

当我们遇到以上问题的时候如何解决？

**因为string 是一个类 所以有他的属性 字段 方法 构造方法等**

**那么他有什么方法？**

**方法一：把字符串改成大小写非静态**

**可以把字符串转化为大写的方法：**

**实例名.ToUpper();**

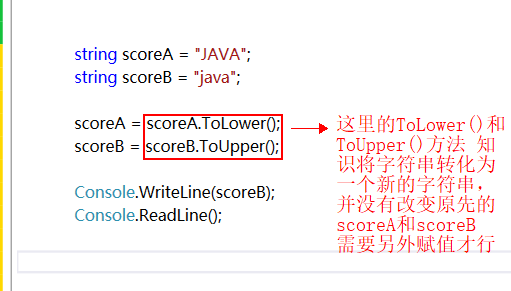
**(这个方法为非静态方法，平常静态方法像访问非静态方法的时候需要通过实例化，然后通过实例名.方法名来调用，但是因为string类型的特殊性，我们在一开始定义并赋值的时候就已经实例化完成，所以这时候的实例名其实就是你一开始定义的变量名，然后只需要通过实例名.方法名调用即可)**

**可以把字符串转化为小写的方法：**

**实例名.ToLower();**

**(这个方法为非静态方法，平常静态方法像访问非静态方法的时候需要通过实例化，然后通过实例名.方法名来调用，但是因为string类型的特殊性，我们在一开始定义并赋值的时候就已经实例化完成，所以这时候的实例名其实就是你一开始定义的变量名，然后只需要通过实例名.方法名调用即可)**

如下图：



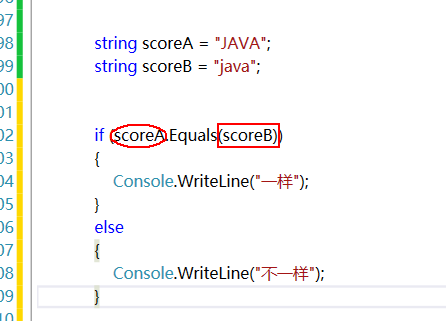
**（我们需要注意的是 scoreA.ToLower()和scoreB.ToUpper() 这里只是把字符串转化为一个新的大写或者小写的字符串，因为字符串的不可变性，所以并没有改变scoreA和scoreB的值，需要另外赋值给ScoreA和scoreB）**

**通过使用这两种方法并且进行判断就很好的实现了 无视大小写 的功能**

**方法二：无视字符串的大小写，对应方法一非静态/静态**

**String还有一种方法能实现比较目的，但是暂且不能无视大小写**

如图：



**具体的语法是：**

**实例名.Equals(需要比较的参数名);（图中的圆形为实例名，正方形为参数名）**

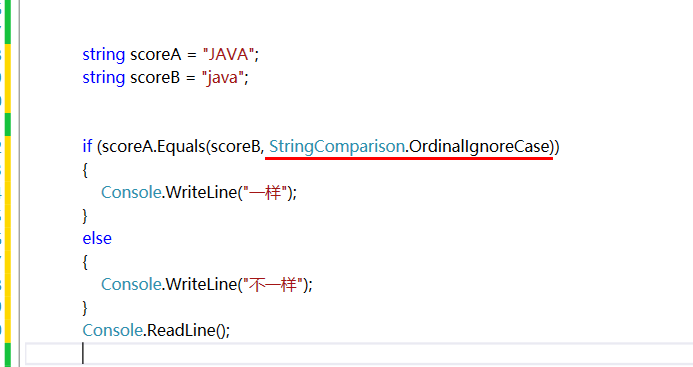
**通过这种方法能返回一个true和false，这样我们就能通过if来比较**

**(这个方法为非静态方法，平常静态方法像访问非静态方法的时候需要通过实例化，然后通过实例名.方法名来调用，但是因为string类型的特殊性，我们在一开始定义并赋值的时候就已经实例化完成，所以这时候的实例名其实就是你一开始定义的变量名，然后只需要通过实例名.方法名调用即可)**

