Reading and Writing Entire Files Into Memory

Jason Roberts
.NET Developer

@robertsjason | dontcodetired.com

Module Overview

- Reading entire text files into an in-memory string
- Writing an entire string to a file
- Reading entire text files into an in-memory string array
- Writing text files from string arrays
- Specifying text file encodings
- Appending text content to existing files
- Reading and writing entire contents of binary files
- Considerations of in-memory file processing

Specifying Text Encodings

File.ReadAllText(InputFilePath)

File.ReadAllLines(InputFilePath)

File encoding detection

- Byte order mark (BOM)
- UTF-8 fallback

Can also explicitly specify encoding

Specifying Text Encodings

```
File.ReadAllText(InputFilePath, Encoding);
File.ReadAllLines(InputFilePath, Encoding);

using System.Text;
```

Encoding Class Static Convenience Properties

Encoding.ASCII

- ASCII (7-bit)
- new ASCIIEncoding()

Encoding.UTF7

- UTF-7
- new UTF7Encoding()

Encoding.UTF8

- UTF-8
- new UTF8Encoding(...)

Encoding Class Static Convenience Properties

Encoding.BigEndianUnicode

- UTF-16 big endian byte order
- new UnicodeEncoding(...)

Encoding.Unicode

- UTF-16 little endian byte order
- new UnicodeEncoding(...)

Encoding.UTF32

- UTF-32 little endian byte order
- new UTF32Encoding(...)

new UTF32Encoding(true, true)

Specifying Text Encodings

```
using System.Text;

File.ReadAllText(InputFilePath, Encoding.UTF32)
File.ReadAllLines(InputFilePath, Encoding.ASCII)

// UTF-32 big endian
File.ReadAllLines(InputFilePath,
                  new UTF32Encoding(true, true));
```

Specifying Text Encodings

```
// UTF-8 encoding with no BOM
File.WriteAllText(OutputFilePath, text);
File.WriteAllLines(OutputFilePath, lines);

File.WriteAllText(OutputFilePath, text, Encoding.UTF32);
File.WriteAllLines(OutputFilePath, lines, Encoding.UTF32);

File.WriteAllText(OutputFilePath, text,
                  new UTF8Encoding(true));
```

Appending Text Content

```
// Opens existing file (or creates new file if not exist)
// Appends specified text
// Closes file

// UTF-8, no BOM
File.AppendAllText(@"C:\temp\log.txt", "error xyz");

File.AppendAllText(@"C:\temp\log.txt", "error xyz",
                   Encoding.UTF32);
```

Appending Text Content

```
IEnumerable<string> lines = new string[] {"line1", "line2"};

// Opens existing file (or creates new file if not exist)
// Appends specified lines one by one
// Closes file
// UTF-8, no BOM
File.AppendAllLines(@"C:\temp\log.txt", lines);

File.AppendAllLines(@"C:\temp\log.txt", lines, Encoding.UTF32);
```

Considerations

Benefits	Drawbacks
Simple code	May be slow
Easier to write	May crash program (out of memory)
Easy to read & maintain	No random access / seeking

Module Summary

File.ReadAllText()

File.WriteAllText()

File.ReadAllLines()

File.WriteAllLines()

Encoding.UTF32

new UTF8Encoding(true)

File.AppendAllText()

File.ReadAllBytes()

File.WriteAllBytes()

In-memory file processing considerations

Up Next:

Reading and Writing Data Incrementally Using Streams