C# 温故而知新：Stream篇（**二**）TextReader 和StreamReader

为什么要介绍 TextReader？

首先让我们来理解下什么是TextReader，从字面上的意思入手的话，大家就会恍然大悟了

一个对于Text的读取器，可是又是怎么读取的呢？聪明的你肯定会想到，当然是通过连续的字符进行读取， 为什么在介绍StreamReader之前，要搞这个东东？ 答案其实很简单：他们两个就是父子关系，要了解StreamReader最好先了解他的父亲，请允许我对他们进行下简单介绍：

TextReader的常用属性和方法:

我们闭上眼可以想象一下 Text这个词的范围，它囊括了许多的文件类型，我们可以在记事本上使用任何语言(英语，中文，c# ,天书，javascript,jquery,xml,xaml,sql,c++……)，如此多的语言文本归根结底还是通过一个个char组成的，所以微软构造出了TextReader这个抽象类对于

读取text的一系列操作，同样对于TextReader我们无法直接实例化，应为它是个抽象类，只有定义类的行为，不针对特定实现。那好吧，看看 TextReader定义了哪些类的行为:

1:具有一个protected类型的构造函数

\*2: void Close()方法：和上篇Stream一样，TextReader也有Close方法，我们必须牢记，在用完之后应该主动关闭它

\*3: void Dispose()方法：释放所有该TextReader 所持有的所有资源(注意，假如TextReader中持有stream或其他

对象，当TextReader执行了Dispose方法时，stream对象也被回收了)

\*4:int Peek()方法： 这个方法主要是寻找当前char的下个 char,当返回值是-1时，表示下个 char已经是最后一个位置的char了

\*5：int Read()方法： 同样，read()方法是读取下一个char, 但是和peek方法不同，read()方法使指针指向下个字符，但是peek 还是指向原来那个字符

\*6：int Read(Char[] buffer,int index,int count)方法：  这个重载read方法和上一章Stream的read方法有点神似，区别是一个参数是byte数组，而这个是char数组，（注意:是通过reader 将数据数据读入buffer数组）,index:从哪个位置开始，count:读取char数量

\*7: int ReadBlock(Char[] buffer,int index,int count)方法：  和Read方法基本一致，区别是从效率上来说ReadBlock更高点，而且ReadBlock并非属于线程安全，使用时要注意

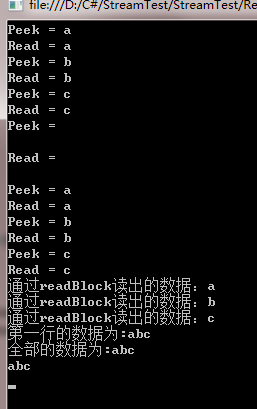
\*8：virtual string ReadLine() 方法：   顾名思义，这个方法将读取每一行的数据并返回当前行的字符的字符串

\*9：virtual string ReadToEnd()方法：   包含从当前位置到 TextReader 的结尾的所有字符的字符串

下面的例子将是对上述方法的温故而知新：

string text = "abc\nabc";  
 using (TextReader reader = new StringReader(text)){  
 while (reader.Peek() != -1){  
 Console.WriteLine("Peek = {0}", (char)reader.Peek());  
 Console.WriteLine("Read = {0}", (char)reader.Read());  
 }  
 reader.Close();  
 }  
  
 using (TextReader reader = new StringReader(text)){  
 char[] charBuffer = new char[3];  
 int data = reader.ReadBlock(charBuffer, 0, 3);  
 for (int i = 0; i < charBuffer.Length; i++){  
 Console.WriteLine("通过readBlock读出的数据：{0}", charBuffer[i]);  
 }  
 reader.Close();  
 }  
  
 using (TextReader reader = new StringReader(text)){  
 string lineData = reader.ReadLine();  
 Console.WriteLine("第一行的数据为:{0}", lineData);  
 reader.Close();  
 }  
  
 using (TextReader reader = new StringReader(text)){  
 string allData = reader.ReadToEnd();  
 Console.WriteLine("全部的数据为:{0}", allData);  
 reader.Close();  
 }  
 Console.ReadLine();

输出结果：



StreamReader

从StreamReader想到多态

在说明StreamReader之前还有一件事要提起，那就是多态, 多态到底是什么概念呢？聪明的你肯定会想到多态不就是子类的多种

表现形式？不错,但这还是不是完全的，不仅如此，现实世界中，父亲帮儿子买了套房子，但是他没有在房产证上写儿子的名字，

所以这个房子儿子和父亲能共同使用，儿子能根据自己的爱好装修房子，父亲也能住在儿子装修好的房子内，也就是说父类能够

灵活使用子类的功能，更科学的一点就是子类的指针允许（赋值给）父类指针。上述例子中

TextReader reader = new StringReader(text)

 这个就是个多态的经典例子大家不妨深刻理解下这个重要的概念

简单介绍下Encoding编码

为什么要简单介绍Encoding编码？因为Encoding编码在Stream和相关类中起的非常重要的作用，由于Encoding类会在后续章节详细解释，现在我就先介绍下 Encoding类一些重要编码



以上便是Encoding类中一些特定的编码，大家先了解即可，但使用Default时有点必须注意，如果你用不一样编码的机器的时候，注意服务器或者其他操作系统的编码规则，举个例子，如果你在一个中文操作系统进行编码，但是发布到了一个其他语言的操作系统上那就会出问题了这时候你必须选择一个通用编码

StreamReader 类的定义和作用

StreamReader 的定义：实现一个 [TextReader](http://msdn.microsoft.com/zh-cn/library/system.io.textreader.aspx)，使其以一种特定的编码从字节流中读取字符。

在对于流的操作中，StreamReader对于流的读取方面非常重要,为什么这么说呢，我们常用的文件的复制，移动，上传，下载，压缩，保存，

远程FTP文件的读取，甚至于HttpResponse等等只要是于流相关的任何派生类StreamReader 都能够轻松处理，当然，大家甚至可以自定义

相关的派生类去实现复杂的序列化。在实际项目，我们可能碰到过许多上述的情况，有时乱码的问题会让我们发狂，但是只要深刻去理解基础的话，我相信大家都能找到适合自己的解决方法

StreamReader 类的常用属性及方法

其实StreamReader的一些方法已经在其父类TextReader中说的很仔细了，但是个人觉得构造函数和属性才是重点.