**但是**

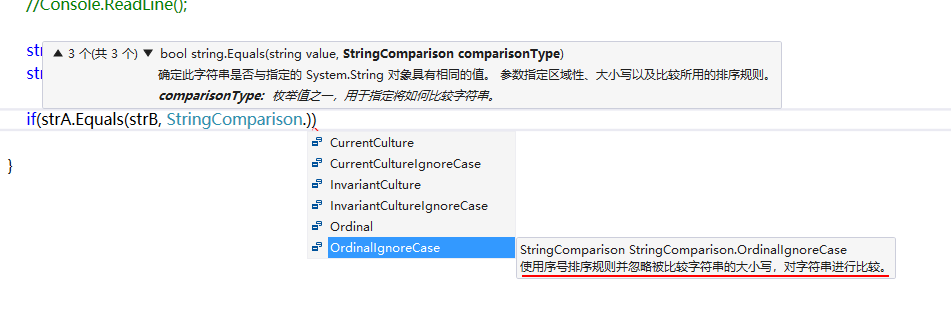
**这种方法还不能实现无视大小写的功能，除非我们这样写**

如图：

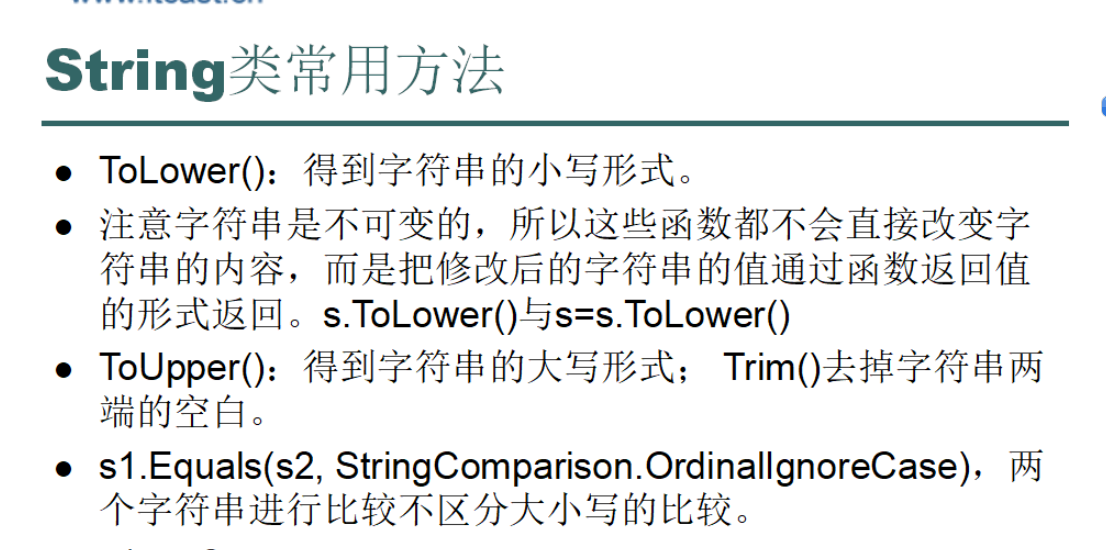


**通过 StringComparison.Ordinallgnorecase就能忽略字符串的大小写，并且进行比较**

**需要注意的是StringComparison.Ordinallgnorecase是个枚举，所以当我们输完scoreB（参数名），再+个空格就会出现自动提示了 代码太长就不用记了**

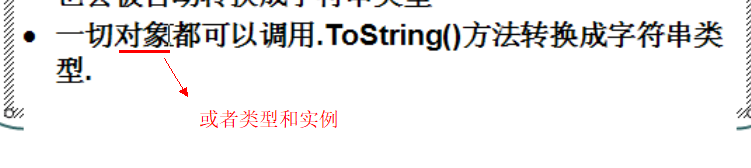
****

**(由此图可见，该枚举的作用)**

****

**方法三：把对象转换成string类型非静态**

**.ToSting()的用法如图：**

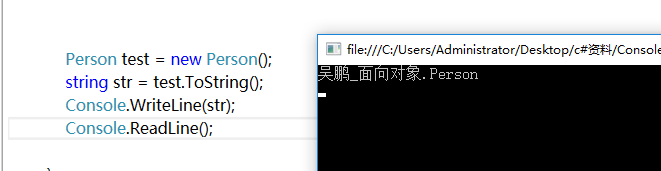
****

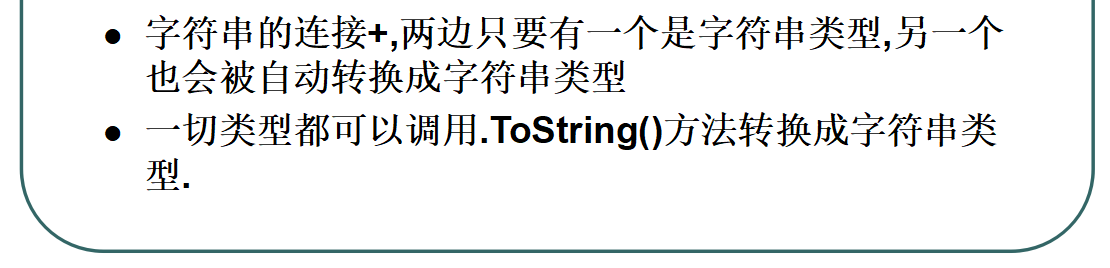
**转换为指定的格式（四舍五入）**

**我们都知道，在数据类型和其他类型使用ToString()的时候都会把转换的变量转化为字符串**

**那么如果用在对象在有什么效果？**

**如图：**

****

****

**方法四：字符串的分割非静态**

**当我们想把一个字符串 如”Hello World”分割成两个string类型数组的元素的时候就需要用到**

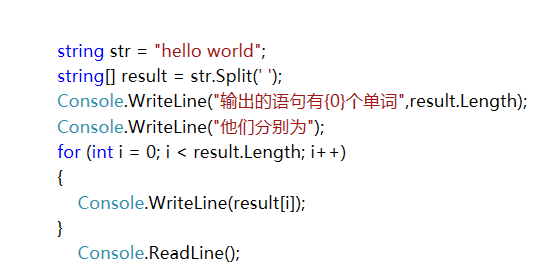
**string[] Split(params char[] separator);**

**（这里的string[]为返回值，返回的是一个string类型的数组）**

**（Split(params char[] separator)而这个params可变char数组则是我们需要输入分割的char元素，比如这里我们需要分割空格 那我们就写 ’ ‘ 因为是可变参数，并且是数组，所以我们能分割多个值，也就是写多个char元素，也可以传一个char类型的数组）**

****

**用法如图：**

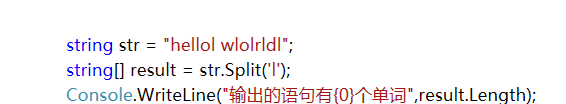
