File Handling ใน C#

Suphot Sawattiwong

tohpus@gmail.com

บทความเรื่องวุ่นของตัวอักษร โดยคุณตาหวาน

http://www.thaigamedevx.com/features/61

http://www.thaigamedevx.com/features/62

http://www.thaigamedevx.com/features/63

File คือ?

- ที่เก็บข้อมูลต่าง ๆ ของคอมพิวเตอร์
- การติดต่อกับไฟล์ต้องผ่านลิจิกคอลอินเตอร์เฟส (Logical Interfaces) ที่เรียกว่าสตรีม (Stream) สตรีมช่วยให้ผู้ใช้เขียนโปรแกรมติดต่อกับไฟล์ข้อมูล ซึ่งสตรีมที่ใช้ติดต่อกับไฟล์ ไฟล์จะมีอยู่ 2 ประเภทคือ
- **Text file** เป็นไฟล์ของตัวอักษร เพราะมีโครงสร้างในการเก็บข้อมูลจะเป็นตัวอักษรไฟล์นั้นจึงไม่สามารถที่จะเก็บข้อมูลที่ค่าตัวเลขจำนวนเต็ม จุดทศนิยม หรือในลักษณะที่เป็นโครงสร้างซึ่งการเก็บข้อมูลถูกแปลงเป็นเลขฐานสองตามรหัส ASCII, Unicode, UTF8 เป็นต้น
- **Binary File** เป็นไฟล์ที่เก็บข้อมูลที่อยู่ในรูปแบบของค่าตรง ๆ ซึ่งข้อมูลที่ไฟล์ประเภทนี้จัดเก็บ จะสามารถเป็นได้ทั้งตัวเลขจำนวนเต็ม ตัวเลขทศนิยม ตัวอักษร อาร์เรย์ และข้อมูลแบบโครงสร้าง โดยการจัดเก็บนั้นจะเก็บลงไปตรง ๆ เลย

System.IO

- การจัดการกับ File ใน C# นั้น จำเป็นต้องขอใช้ namespace ที่ชื่อ System.IO โดยทำการประกาศบนหัว class ดังนี้

```
using System.IO;
```

เขียนไฟล์ StreamWriter

- จากที่บอกก่อนหน้านี้ จำเป็นต้องใช้ stream ในการอ่านและเขียนไฟล์
- StreamWriter เป็นclass สำหรับเปิดไฟล์มาเพื่อเขียนไฟล์โดยตรง
- หากต้องการอ่านเพิ่มเติมได้ที่

http://msdn.microsoft.com/en-us/library/system.io.streamwriter.aspx

StreamWriter Constructor

StreamWriter(String)	เป็นการสร้าง instance ของ StreamWriter โดยเป็นการระบุชื่อไฟล์ลงไปใน parameter string และเปิดด้วย default encoding โดยสร้างไฟล์ใหม่ทุกครั้ง หากมีไฟล์อยู่แล้วจะทำการทับไฟล์เดิม
StreamWriter(String, Boolean)	เป็นการสร้าง instance ของ StreamWriter โดยระบุชื่อไฟล์ ลงไปใน parameter string และ parameter Boolean เป็นการบอกว่าจะให้ทำการเพิ่มลงไปในไฟล์ที่มีอยู่แล้วหรือไม่ ถ้าเป็น true เป็นการอนุญาตให้เพิ่มลงไปในไฟล์เดิม
StreamWriter(String, Boolean, Encoding)	เป็นการสร้าง instance ของ StreamWriter โดยระบุชื่อไฟล์ ลงไปใน parameter string, parameter Boolean เป็นการบอกว่าจะให้ทำการเพิ่มลงไปในไฟล์ที่มีอยู่แล้วหรือไม่ ถ้าเป็น true เป็นการอนุญาตให้เพิ่มลงไปในไฟล์เดิม, และส่วน parameter Encoding เป็นระบุรหัส Encoding เช่น ASCII, UTF8, Unicode เป็นต้น

ตัวอย่าง

```
StreamWriter tw = new StreamWriter("data.txt");
```

หรือ

```
StreamWriter tw = new StreamWriter("data.txt", true);
```

หรือ

```
StreamWriter tw = new StreamWriter("data.txt", true, System.Text.Encoding.ASCII);
```

