

潇十一郎

目光聚集之处，金钱必将追随☆

[首页](#) [新随笔](#) [联系](#) [管理](#)

随笔 - 73 文章 - 0 评论 - 158

C# 正则表达式大全

阅读目录

- [前言](#)
- [文章导读](#)
- [基础梳理](#)
- [Regex类常用的方法](#)
- [Regex类的实例](#)
- [命名空间说明](#)
- [正则常用表达式](#)
- [使用demo](#)

前言

在网上看到一个不错的简易版正则匹配和替换的工具，现在补充进来，感觉还不错，效果如下（输入验证中文汉字的正则表达式）

公告

C#从入门到精通

交流群:



加入QQ群

管理系统墙裂推

荐: **CRM全家桶**

潇十一郎个人站
简介

fuyue.xyz是一个充满意义的网

它
着
故
展
人
说
深
长
有

不

昵称: [潇十一郎](#)
园龄: 3年1个月
粉丝: 117

js正则查找替换工具

请输入正则表达式:
[\\u4e00-\\u9fa5]

小写 ☒ 查找 ☐ 替换

请输入替换的正则表达式:

填写你想查找或替换的文本:
123你123acbs好! |

开始

关注: 37

[+加关注](#)

我的标签

[MyEclipse\(1\)](#)[MyEclipse2014](#)[破解\(1\)](#)[Sql Server\(1\)](#)[存储过程\(1\)](#)[激活\(1\)](#)[破解\(1\)](#)[网络编程\(1\)](#)[抓包工具\(1\)](#)[在线下载](#) 密码: 5tpt

注: 好像也是一位园友写的, 但是找不到地址了, 有看到的可以留言告知下, thx

文章导读

正则表达式的本质是**使用一系列特殊字符模式, 来表示某一类字符串**。正则表达式无疑是处理文本最有力的工具, 而.NET提供的Regex类实现了验证正则表达式的方法。Regex 类表示不可变(只读)的正则表达式。它还包含各种静态方法, 允许在不显式创建其他类的实例的情况下使用其他正则表达式类。

基础梳理

随笔分类

[C#\(42\)](#)[MVC\(19\)](#)[其他\(16\)](#)[前端\(5\)](#)[区块链\(3\)](#)[设计模式\(4\)](#)[数拼](#)[网络](#)[微信](#)[入门](#)

随笔

[2018](#)[2018年5月 \(3\)](#)[2018年4月 \(1\)](#)

字 符	描 述
\	转义字符，将一个具有特殊功能的字符转义为一个普通
^	匹配输入字符串的开始位置
\$	匹配输入字符串的结束位置
*	匹配前面的零次或多次的子表达式
+	匹配前面的一次或多次的子表达式
?	匹配前面的零次或一次的子表达式
{n}	n是一个非负整数，匹配前面的n次子表达式
{n, }	n是一个非负整数，至少匹配前面的n次子表达式
{n, m}	m和n均为非负整数，其中n<=m，最少匹配n次且最多匹
?	当该字符紧跟在其他限制符 (*, +, ?, {n}, {n, }, {n
.	匹配除 “\n” 之外的任何单个字符
(pattern)	匹配pattern并获取这一匹配
(?:pattern)	匹配pattern但不获取匹配结果
(?=pattern)	正向预查，在任何匹配pattern的字符串开始处匹配查
(?!pattern)	负向预查，在任何不匹配pattern的字符串开始处匹配查
x y	匹配x或y。例如， ‘z food’ 能匹配 “z” 或 “food”。 匹配 “zood” 或 “food”
[xyz]	字符集合。匹配所包含的任意一个字符。例如， ‘[abc

2018年1月 (5)

2017年8月 (1)

2017年7月 (8)

2017年6月 (2)

2017年5月 (1)

2017年3月 (1)

2017年2月 (2)

2017年1月 (6)

2016年12月 (4)

2016年11月 (5)

2016年10月 (3)

2016年9月 (2)

2016年8月 (4)

2016年7月 (2)

2016年5月 (1)

2016年4月 (4)

2016年3月 (2)

2016年2月 (1)

2016年1月 (5)

2015年12月 (5)

2015年6月 (1)

2015年5月 (1)

