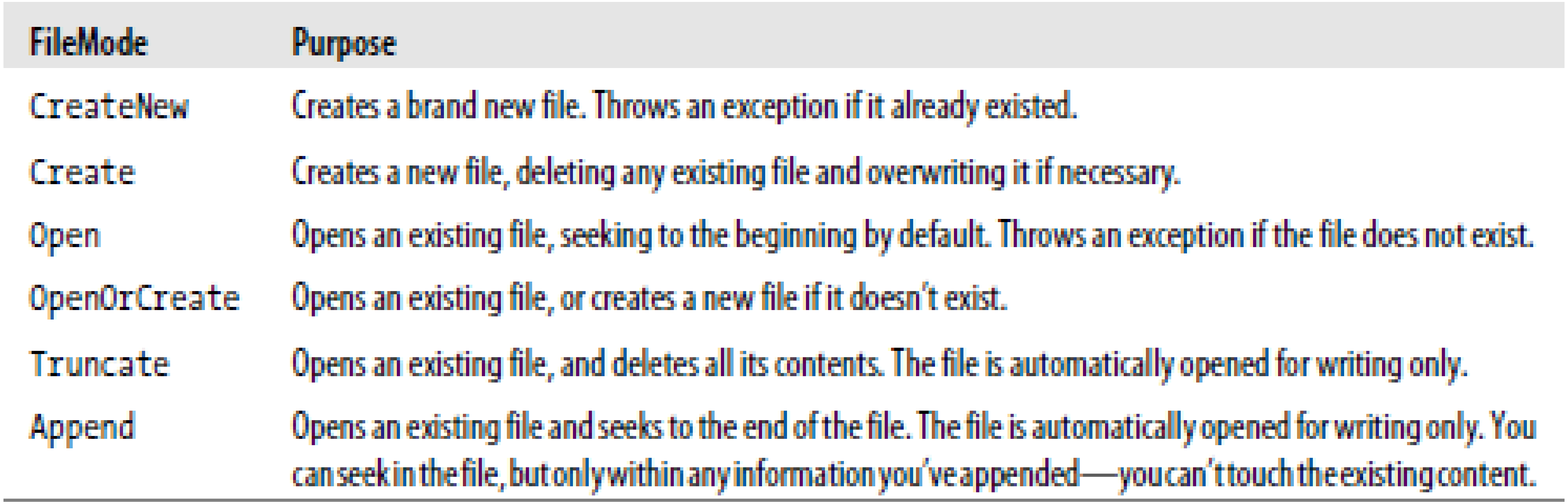


`System`	*C:\Windows\system32*	The Windows system folder.
`Templates`	*C:\Users\mwa\AppData\Roaming\Microsoft\Windows\Templates*	A location in which applications can store document templates for the current user. Again, this is a potentially roaming location.

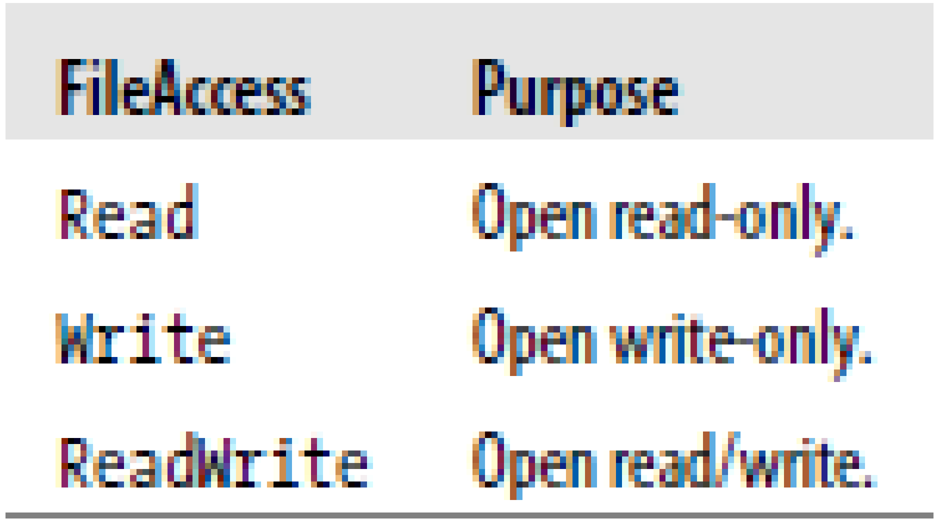


Path 1	Path 2	Combined
C:\hello\	world	C:\hello\world
C:\hello	world	C:\hello\world
C:\hello\	\world	C:\hello\world
hello	world	hello\world
C:\hello	world.exe	c\hello\world.exe
\\mybox\hello	world	\\mybox\hello\world
world	C:\hello	C:\hello