Character Encoding

- รหัสตัวอักษร (Character Code) รหัสตัวอักษรก็คือรหัสตัวเลขที่คอมพิวเตอร์ใช้เพื่อแทนตัวอักษรนั่นเอง
- การเข้ารหัส (Encoding) การเข้ารหัสจะเป็นการกำหนดว่ารหัสที่ใช้นั้นสร้างขึ้นมายังไง เช่น อาจจะเป็นตัวเลขชุดเดียว บอกว่าเป็นตัวอักษรที่เท่าไหร่ นับจากตัว A หรืออาจจะเป็นสองชุด
- Encoding มีมาตรฐานให้ใช้อยู่มากมาย มาตรฐานที่ควรรู้จักก็ได้แก่ ASCII, UTF8, ส่วนมาตรฐานของไทยก็มี มอก.620 (tis620) เป็นต้น และมาตรฐานที่ใช้ร่วมกันทั่วโลกอย่าง Unicode

Character Encoding (ต่อ)

- String คือ ชุดของตัวอักษรที่เรียงต่อกันเป็นแนวยาว
- ลักษณะพิเศษของ String ก็คือ มันเป็นข้อมูลแบบ Stream ที่ไหลไปทางเดียวกันตลอด จะสังเกตุได้ว่าเวลาเราอ่านตัวหนังสือเราจะไม่ค่อยมีการอ่านย้อนกลับมากนัก (การอ่านข้อมูลในทิศทางเดียวเป็นลักษณะเฉพาะของ Stream) ดังนั้นเราจะมองว่า String คือ Stream ของ Character ก็ได้เหมือนกัน

Character Encoding (ต่อ)

Fixed-Length Character Encoding (การเข้ารหัสโดยใช้รหัสที่มีจำนวนหลักตายตัว)

- การเข้ารหัสในลักษณะนี้จะเป็นการแทนที่ตัวอักษรใด ๆ ด้วยตัวเลขที่มีจำนวนหลัก (digit) แบบคงที่ เช่น การเข้ารหัสด้วยตัวเลข 8หลัก (8บิท) 16หลัก หรือ 32 หลัก
- ASCII เป็น การเข้ารหัสโดยใช้รหัสแบบ 8 บิท (หรือ 1ไบท์) โดยที่หลักสุดท้ายนั้นจะเป็น 0 เสมอ โดยเราจะนับหลักหน่วย (หลักขวาสุด) เป็นหลักแรกนะครับ ดังนั้นจะมีจำนวนรหัสที่เป็นไปได้ทั้งหมดที่ 128 รหัส (ก็คือ 0-127 นั่นเอง) ทั้งยังเป็นมาตรฐานที่รวบรวมเอาตัวอักษรและสัญลักษณ์ในภาษาอังกฤษ ซึ่งรวมถึงรหัสอักขระพิเศษที่ไม่ได้ใช้แสดงผลเอาไว้ด้วย (รหัสอักขระพิเศษ คือรหัสอักขระที่มีหน้าที่พิเศษ เช่น ตัดบรรทัด ขึ้นย่อหน้าใหม่ เป็นต้น เป็นอักขระที่ไม่ได้ใช้ในการแสดงผล) ASCII เป็นมาตรฐานแรก ๆ ในโลกที่ยังคงใช้จนถึงปัจจุบันนี้
- Extended ASCII เป็นการเพิ่มเติมส่วนขยายให้แก่ ASCII โดยจะมีการใช้ 1 บิทสุดท้าย (ที่ไม่ได้ใช้ใน ASCII) เพื่อเพิ่มจำนวนรหัสให้มากขึ้นเป็น 256 รหัส เพื่อที่จะเพิ่มรหัสสำหรับตัวอักษรที่ไม่ได้ใช้ในภาษาอังกฤษนั่นเอง
- ตัวอย่างสำหรับมาตรฐาน Extended ASCII ก็เช่น ISO8859-1 (ชุดตัวอักษรลาติน) TIS620 (ชุดตัวอักษรภาษาไทย) เป็นต้น

Character Encoding (ต่อ)