阅读排行榜

1. 行

2. 。

3. 行

4. 行

5. 。

[^xyz]	负值字符集合。匹配未包含的任意字符。例如，‘[^ab]’匹配“plain”中的‘p’
[a-z]	匹配指定范围内的任意字符。例如，‘[a-z]’可以匹配任意小写字母字符
[^a-z]	匹配不在指定范围内的任意字符。例如，‘[^a-z]’可在‘a’ ~ ‘z’内的任意字符
\b	匹配一个单词边界，指单词和空格间的位置
\B	匹配非单词边界
\d	匹配一个数字字符，等价于[0-9]
\D	匹配一个非数字字符，等价于[!0-9]
\f	匹配一个换页符
\n	匹配一个换行符
\r	匹配一个回车符
\s	匹配任何空白字符，包括空格、制表符、换页符等

\S	匹配任何非空白字符
\t	匹配一个制表符
\v	匹配一个垂直制表符。等价于\x0b和\cK
\w	匹配包括下划线的任何单词字符。等价于‘[A-Za-z0-9_]’
\W	匹配任何非单词字符。等价于‘[^A-Za-z0-9_]’

说明：

由于在正则表达式中“\”、“?”、“*”、“^”、“\$”、“+”、“(”、“)”、“|”、“{”、“[”等字符已经具有一定特殊意义，如果需要它们的原始意义，则应该对它进行转义，例如希望在字符串中至少有一个“\”，那么正则表达式应该这么写：\\+。

Regex类常用的方法

①静态Match方法

69 使用静态Match方法，可以得到源中第一个匹配模式的连续子串。

静态的Match方法有2个重载，分别是

推荐排行榜

1. [C# 正则表达式大全\(69\)](#)
2. [C# TCP多线程服务器示例\(20\)](#)
3. [Word/Excel 在线预览\(17\)](#)
4. [C# MVC权限验证\(16\)](#)
5. [微信公众号开发之VS远程调试\(15\)](#)

```
Regex.Match(string input, string pattern);  
Regex.Match(string input, string pattern, RegexOptions  
options);
```

第一种重载的参数表示：输入、模式

第二种重载的参数表示：输入、模式、RegexOptions枚举的“按位或”组合。

RegexOptions枚举的有效值是：

Compiled表示编译此模式

CultureInvariant表示不考虑文化背景

ECMAScript表示符合ECMAScript，这个值只能和IgnoreCase、Multiline、Compiled连用

ExplicitCapture表示只保存显式命名的组

IgnoreCase表示不区分输入的大小写

IgnorePatternWhitespace表示去掉模式中的非转义空白，并启用由#标记的注释

Multiline表示多行模式，改变元字符^和\$的含义，它们可以匹配行的开头和结尾

None表示无设置，此枚举项没有意义

RightToLeft表示从右向左扫描、匹配，这时，静态的Match方法返回从右向左的第一个匹配

Singleline表示单行模式，改变元字符.的意义，它可以匹配换行符

注意：Multiline在没有ECMAScript的情况下，可以和Singleline连用。Singleline和Multiline不互斥，但是和ECMAScript互斥。

②静态的Matches方法

这个方法的重载形式同静态的Match方法，返回一个MatchCollection，表示输入中，匹配模式的匹配的集合。

③静态的IsMatch方法

此方法返回一个bool，重载形式同静态的Matches，若输入中匹配模式，返回true，否则返回false。

69

可以理解为：IsMatch方法，返回Matches方法返回的集合是否为空。

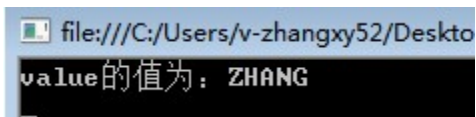
Regex类的实例

(1)字符串替换

```
//例如我想把如下格式记录中的NAME值修改为WANG
string line = "ADDR=1234;NAME=ZHANG;PHONE=6789";
Regex reg = new Regex("NAME=(.+)");
string modified = reg.Replace(line, "NAME=WANG;");
//修改后的字符串为 ADDR=1234;NAME=WANG;PHONE=6789
```