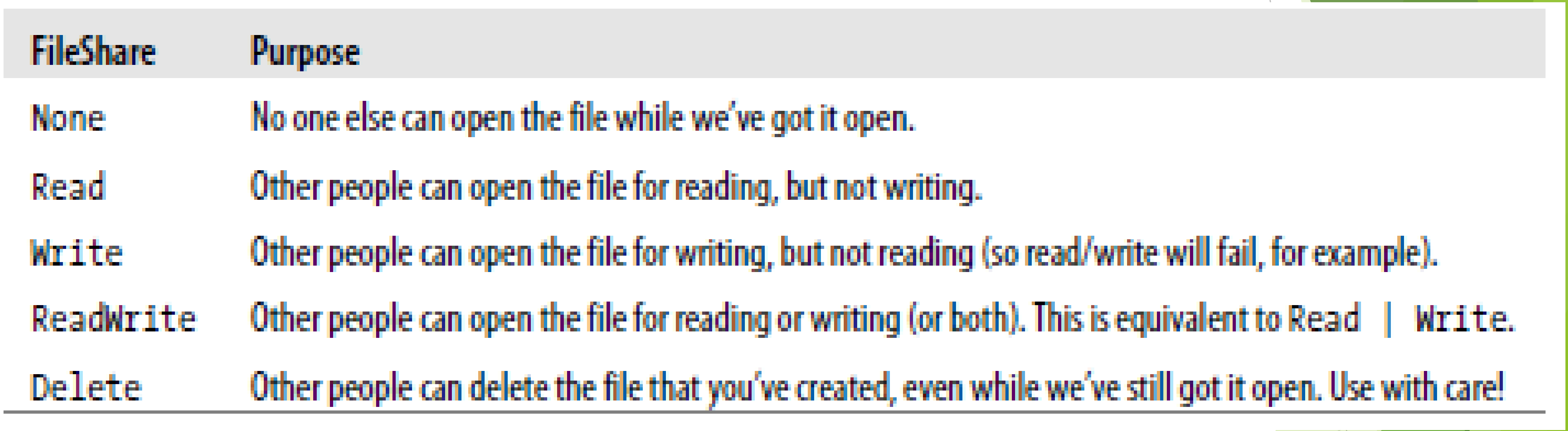
Example results of Path.Combine



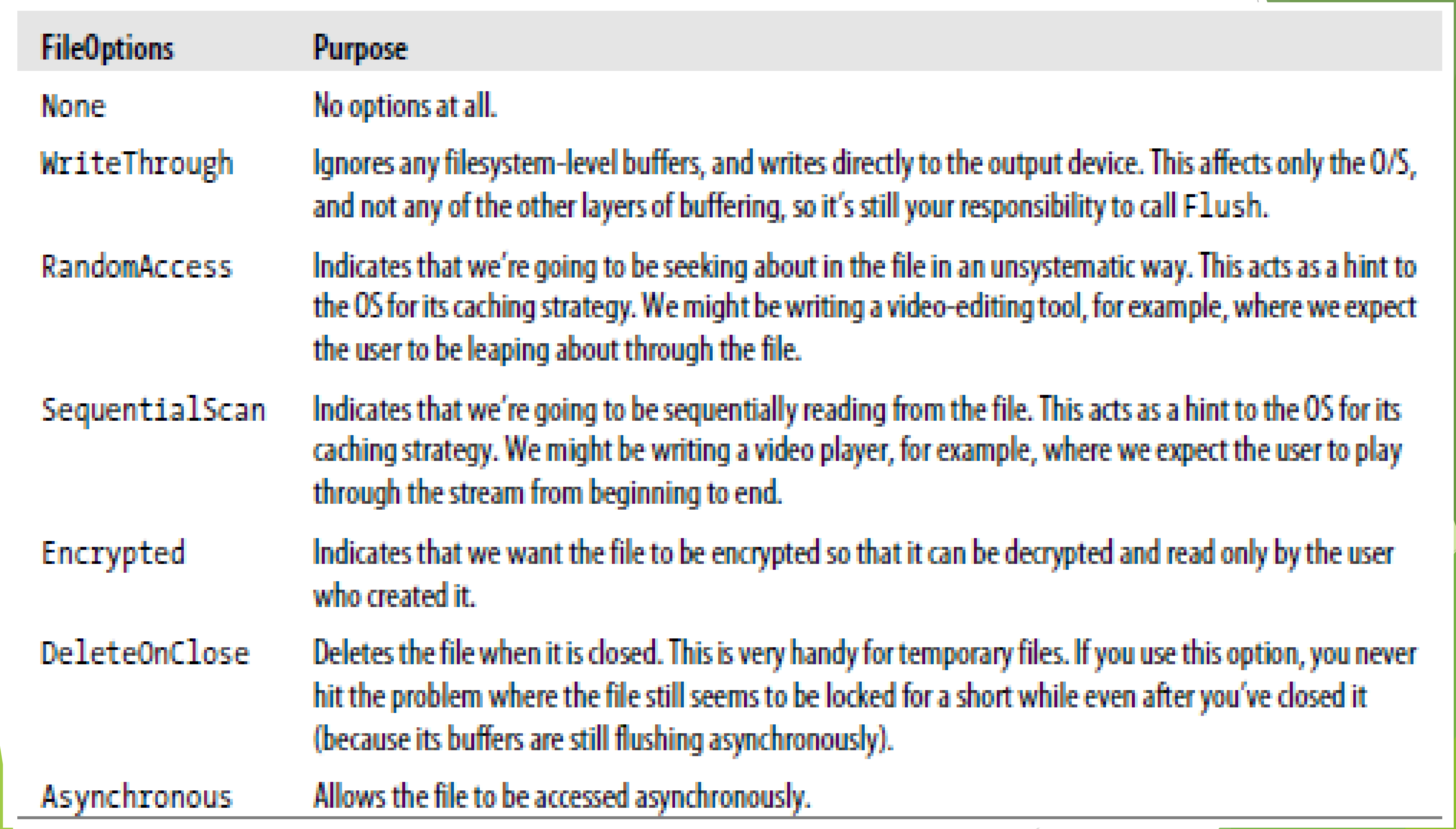
FileMode	Purpose
`CreateNew`	Creates a brand new file. Throws an exception if it already existed.
`Create`	Creates a new file, deleting any existing file and overwriting it if necessary.
`Open`	Opens an existing file, seeking to the beginning by default. Throws an exception if the file does not exist.
`OpenOrCreate`	Opens an existing file, or creates a new file if it doesn't exist.
`Truncate`	Opens an existing file, and deletes all its contents. The file is automatically opened for writing only.
`Append`	Opens an existing file and seeks to the end of the file. The file is automatically opened for writing only. You can seek in the file, but only within any information you've appended—you can't touch the existing content.



FileAccess	Purpose
Read	Open read-only.
Write	Open write-only.
ReadWrite	Open read/write.



FileShare	Purpose
None	No one else can open the file while we've got it open.
Read	Other people can open the file for reading, but not writing.
Write	Other people can open the file for writing, but not reading (so read/write will fail, for example).
ReadWrite	Other people can open the file for reading or writing (or both). This is equivalent to Read | Write.
Delete	Other people can delete the file that you've created, even while we've still got it open. Use with care!



FileOptions	Purpose
`None`	No options at all.