**（我们指定了空格为分割符，这就成功的字符串分割）**

**原理详解：**

**1）这里我们通过传参的方式传给Split方法，并且这个是可变参数，告诉这个方法我们需要分割的是什么，在这里我们传的其实就是char数组的元素，并且传的这个元素 叫做 分割符**

**2）传参后这个方法会返回一个string类型的数组，因为这个方法会自动把分割的字符串组成一个string类型的数组，比如说hello为resul[0]，world为result[1]，那么我们就要定义一个string类型的数组来接收**

**3）分割过程：**

****

**我这里把分割符换成 l 这样更加能看得懂**

**Ps:这里 l 为分割符，当我们需要写多个分割符的时候比如”!”,”,”，那么这两个都是分割符**

**判断第一步：在分割的时候这个方法是先从头开始判断至字符串中出现 分割符 结束，然后就把这个分割结果写成数组中第一个元素，由图可以看出：是从h先开始，然后判断到e的时候 e后面出现了一个l，那么he就是该数组第一个元素，如果在执行第一步的的时候，前面的元素到分割符的位置没有字符，那么这个方法就自动把 空格 即“” 当作是一个数组的元素，即result[0] = “”;**

**判断第二步：在执行完第一步后，方法就会把两个分割符之间的字符作为一个元素赋值给result数组，如果两个分割符之间没有元素，那么就会把 空格 即“” 当作是二个数组的元素，即result[1] = “”;**

**由图可以看出 ：这里我们可以看到，当第一步结束（从头开始判断至分割符的步骤结束），就开始判断两个分割符之间有什么元素了微信截图_20171207011719，但是两个分割符之间没有元素，那么就把“” 即空格当作一个元素赋给数组 然后再判断微信截图_20171207011631，这里两个分割符之间是o 那么o就作为该数组的第三个元素，然后一路判断至d，进入步骤3**