(2)字符串匹配

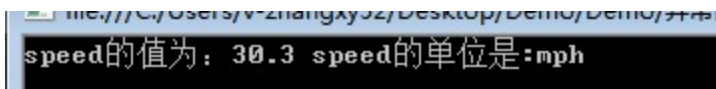
```
string line = "ADDR=1234;NAME=ZHANG;PHONE=6789";
Regex reg = new Regex("NAME=(.+)");
//例如我想提取line中的NAME值
Match match = reg.Match(line);
string value = match.Groups[1].Value;
Console.WriteLine("value的值为: {0}", value);
```



file:///C:/Users/v-zhangxy52/Desktop
value的值为: ZHANG

(3)Match实例

```
//文本中含有"speed=30.3mph",需要提取该速度值,但是速度的单位可能是公制也可能是英制, mph, km/h, m/s都有可能;另外前后可能有空格。
string line = "lane=1;speed=30.3mph;
acceleration=2.5mph/s";
Regex reg = new Regex(@"speed\s*=\s*([\d\.]+)\s*
(mph|km/h|m/s)");
Match match = reg.Match(line);
//那么在返回的结果中match.Groups[1].Value将含有数值,而
match.Groups[2].Value将含有单位。
var 值 = match.Groups[1].Value; //此处方便演示,在实际开发中请勿使用中文命名变量
var 单位 = match.Groups[2].Value;
Console.WriteLine("speed的值为: {0} speed的单位是: {1}", 值, 单位);
```



file:///C:/Users/v-zhangxy52/Desktop/Demo/Demo/Program.cs
speed的值为: 30.3 speed的单位是: mph

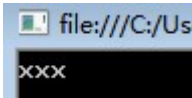
69 (4)解码gps的GPRMC字符串

```
//就可以获得经度、纬度值,而以前需要几十行代码。
Regex reg = new Regex(@"^$GPRMC, [\d\.]*, [A|V], (-?[0-9]*)
```

```
\.?[0-9]+), ([NS]*), (-?[0-9]*\.?[0-9]+), ([EW]*), .*" );
```

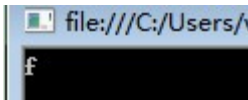
(5)提取[]的值

```
string pattern1 = @"(?is) (?<=\[) (.*) (?=\]) ";  
string result1 = new  
Regex(pattern1).Match("sadff[xxx]sdfdsf").Value;
```



(6)提取()的值

```
string pattern2 = @"(?is) (?<=\() (.*) (?=\)) ";  
string result2 = new  
Regex(pattern2).Match("sad(f)dsf").Value;
```



(7)提取{}的值

```
string pattern3 = @"(?is) (?<=\{) (.*) (?=\}) ";  
string result3 = new  
Regex(pattern3).Match("sadff[{xxx}sdfd}sf").Value;
```



命名空间说明

System.Text.RegularExpressions命名空间的说明

该名称空间包括8个类，1个枚举，1个委托。他们分别是：

```
Capture: 包含一次匹配的结果;  
CaptureCollection: Capture的序列;  
Group: 一次组记录的结果, 由Capture继承而来;  
GroupCollection: 表示捕获组的集合  
Match: 一次表达式的匹配结果, 由Group继承而来;  
MatchCollection: Match的一个序列;  
MatchEvaluator: 执行替换操作时使用的委托;  
RegexCompilationInfo: 提供编译器用于将正则表达式编译为独立程序
```

集的信息

RegexOptions 提供用于设置正则表达式的枚举值

Regex类中还包含一些静态的方法:

Escape: 对字符串中的regex中的转义符进行转义;

IsMatch: 如果表达式在字符串中匹配, 该方法返回一个布尔值;

Match: 返回Match的实例;

Matches: 返回一系列的Match的方法;

Replace: 用替换字符串替换匹配的表达式;

Split: 返回一系列由表达式决定的字符串;