`WriteThrough`	Ignores any filesystem-level buffers, and writes directly to the output device. This affects only the O/S, and not any of the other layers of buffering, so it's still your responsibility to call `Flush`.
`RandomAccess`	Indicates that we're going to be seeking about in the file in an unsystematic way. This acts as a hint to the OS for its caching strategy. We might be writing a video-editing tool, for example, where we expect the user to be leaping about through the file.
`SequentialScan`	Indicates that we're going to be sequentially reading from the file. This acts as a hint to the OS for its caching strategy. We might be writing a video player, for example, where we expect the user to play through the stream from beginning to end.
`Encrypted`	Indicates that we want the file to be encrypted so that it can be decrypted and read only by the user who created it.
`DeleteOnClose`	Deletes the file when it is closed. This is very handy for temporary files. If you use this option, you never hit the problem where the file still seems to be locked for a short while even after you've closed it (because its buffers are still flushing asynchronously).
`Asynchronous`	Allows the file to be accessed asynchronously.

XML

- Elements
- XHTML
- *Creating an XML document*
- *Creating an XML document containing elements and attributes*
- *Constructing an XElement all at once*
- *Generating XML elements with LINQ*
- *Building XML with LINQ*
- Searching in XML with LINQ
- *Searching an XML document using LINQ*

XML

- *Simple XML serialization and deserialization*
- *Customizing XML serialization with attributes*