**CultureInfo简述**

CultureInfo类位于System.Globalization命名空间内，这个类和这个命名空间许多人都不了解也认为不需要太多了解，实际上，你写的程序中会经常间接得使用这些类。

简单的说：当进行数字，日期时间，字符串匹配时，都会进行CultureInfo的操作，也就是不同的CultureInfo下，这些操作的结果可能会不一样。这里要介绍一下非常容易被忽视的InvariantCulture。

[返回目录](http://www.cnblogs.com/mgen/archive/2011/05/05/2037603.html#_hContent)

**通过示例了解InvariantCulture**

前面提到过，不同的CultureInfo会影响某些函数的执行结果，.NET中有一个特殊的 CultureInfo：InvariantCulture，这个CultureInfo有点像英语格式，但它不和国家地区挂钩，它可以提供一个可靠的在 多语言环境下的规范格式化。

比如你编写一个程序，要向数据中心服务器传递一些时间数据，你会怎么写？直接DateTime.ToString()？那你就大错特错了，下面用代码，举个非常形象的例子。在一个控制台里，模拟数据中心，然后放出多个线程，模拟客户端程序传递数据。

static readonly string[] CultureSources = { "en-us", "zh-cn", "ar-iq", "de-de" };

static readonly Random Ran = new Random(Environment.TickCount);

static void Main()

{

    Console.WriteLine("数据中心开始接受客户端数据：");

    for (int i = 0; i < CultureSources.Length; i++)

        ThreadPool.QueueUserWorkItem(Client, i);

    Console.ReadKey(true);

    Console.WriteLine("");

    Console.WriteLine("数据中心：…………");

}

static void Client(object obj)

{

    int id = (int)obj;

    Thread.Sleep(Ran.Next(1000));

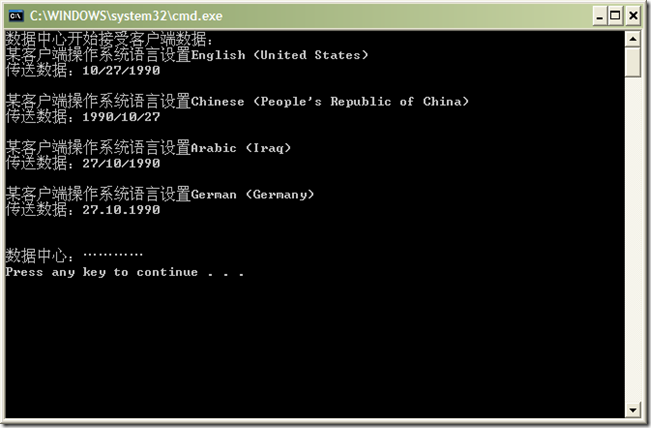
    CultureInfo cul = CultureInfo.GetCultureInfo(CultureSources[id]);

    Thread.CurrentThread.CurrentCulture = cul;

    Console.WriteLine("某客户端操作系统语言设置{0}\n传送数据：{1}\n", cul.DisplayName, newDateTime(1990, 10, 27).ToShortDateString());

}

运行结果：

[](http://images.cnblogs.com/cnblogs_com/mgen/201105/201105051421491088.png)

结果看到了吧，同样的DateTime.ToShortDateString()，在英语-美国，中文-中国，阿拉伯语-伊拉克和德语-德国的不同环境下，1990年10月27日竟然有如此不同的输出结果，这些数据让数据中心服务器情何以堪啊……

原因则已提到过，在进行日期时间输出时，.NET会考虑当前线程的CultureInfo，即 Thread.CurrentThread.CurrentCulture（或者CultureInfo.CurrentCulture），并根据 CultureInfo，进行相应地区文化的数据处理。注意不要和UICulture混淆。

解决方案就是使用这个特殊的InvariantCulture。

把输出代码改成

Console.WriteLine("某客户端操作系统语言设置{0}\n传送数据：{1}\n", cul.DisplayName,

    new DateTime(1990, 10, 27).ToString(

    CultureInfo.InvariantCulture.DateTimeFormat.ShortDatePattern, CultureInfo.InvariantCulture));

这样不管客户端运行在什么语言环境下，输出的时间格式都是统一的，方面数据中心服务器对数据做后续处理。

（当然这个例子仅用来演示InvariantCulture的用法，是否存在其他不妥处这里不做讨论）

[返回目录](http://www.cnblogs.com/mgen/archive/2011/05/05/2037603.html#_hContent)