**判断第三步：在执行完判断2后 也就是该字符串里已经没有两个分割符相连的可以判断了，那么就开始判断尾巴，如果尾巴还存在字符的话，那么该字符就作为该数组最后一个元素，如果不存在字符的话 那么就是“”即空格当作该数组中的最后一个元素。由图可以看出 ：该字符串最后面的分割符后面已经没有字符了，那么“”即空格就作为该数组的最后一个元素！**

**但是通过这种方法有种缺陷，那就是会把“”即空格作为元素赋给数组，那么如何解决这种缺陷呢？**

**↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓**

**方法五:解决方法四的数组存在空格的问题非静态**

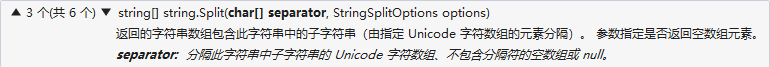
**String[] Split(char[] separator,StringSplitOptions)这个方法就能解决以上的问题！**

**这里的string[]是返回值，即这个方法返回的是一个string类型的数组，与方法四相同**

**不同点是：**

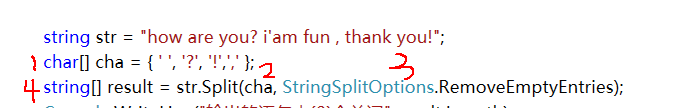
**方法四的方法传参的时候是可变参数，即params，即可以直接把char数组的元素直接传过去，系统会自动组成一个char类型的数组，并把这个数组中的元素当成是分割符！**