Unescape: 不对字符串中的转义字符转义。

正则常用表达式

(-)校验数字的表达式

```
//数字
Regex reg = new Regex(@"^[0-9]*$");
//n位的数字
Regex reg = new Regex(@"^d{n}$");
//至少n位的数字
Regex reg = new Regex(@"^d{n,}$");
//m-n位的数字
Regex reg = new Regex(@"^d{m,n}$");
//零和非零开头的数字
Regex reg = new Regex(@"^(0|[1-9][0-9]*)$");
//非零开头的最多带两位小数的数字
Regex reg = new Regex(@"^([1-9][0-9]*)+
(. [0-9]{1,2})?$");
//带1-2位小数的正数或负数
Regex reg = new Regex(@"^(\-)?\d+(\.
\d{1,2})?$");
//正数、负数、和小数
Regex reg = new Regex(@"^(\-|\+)?\d+
(\. \d+)?$");
//有两位小数的正实数
Regex reg = new Regex(@"^[0-9]+(.[0-9]
{2})?$");
//有1~3位小数的正实数
Regex reg = new Regex(@"^[0-9]+(.[0-9]
{1,3})?$");
//非零的正整数
Regex reg = new Regex(@"^[1-9]\d*$ 或
^([1-9][0-9]*){1,3}$ 或 ^\+?[1-9][0-9]*$");
//非零的负整数
Regex reg = new Regex(@"^- [1-9][0-9]*$ 或
^- [1-9]\d*$");
```



```
//非负整数
Regex reg = new Regex(@"^0+$ 或
^[1-9]\d*|0$");
//非正整数
Regex reg = new Regex(@"^-([1-9]\d*|0$ 或
^((- \d+) | (0+))$");
//非负浮点数
Regex reg = new Regex(@"^0+(\.\d+)?$ 或
^[1-9]\d*\.\d*|0\.\d*[1-9]\d*|0?\.\d+|0$");
//非正浮点数
Regex reg = new Regex(@"^((- \d+(\.\d+)?) |
(0+(\.\d+)?))$ 或 ^(-([1-9]\d*\.\d*|0\.\d*[1-9]\d*)|0?
\.\d+|0$");
//正浮点数
Regex reg = new Regex(@"^[1-9]\d*\.\d*|0
\.\d*[1-9]\d*$ 或 ^((([0-9]+\.[0-9]*[1-9][0-9]*) | ([0-9]*
[1-9][0-9]*\.[0-9]+) | ([0-9]*[1-9][0-9]*) )$");
//负浮点数
Regex reg = new Regex(@"^-([1-9]\d*\.\d*|0
\.\d*[1-9]\d*)$ 或 ^(-((([0-9]+\.[0-9]*[1-9][0-9]*) |
([0-9]*[1-9][0-9]*\.[0-9]+) | ([0-9]*[1-9][0-9]*) )$");
//浮点数
Regex reg = new Regex(@"^(-?\d+)(\.\d+)?$ 或
^-?([1-9]\d*\.\d*|0\.\d*[1-9]\d*|0?\.\d+|0)$");
```

(二)校验字符的表达式

```
//汉字
Regex reg = new Regex(@"^[\u4e00-\u9fa5]
{0,}$");
//英文和数字
Regex reg = new Regex(@"^[A-Za-z0-9]+$ 或
^[A-Za-z0-9]{4,40}$");
//长度为3-20的所有字符
Regex reg = new Regex(@"^.{3,20}$");
//由26个英文字母组成的字符串
Regex reg = new Regex(@"^[A-Za-z]+$");
//由26个大写英文字母组成的字符串
Regex reg = new Regex(@"^[A-Z]+$");
//由26个小写英文字母组成的字符串
Regex reg = new Regex(@"^[a-z]+$");
//由数字和26个英文字母组成的字符串
Regex reg = new Regex(@"^[A-Za-z0-9]+$");
//由数字、26个英文字母或者下划线组成的字符串
Regex reg = new Regex(@"^[a-zA-Z0-9_]+$ 或
^w{3,20}$");
//中文、英文、数字包括下划线
```

```

Regex reg = new Regex(@"^\u4E00-\u9FA5A-Za-
z0-9_]+$");
//中文、英文、数字但不包括下划线等符号
Regex reg = new Regex(@"^\u4E00-\u9FA5A-Za-
z0-9]+$ 或 ^[\u4E00-\u9FA5A-Za-z0-9]{2,20}$");
//可以输入含有^%&' , ; = ? $ \ " 等字符
Regex reg = new Regex(@"^[^%&' , ; = ? $ \x22]+$");
//禁止输入含有~的字符
Regex reg = new Regex(@"^[^\~\x22]+$");