- Variable-Length Character Encodeing (การเข้ารหัสโดยใช้รหัสที่มีจำนวนหลักไม่คงที่)
- คือ รหัสแต่ละตัวอาจจะสั้นยาวไม่เท่ากัน ตามแต่ผู้ออกแบบกำหนด โดยส่วนใหญ่จะใช้วิธีกำหนดช่วงเอาไว้ว่า รหัสในช่วงใดช่วงนึงจะมีความยาวที่ระดับนึง ในขณะที่ในอีกช่วงนึงก็จะมีความยาวที่อีกระดับนึง เช่น ตัวอักษร 128 ตัวแรก ใช้รหัสที่ยาว 8 หลัก (8บิท หรือ 1 ไบท์) ในขณะที่ตัวที่เหลือจะใช้ความยาวที่ 16 หลัก (16 บิท หรือ 2 ไบท์) เป็นต้น
- การเข้ารหัสแบบนี้เกิดขึ้นในสมัยที่หน่วยความจำยังมีราคาแพง แต่ จำนวนตัวอักษรที่ Extended ASCII รองรับนั้นไม่เพียงพอต่อการใช้งานในบางภาษ เช่น ในภาษาจีนมีอักขระที่แตกต่างกันถึงสี่พันตัว ภาษาญี่ปุ่นที่มีชุดอักขระมากถึงสามชุด เป็นต้น แต่การที่จะไปใช้การเข้ารหัสแบบ 16 บิทนั้นใช้หน่วยความจำมาถึง 2 เท่าของปรกติ และ ซอฟท์แวร์ที่ใช้จะเข้ากับซอฟท์ดั้งเดิมที่ใช้การเข้ารหัสแบบ 8 บิทไม่ได้ จึงมีการพัฒนาการเข้ารหัสแบบนี้ขึ้นมาครับ
- สามภาษาที่เป็นต้นเหตุของการเข้ารหัสแบบนี้นั้น เป็นประเทศในเอเซียตะวันออกทั้งหมดเลย นั่นคือ จีน ญี่ปุ่น และเกาหลี เราจะเรียกกลุ่มเประเทศเรื่องมากพวกนี้ว่า กลุ่ม CJK ครับ
- ตัวอย่างของการเข้ารหัสแบบนี้ก็คือ BIG5 (Chinese), Shift-JIS(Japanese), EUC-KR (Korean) เป็นต้น

Character Encoding (ต่อ)

- UTF-8 เป็นการเข้ารหัสที่กำหนดโดย UNICODE การเข้ารหัสแบบนี้จะใช้รหัสที่สั้นที่สุดที่ 8 บิท (1ไบท์) จนถึงยาวที่สุดที่ 4ไบท์ โดยตัวที่จะบอกว่ารหัสนั้นยาวแค่ไหนก็คือบิทสุดท้ายของไบท์แรกที่อ่าน เช่น
  - ถ้าหลักสุดท้ายของไบท์แรก คือ 0 รหัสนี้จะยาว 8บิท
  - ถ้า 3 หลักสุดท้ายของไบท์แรก คือ 110 รหัสนี้จะยาว 16บิท
  - ถ้า 4 หลักสุดท้ายของไบท์แรก คือ 1110 รหัสนี้จะยาว 32บิท

Character Encoding (ต่อ)

- Fixed-Length ข้อดีคือ มันง่ายที่จะ Implementเพราะรหัสทุกตัวจะยาวเท่ากันหมด ส่วนข้อเสียคือมันใช้พื้นที่เยอะกว่า
- Variable-Length นั้น จะใช้พื้นที่น้อยกว่า โดยเฉพาะถ้าเราเอาตัวอักษรที่ใช้บ่อยมาอยู่ในส่วนที่ใช้รหัสสั้น ๆ ได้ แต่ข้อเสียคือมันค่อนข้างลำบากกว่าที่จะ Implement

Encoding ใน StreamWriter

- การเรียกดูว่า StreamWriter ได้เปิดไฟล์โดยใช้ Encoding แบบใด โดย

```
<ชื่อตัวแปร   StreamReader>. Encoding;
```

- เช่น

```
Console.WriteLine("Encoding is {0}", tw.Encoding.ToString());
```

การเขียนลงไฟล์ด้วยคำสั่ง Write กับ WriteLine

- คำสั่ง Write กับ WriteLine ของ StreamWriter มีลักษณะคล้ายกับ Write กับ WriteLine ของ Console นั่นเอง
- เพียงแต่ว่าWrite กับ WriteLine ของ Console จะแสดงออกทางจอภาพ
- แต่ Write กับ WriteLine จะนำเข้าไปบันทึกในไฟล์นั่นเอง

```
tw.WriteLine("Hello{0}",i);
```