**在这方法中没有可变参数，所以在使用的时候我们需要自己另外定义一个char类型的数组，并且把这个数组中的元素当成是分割符**

****

**(string数组也能使用)**

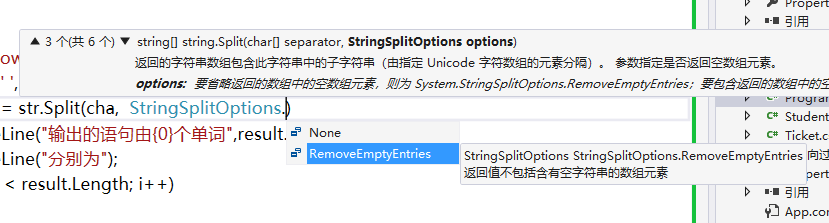
**步骤图：**

****

**步骤一：我们是在使用这个方法前就先定义了一个char类型的数组，并且把数组中的元素的值当成是分割符**

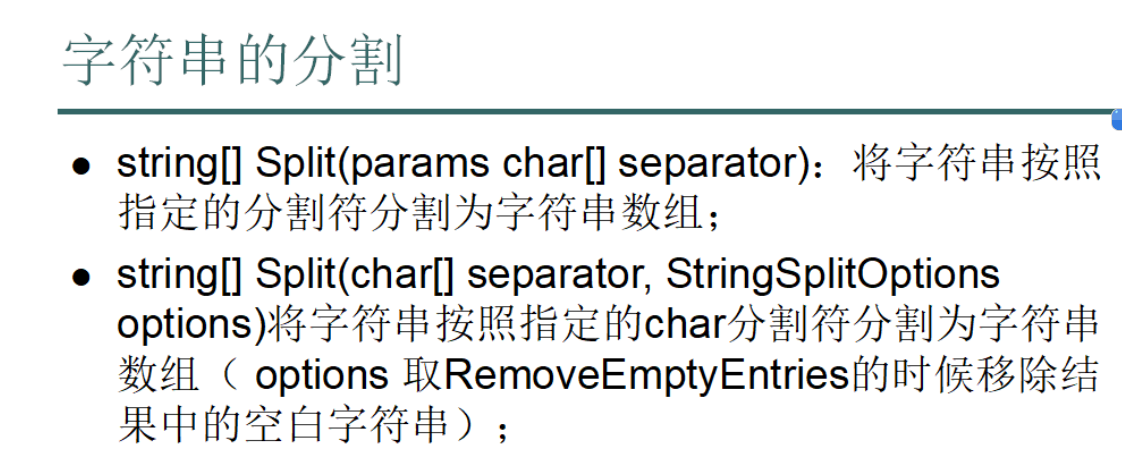
**步骤二：把char类型的数组传过去**

**步骤三：当这里是一个枚举类型，当我们输入空格后会有自动提示我们输入，关于这个枚举类型有两种情况提供给我们使用：**

****

**（第一个None为包含空字符，也就是方法四的缺陷，第二个就是不包含空字符串的元素，所以选定第二个）**

**步骤四：通过了这种分割符的方法后，把分割后的sting类型数组的元素传入string[] result中，这时候result的值便是分割后的结果，并且不存在“”即空格的元素！**

****

**方法六:替换字符串的方法非静态**

**当我们想替换字符串的字符或子字符串的时候应该怎么办？**

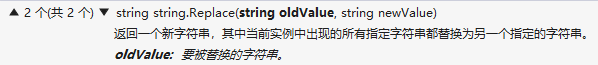
**使用 string Replace(string oldValue, string newValue);**

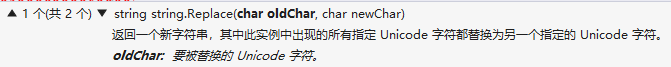
**需要注意的是：**

1. **这里的string 是一个返回值，即这个方法不能直接改变字符串，而是通过Replace(string oldValue, string newValue);这个方法得到一个全新的字符串，然后返回一个string类型的字符串，然后通过赋值或输出来改变。**
2. **将字符串中的出现oldValue的地方替换为newValue**
3. **在这个方法中除了能用能替换string类型，也可以替换char类型，但是替换char的话只能替换一个字符，而使用string的话则可以改变子字符串**

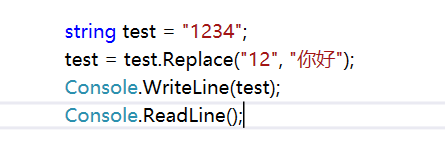
**疑问：为什么除了能改变string 也能改变char？**

**如 string a = “1234”，这个其实就是一个char数组的集合，其实也就是char[] a = {1,2,3,4}，而这些元素其实都是包含在a这个字符串里面的字符，所以当我们只想改变一个字符的话用char，也就是改变char[] a数组中某个元素的值，但注意都是一个的，想改变多个字符的话用string**

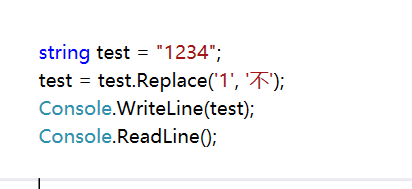




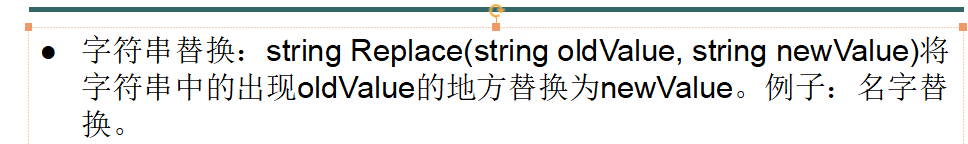
**方法如图：**

****

**(这个为要改变子字符串的代码图)**

****

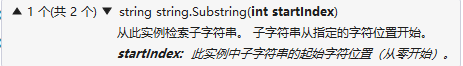
**(这个为要改变字符的代码图)**

****

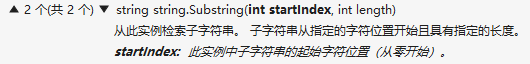
**方法七:取字符串某个位置中的值 非静态**

**当我们想取一个字符串某个位置中得值的时候就需要用到这两种方法：**

1. **string Substring(int startIndex)，取从位置startIndex开始的字符一直到最后的字符的字符串；非静态**



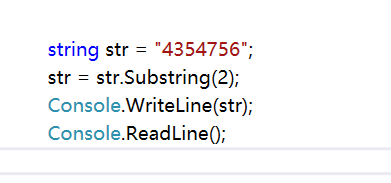
1. **string Substring(int startIndex, int length)，取从位置startIndex开始的字符，截取长度为length的字符串，如果子字符串的长度不足length则报错。**



**注意：**

1. **这里的string 是一个返回值，即这个方法不能直接改变字符串，返回值一个string类型的返回值，然后我们再对这个返回值赋值或者输出**
2. **string Substring(startIndex)括号里面startIndex是制定的值，从0开始指定，然后从制定的值开始一直到字符串的最后**