**首先上构造函数**：

\*1: StreamReader(Stream stream)： 将stream作为一个参数 放入StreamReader，这样的话StreamReader可以对该stream进行读取操作,Stream对象可以非常广泛，包括所有Stream的派生类对象

\*2: StreamReader(string string, Encoding encoding)： 这里的string对象不是简单的字符串而是具体文件的地址,然后根据用户选择编码去读取流中的数据

\*3: StreamReader(string string，bool detectEncodingFromByteOrderMarks) ：有时候我们希望程序自动判断用何种编码去读取，这时候detectEncodingFromByteOrderMarks这个参数就能起作用了，当设置为true的时候数通过查看流的前三个字节来检测编码。如果文件以适当的字节顺序标记开头，该参数自动识别 UTF-8、Little-Endian Unicode 和 Big-Endian Unicode 文本，当为false 时，方法会去使用用户提供的编码

\*4: StreamReader(string string, Encoding encoding, bool detectEncodingFromByteOrderMarks,int bufferSize)：这个放提供了4个参数的重载，前3个我们都已经了解，最后个是缓冲区大小的设置，

\*StreamReader 还有其他的一些构造函数，都是上述4个的扩充，所以本例就取上述的4个构造函数来说明

**属性:**

1:BaseStream

  大家对于前一章流的操作应该没什么问题，我就直切主题，最简单的理解就是将上述构造函数的流对象在重新取出来进行一系列的操作，

  可是如果构造函数中是路径怎么办，一样，构造函数能够将路径文件转化成流对象

FileStream fs = new FileStream ( "D:\\TextReader.txt", FileMode.Open , FileAccess.Read ) ;   
StreamReader sr= new StreamReader ( fs ) ;

//本例中的BaseStream就是FileStream  
sr.BaseStream.Seek (0 , SeekOrigin.Begin ) ;

 2:CurrentEncoding：获取当前StreamReader的Encoding

3:EndOfStream：判断StreamReader是否已经处于当前流的末尾

最后用FileStream的示例来温故下StreamReader

static void Main(string[] args){  
 //文件地址  
 string txtFilePath="D:\\TextReader.txt";  
 //定义char数组  
 char[] charBuffer2 = new char[3];  
  
 //利用FileStream类将文件文本数据变成流然后放入StreamReader构造函数中  
 using(FileStream stream = File.OpenRead(txtFilePath)){  
 using (StreamReader reader = new StreamReader(stream)){  
 //StreamReader.Read()方法  
 DisplayResultStringByUsingRead(reader);  
 }  
 }  
  
 using (FileStream stream = File.OpenRead(txtFilePath)){  
 //使用Encoding.ASCII来尝试下  
 using (StreamReader reader = new StreamReader(stream,Encoding.ASCII,false)){  
 //StreamReader.ReadBlock()方法  
 DisplayResultStringByUsingReadBlock(reader);  
 }  
 }  
  
 //尝试用文件定位直接得到StreamReader，顺便使用 Encoding.Default  
 using(StreamReader reader = new StreamReader(txtFilePath, Encoding.Default,false,123)){  
 //StreamReader.ReadLine()方法  
 DisplayResultStringByUsingReadLine(reader);  
 }  
  
 //也可以通过File.OpenText方法直接获取到StreamReader对象  
 using (StreamReader reader = File.OpenText(txtFilePath)) {  
 //StreamReader.ReadLine()方法  
 DisplayResultStringByUsingReadLine(reader);  
 }  
 Console.ReadLine();  
 }  
  
 /// <summary>使用StreamReader.Read()方法</summary>  
 /// <param name="reader"></param>  
 public static void DisplayResultStringByUsingRead(StreamReader reader) {  
 int readChar = 0;  
 string result = string.Empty;  
 while ((readChar=reader.Read()) != -1) {  
 result += (char)readChar;  
 }  
 Console.WriteLine("使用StreamReader.Read()方法得到Text文件中的数据为 : {0}", result);  
 }  
  
 /// <summary>使用StreamReader.ReadBlock()方法</summary>  
 public static void DisplayResultStringByUsingReadBlock(StreamReader reader){  
 char[] charBuffer = new char[10];  
 string result = string.Empty;  
 reader.ReadBlock(charBuffer,0,10);  
 for (int i = 0; i < charBuffer.Length; i++){  
 result += charBuffer[i];  
 }  
 Console.WriteLine("使用StreamReader.ReadBlock()方法得到Text文件中前10个数据为 : {0}", result);  
 }  
  
 /// <summary>使用StreamReader.ReadLine()方法</summary>  
 public static void DisplayResultStringByUsingReadLine(StreamReader reader){  
 int i=1;  
 string resultString = string.Empty;  
 while ((resultString=reader.ReadLine() )!= null){  
 Console.WriteLine("使用StreamReader.Read()方法得到Text文件中第{1}行的数据为 : {0}", resultString, i);  
 i++;  
 }   
 }

 输出结果：