```

(三)特殊需求表达式

```

//Email地址
Regex reg = new Regex(@"^\w+([-+.] \w+)*@
\w+([-.] \w+)*\.\w+([-.] \w+)*$");
//域名
Regex reg = new Regex(@"[a-zA-Z0-9] [-a-zA-
Z0-9]{0,62} (/[a-zA-Z0-9] [-a-zA-Z0-9]{0,62})+/.?");
//InternetURL
Regex reg = new Regex(@"[a-zA-z]+://[^\s]*
或 ^http://([\w-]+\.)+[\w-]+(/[\w-./?%&=]*)?$");
//手机号码
Regex reg = new
Regex(@"^(13[0-9]|14[5|7]|15[0|1|2|3|5|6|7|8|9]|18[0|1|2
|3|5|6|7|8|9])\d{8}$");
//电话号码("XXX-XXXXXXX"、"XXXX-
XXXXXXX"、"XXX-XXXXXXX"、"XXXXXXX"
和"XXXXXXX")
Regex reg = new Regex(@"^($\d{3,4}-|
\d{3,4}-)?\d{7,8}$");
//国内电话号码(0511-4405222、021-87888822)
Regex reg = new Regex(@"\d{3}-\d{8}|\d{4}-
\d{7}");
//身份证号(15位、18位数字)
Regex reg = new Regex(@"^\d{15}|\d{18}$");
//短身份证号码(数字、字母x结尾)
Regex reg = new Regex(@"^([0-9]){7,18}
(x|X)?$ 或 ^\d{8,18}|[0-9x]{8,18}|[0-9X]{8,18}?");
//帐号是否合法(字母开头,允许5-16字节,允许字母数字
下划线)
Regex reg = new Regex(@"^[a-zA-Z] [a-zA-
Z0-9_] {4,15}$");
//密码(以字母开头,长度在6~18之间,只能包含字母、数
字和下划线)
Regex reg = new Regex(@"^[a-zA-
Z]\w{5,17}$");
//强密码(必须包含大小写字母和数字的组合,不能使用特

```

殊字符, 长度在8-10之间)

```
Regex reg = new Regex(@"^(?=[*\d])(?=[*a-z])(?=[*A-Z]).{8,10}$");
```

//日期格式

```
Regex reg = new Regex(@"^\d{4}-\d{1,2}-\d{1,2}$");
```

//一年的12个月(01~09和1~12)

```
Regex reg = new Regex(@"^(0?[1-9]|1[0-2])$");
```

//一个月的31天(01~09和1~31)

```
Regex reg = new Regex(@"^((0?[1-9])|((1|2)[0-9])|30|31)$");
```

//钱的输入格式:

//有四种钱的表示形式我们可以接受: "10000.00" 和 "10,000.00", 和没有 "分" 的 "10000" 和 "10,000"

```
Regex reg = new Regex(@"^[1-9][0-9]*$");
```

//这表示任意一个不以0开头的数字, 但是, 这也意味着一个字符"0"不通过, 所以我们采用下面的形式

```
Regex reg = new Regex(@"^(0|[1-9][0-9]*)$");
```

//一个0或者一个不以0开头的数字. 我们还可以允许开头有一个负号

```
Regex reg = new Regex(@"^(0|-?[1-9][0-9]*)$");
```

//这表示一个0或者一个可能为负的头不为0的数字. 让用户以0开头好了. 把负号的也去掉, 因为钱总不能是负的吧. 下面我们要加的是说明可能的小数部分

```
Regex reg = new Regex(@"^[0-9]+(.[0-9]+)?$");
```

//必须说明的是, 小数点后面至少应该有1位数, 所以"10."是不通过的, 但是 "10" 和 "10.2" 是通过的

```
Regex reg = new Regex(@"^[0-9]+(.[0-9]{2})?$");
```

//这样我们规定小数点后面必须有两为, 如果你认为太苛刻了, 可以这样

```
Regex reg = new Regex(@"^[0-9]+(.[0-9]{1,2})?$");
```

//这样就允许用户只写一位小数. 下面我们该考虑数字中的逗号了, 我们可以这样

```
Regex reg = new Regex(@"^[0-9]{1,3}([,][0-9]{3})*(.?[0-9]{1,2})?$");
```