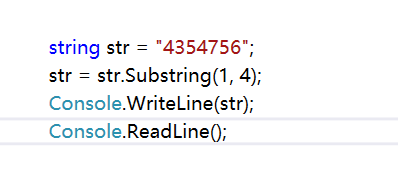
**如图：**



**（这里我们是从2开始指定，也就是从字符串 0 1 开始数，数到2，也就是从4 3 开始数，然后数到了5，那么输出的结果就是54756，由5开始一直到字符串的结束）**

1. **string Substring(int startIndex, int length)括号里面startIndex是制定的值，从0开始指定，然后length为从指定的值开始且具有长度，长度也就是length，从 1 开始算，比如说长度是4，那么就要算 1， 2，3，4，而不是0，1，2，3，4！**

**如图：**



**（这里我们是从1开始指定，那么从0开始数，数到1，也就是从4开始数，数到 3，那么就是从3开始，而我们代码中写的长度是4，那么就是往后取4为数并包含3，所以输出的结果为 3547）**

**方法八：判断字符串中是否由某个字符非静态**

**当我们想判断某个字符串里是否存在某种字符的时候，就需要用到了：**

**bool Contains(string value)**

**注意：**

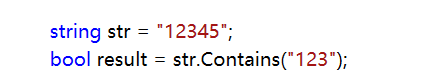
1. **这个方法跟方法方法六一样都是存在char 和string的重载，当我们想判断的元素为一个字符的时候就用char，存在子字符串的时候就用string，可以写入char或者string数组进行判断**

**(string)**

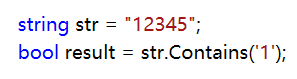
**(char)**

**2）这个方法返回的是一个bool类型，当存在我们所写的子字符串或字符的时候就会返回true，不存在的时候就会返回false，所以一般和if配合使用**

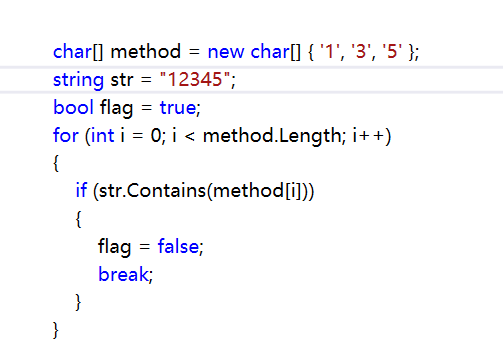
**用法如图：**



**(判断是否存在子字符串）**



**(判断是否存在一个字符）**



**(复杂点的用法，判断是否存在多个子字符串，通过数组和for循环来得出）**

**方法九：取得子字符串或者字符第一次出现的位置 非静态**

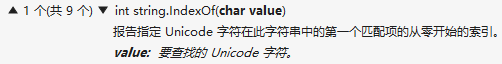
**当我们像判断一个字符串中的子字符串或者字符第一次出现的位置在哪的时候就需要用到这种**

**方法:**

1. **int IndexOf(string value)**



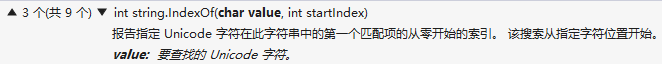
1. **int IndexOf(char value)**

****

1. **Int IndexOf(string value,int startIndex)**

**微信截图_20171207202207**

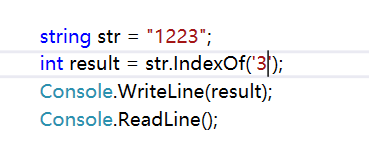
1. **Int IndexOf(char value,int startIndex)**

****

**注意：**

1. **方法1和2是一样的，都是找到value 即子字符串或者字符第一次出现的位置，当我们想找子字符串的时候就用string，想找字符的时候就用char**

**代码图如下：**

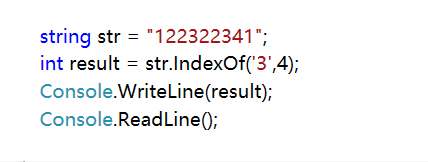


**这里我们指定的是3的字符，并且找到的结果是从0开始数，所以找到的结果为 3**

**当我们在这个字符串中没有找到指定的value的时候，会默认输出 -1**

1. **方法3和方法4是一样的，都是找到value 即子字符串或者字符第一次出现的位置，而后面的 int startIndex的意思是指定从哪个位置开始找，指定的位置要从0开始**