**InvariantCulture和字符串比较**

下面代码进行四种字符串比较方法，分别是zh-cn, en-us文化，数值比较和InvariantCulture比较(全部是区分大小写)。

static void Main()

{

    string[] strs =

        { "a", "A", "b", "B", "abc", "ab", "aB", "AB", "Ab", "aaa", "00", "0001", "002", "a4", "a9", "a33" };

    Console.WriteLine("en-US");

    Array.Sort<string>(strs, StringComparer.Create(CultureInfo.GetCultureInfo("en-us"), false));

    Console.WriteLine(String.Join(" < ", strs));

    Console.WriteLine("zh-CN");

    Array.Sort<string>(strs, StringComparer.Create(CultureInfo.GetCultureInfo("zh-CN"), false));

    Console.WriteLine(String.Join(" < ", strs));

    Console.WriteLine("Ordinal");

    Array.Sort<string>(strs, StringComparer.Ordinal);

    Console.WriteLine(String.Join(" < ", strs));

    Console.WriteLine("Invariant");

    Array.Sort<string>(strs, StringComparer.InvariantCulture);

    Console.WriteLine(String.Join(" < ", strs));

}

比较结果

en-US

00 < 0001 < 002 < a < A < a33 < a4 < a9 < aaa < ab < aB < Ab < AB < abc < b < B

zh-CN

00 < 0001 < 002 < a < A < a33 < a4 < a9 < aaa < ab < aB < Ab < AB < abc < b < B

Ordinal

00 < 0001 < 002 < A < AB < Ab < B < a < a33 < a4 < a9 < aB < aaa < ab < abc < b

Invariant

00 < 0001 < 002 < a < A < a33 < a4 < a9 < aaa < ab < aB < Ab < AB < abc < b < B

（下面全部是区分大小写）

Ordinal是传统比较方式，即比较每个字符的数值，如果相等的话，继续比较下一组，如果有一个没有的话，长度大的算大。

Invariant和大多数CultureInfo都用一种更人性化的比较方法。首先判断一组字符是否是不一样（这里不区分大小写），直接返回结果 如果不一样，因此b>AB，B>abc。如果一样的话比较下一组，如果都一样则比较长度，所以abc>AB>ab，如果长度都一 样最后再比较大小写，另外一个字符大写永远大于小写，所以AB>Ab>aB>ab

更新：

那么InvariantCulture这样比较字符串有什么作用呢？我觉得这样输出形式可读性更高，其实InvariantCulture比较字符 串就是先进行一次不区分大小写的Ordinal比较（不过这里大写字母>小写字母），如果结果不相同的话在进行一遍区分大小写的Ordinal比较 （同样这里也是大写字母>小写字母）。这样可以将字符串先进行一次大的筛选，然后再比较细节，看下面的示例，Ordinal和 InvariantCulture的比较结果可读性更高！

string[] arr = { "Ab", "aB", "AB", "ab", "Abccccccc", "aBccccc", "Abd" };

Array.Sort<string>(arr, StringComparer.Ordinal);

Console.WriteLine(String.Join("\n", arr));

Console.WriteLine();

Array.Sort<string>(arr, StringComparer.InvariantCulture);

Console.WriteLine(String.Join("\n", arr));

比较结果：

|  |  |
| --- | --- |
| Ordinal | InvariantCulture |
| AB  Ab  Abccccccc  Abd  aB  aBccccc  ab | ab  aB  Ab  AB  aBccccc  Abccccccc  Abd |

结果显而易见，Ordinal机器式的纯数值比较，上面的Ordinal排序结果看起来还是很乱，而InvariantCulture则可读性更高。

[返回目录](http://www.cnblogs.com/mgen/archive/2011/05/05/2037603.html#_hContent)

**总结**

支持多种CultureInfo是整个.NET Framework更加人性化，因为这可以使同一个数据适应不同地区和文化，这样当然满足处于不同地区和文化的用户，但前提是数据给“人”看，如果这些数 据用于计算机之间的传输，即给“机器”看，这样的多文化处理反而不妥，造成同一个数据的不同展现形式，尤其是读写两方的文化地区不同时，数据可能根本无法 被正常读取或者产生潜在bug，因此这里，正是InvariantCulture的用武之地。

来源： <<http://www.cnblogs.com/mgen/archive/2011/05/05/2037603.html>>