ทำการปิดไฟล์ เมื่อใช้เสร็จแล้ว

- ใน StreamWriter นั้น หากมีการเปิดใช้ และเขียนข้อมูลเสร็จเรียบร้อยแล้วควรทำการปิดไฟล์ด้วย โดยใช้คำสั่ง Close(); มิฉะนั้นไฟล์จะบันทึกไม่สมบูรณ์

```
<ชื่อตัวแปร   StreamReader>.Close();
```

เช่น

```
tr.Close();
```

ตัวอย่างการเขียนไฟล์

```
using System;
using System.IO;
namespace FileExample
{
   class Program
   {
      static void Main(string[] args)
      {
         // create a writer and open the file
         TextWriter tw = new StreamWriter("data.txt");
         Console.WriteLine("Encoding is {0}", tw.Encoding.ToString());
         // write a line of text to the file
         for (int i = 0; i < 100; i++)
         {
            tw.WriteLine("Hello{0}",i);
         }
         // close the stream
         tw.Close();
         Console.WriteLine("Write File Complete!!!");
         Console.ReadLine();
      }
   }
}
```

อ่านไฟล์ด้วย StreamReader

- StreamReader เป็น class ที่มีหน้าที่ในการเปิดและอ่านไฟล์

```
StreamReader   <ชื่อตัวแปร>= new StreamReader(<ชื่อไฟล์>);
                                หรือ
StreamReader   <ชื่อตัวแปร>= new StreamReader(<ชื่อไฟล์>, <Encoding>);
```

เช่น

```
StreamReader tr = new StreamReader("data.txt");
                                หรือ
StreamReader tr = new StreamReader("data.txt",  System.Text.Encoding.ASCII);
```

- หากต้องการอ่านเกี่ยวกับ StreamReader เพิ่มเติมสามารถอ่านได้จาก http://msdn.microsoft.com/en-us/library/system.io.streamreader.aspx

CurrentEncoding ใน StreamReader

- เป็นคำสั่งเพื่อบอกว่า StreamReader ได้ใช้ Encoding แบบใด
- โดยใช้คำสั่งดังนี้

```
<ชื่อตัวแปร stream reader>.CurrentEncoding
```

- เช่น

```
Console.WriteLine(tr.CurrentEncoding.ToString());
```

การอ่านข้อมูลจากไฟล์ เข้ามาในตัวแปร

- สามารถใช้คำสั่งได้ดังนี้

Read	อ่านตัวอักษรจากไฟล์ที่เปิดอ่าน และทำการเลื่อนไปยังตัวถัดไป และส่งค่าออกมาเป็น int
ReadLine	อ่านตัวหนังสือทั้งบรรทัด ทั้งเลื่อนไปยังบรรทัดถัดไป จากนั้นส่งค่ากลับเป็น string ออกมา

ทำการปิดไฟล์ เมื่อใช้เสร็จแล้ว

- ใน StreamReader นั้น หากมีการเปิดใช้ และอ่านข้อมูลเสร็จเรียบร้อยแล้วควรทำการปิดไฟล์ด้วย โดยใช้คำสั่ง Close();

```
<ชื่อตัวแปร   StreamReader>.Close();
```

เช่น

```
tr.Close();
```

ตัวอย่างการอ่านไฟล์

```
using System;
using System.IO;
namespace FileExample
{
   class Program
   {
      static void Main(string[] args)
      {
         StreamReader tr = new StreamReader("data.txt");
         Console.WriteLine(tr.CurrentEncoding.ToString());
         string temp = "";
         while ((temp = tr.ReadLine()) != null)
         {
            Console.WriteLine(temp);
         }
         tr.Close();
         Console.ReadLine();
      }
   }
}
```

ตัวอย่างการอ่านไฟล์

```
using System;
using System.IO;
namespace FileExample
{
   class Program
   {
      static void Main(string[] args)
      {
         StreamReader tr = new StreamReader("data.txt",
                                                System.Text.Encoding.ASCII);
        Console.WriteLine(tr.CurrentEncoding.ToString());
         string temp = "";
         while ((temp = tr.ReadLine()) != null)
         {
            Console.WriteLine(temp);
         }
         tr.Close();
         Console.ReadLine();
      }
   }
}
```

การตรวจสอบไฟล์ว่ามีไฟล์อยู่หรือไม่

- หากต้องการตรวจสอบไฟล์ว่ามีอยู่หรือไม่ ให้ใช้คำสั่งดังนี้

```
File.Exists(<ชื่อไฟล์>)
```

เช่น

```
if (File.Exists(fileName))
{
        Console.WriteLine(fileName + " already exists!");
        return;
}
```