**代码图如下：**

****

**在这里我们指定了从4开始找，因为我们如果不指定从4开始找，那么系统就会只找到从头开始的第一个3，也就是输出结果为 3**

**当我们指定了从4开始找后，输出结果就是6了**

**不管是 指定位置的时候 即int startIndex 还是输出结果，都是从0开始数**

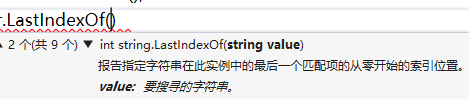
**方法十:取字符串或字符最后一次出现的位置 非静态**

**当我们想判断一个字符串中的一个子字符串或者字符最后一次出现的位置就需要用到**

1. **取一个字符最后出现的位置**

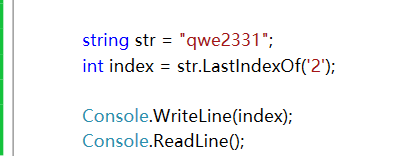
****

1. **取一个子字符最后出现的位置**

****

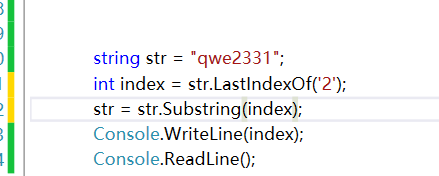
**用法：**

1. **普通的取子字符或者字符最后出现的位置**

****

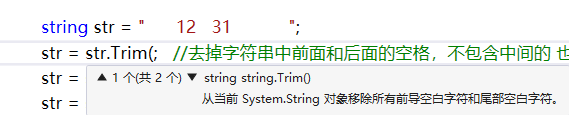
**（输出的结果要从0开始计算，所以输出结果是3）**

1. **配合Substring来使用**

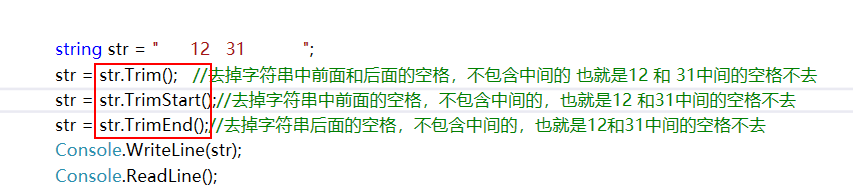
****

**方法十一：去掉一个字符串中的空格非静态**

**当我们需要去掉一个字符串的空格的时候就要用到 Trim（）;**

****

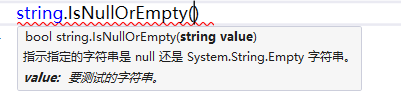
**用法:**

****

**(Trim()的三种用法都列出来了，需要注意的是，当时用Trim()去空格的时候只是去掉前面和后面的，字符串中间的不会去掉，所以还是用split的方法好一点)**

**方法十二：判断一个字符串是否为null或Empty静态**

**当想判断一个字符串是否为空或者没有的时候就需要用到**

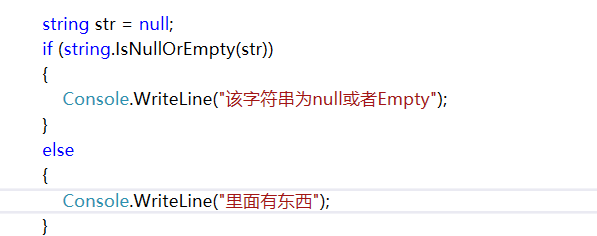
****

**注意：**

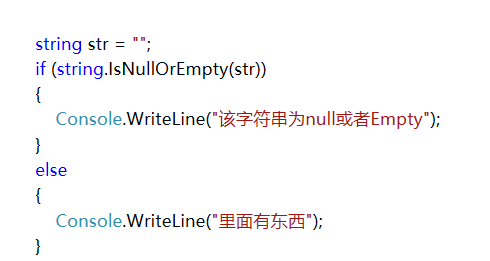
1. **该方法的返回值为bool类型，所以一般配合if来使用**
2. **该方法为静态方法，所以要通过类名.方法名来调用**

**用法：**

1. **判断一个字符串是否为空**

****

1. **判断一个字符串是否为没有**

****

**方法十三：把数组按照指定的字符串进行连接静态**

**当我们想把一个数组按照指定的字符串进行连接的时候就需要用到**

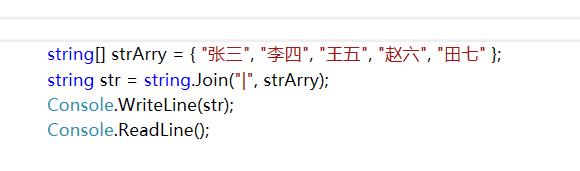
**string string.Join(string separator,params object[] values)**

****

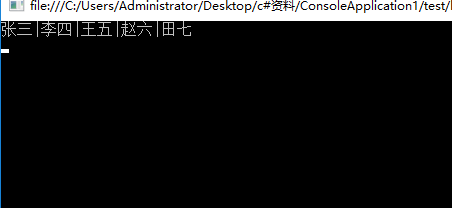
**注意：**

1. **这里的string 是个返回值类型，这个方法会按照我们需要做的分割符分割，分割完成后会返回一个字符串，我们需要顶一个string类型变量进行接收**
2. **string separator 为分割的符号**
3. **params object[] values为可变参数，可以直接在上面写参数，也可以直接写个数组**
4. **这个方法不会在数组的最后一个元素还+上分割符**
5. **这个方法是静态方法，所以调用的时候直接类名.方法名来调用**

**用法：**

****

**(代码图)**

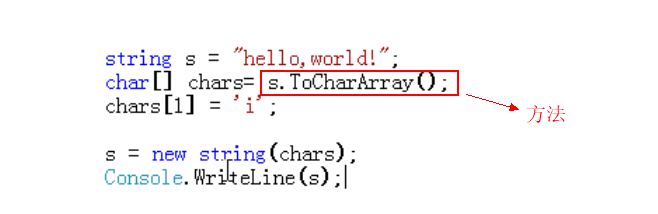
****

**(结果图，数组最后一个元素没有+分割符)**

**方法十四：把字符串转换成char类型的数组非静态**

**法当我们需要把一个字符串转换成一个char类型的数组的时候就需要用到**

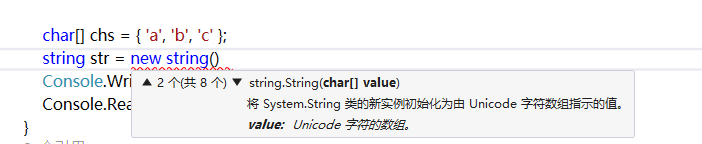
**char[] value ToCharArry();**

****

**方法十五：把char类型的数组转换成字符串创建新的对象调用构造方法**

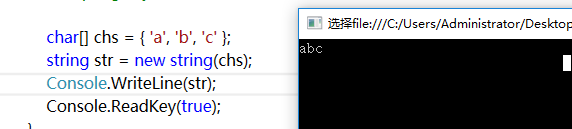
**当我们需要把一个 char[] value数组转换成字符串的时候就需要用到**

**string new string(char[] value)**

****

**具体用法**

**如图：**

****

**(可以看到我们把转换成功后的数组输出，是一个字符串了)**

**方法十六：把任意类型的元素连接成字符串静态**

**注意：该方法的实参如果存在值类型的元素，那么就会发生装箱的形为**

****

**平时我们使用+号来对字符串与任意类型的元素进行相连的时候，如下图：**

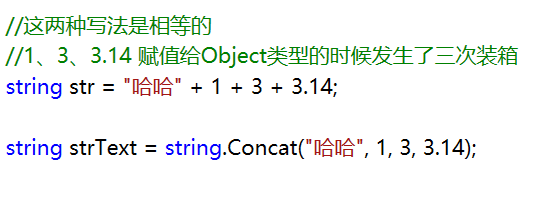
**微信截图_20180216234948**

**其实调用了string.Concat()这个方法，如下图：**

**微信截图_20180216235118**

**但是这个当我们相连的元素存在值类型的时候，会发生装箱的现象，那么为什么？**

**因为：string.Concat()函数重载的参数的问题，在他的函数重载的参数中，有string类型的参数，也有object类型的参数，当我们在使用+号进行字符串相连的时候，其实系统在隐式的调用了string.Concat()这个方法，当我们相连元素中存在值类型的时候，就会默认匹配到有Object类型参数的函数的重载，这时候值类型赋值给了Object引用类型，就发生了一次装箱，如下图：**

****

**方法十七：还有很多 需要自己去msdn上看，常用的已经列出来了**