//1到3个数字, 后面跟着任意个 逗号+3个数字, 逗号成为可选, 而不是必须

```
Regex reg = new Regex(@"^([0-9]+|[0-9]{1,3}([,][0-9]{3})*)(.?[0-9]{1,2})?$");
```

//备注: 这就是最终结果了, 别忘了"+"可以用"*"替代。

如果你觉得空字符串也可以接受的话 (奇怪, 为什么?) 最后, 别忘了在用函数时去掉去掉那个反斜杠, 一般的错误都在这里

//xml文件

```
Regex reg = new Regex(@"^([a-zA-Z]+-?)+[a-
```

```

zA-Z0-9]+\.\[x|X][m|M][l|L]$");
//中文字符的正则表达式
Regex reg = new Regex(@"[\u4e00-\u9fa5]");
//双字节字符
Regex reg = new Regex(@"^[x00-\xff] (包括汉字在内,可以用来计算字符串的长度(一个双字节字符长度计2,ASCII字符计1))");
//空白行的正则表达式,可用来删除空白行
Regex reg = new Regex(@"\n\s*\r");
//HTML标记的正则表达式
Regex reg = new Regex(@"<(\S?)[^>]*>.*?</\1>|<.*? />");// (网上流传的版本太糟糕,上面这个也仅仅能部分,对于复杂的嵌套标记依旧无能为力)
//首尾空白字符的正则表达式
Regex reg = new Regex(@"^\s*|\s*$或(^s*)|(\s*$)");// (可以用来删除行首行尾的空白字符(包括空格、制表符、换页符等等),非常有用的表达式)
//腾讯QQ号
Regex reg = new Regex(@"[1-9][0-9]{4,}");
// (腾讯QQ号从10000开始)
//中国邮政编码
Regex reg = new Regex(@"[1-9]\d{5}(?! \d)");// (中国邮政编码为6位数字)
//IP地址
Regex reg = new Regex(@"\d+\.\d+\.\d+\.\d+");// (提取IP地址时有用)
//IP地址
Regex reg = new Regex(@"((?: (?:25[0-5]|2[0-4]\d|[01]?\d?\d)\.){3} (?:25[0-5]|2[0-4]\d|[01]?\d?\d))");

```

使用demo

正则的使用可以分为验证方法和匹配方法两种

```

public class Validator
{
    匹配方法
    验证方法
}

```

因上文对正则已经做了比较详细的讲解,故在此不多做赘述,直接贴出使用demo

69

目

```

1 public class Validator
2     {

```

```
3         #region 匹配方法
4         /// <summary>
5         /// 验证字符串是否匹配正则表达式描述的规则
6         /// </summary>
7         /// <param name="inputStr">待验证的字符串
</param>
8         /// <param name="patternStr">正则表达式字
字符串</param>
9         /// <returns>是否匹配</returns>
10        public static bool IsMatch(string
inputStr, string patternStr)
11        {
12            return IsMatch(inputStr, patternStr,
false, false);
13        }
14
15        /// <summary>
16        /// 验证字符串是否匹配正则表达式描述的规则
17        /// </summary>
18        /// <param name="inputStr">待验证的字符串
</param>
19        /// <param name="patternStr">正则表达式字
字符串</param>
20        /// <param name="ifIgnoreCase">匹配时是否
不区分大小写</param>
21        /// <returns>是否匹配</returns>
22        public static bool IsMatch(string
inputStr, string patternStr, bool ifIgnoreCase)
23        {
24            return IsMatch(inputStr, patternStr,
ifIgnoreCase, false);
25        }
26
27        /// <summary>
28        /// 验证字符串是否匹配正则表达式描述的规则
29        /// </summary>
30        /// <param name="inputStr">待验证的字符串
</param>
31        /// <param name="patternStr">正则表达式字
字符串</param>
32        /// <param name="ifValidateWhiteSpace">
是否验证空白字符串</param>
33        /// <returns>是否匹配</returns>
34        public static bool IsMatch(string
inputStr, string patternStr, bool ifValidateWhiteSpace)
35        {
36            return IsMatch(inputStr, patternStr,
false, ifValidateWhiteSpace);
37        }
```

```
38
39          /// <summary>
40          /// 验证字符串是否匹配正则表达式描述的规则
41          /// </summary>
42          /// <param name="inputStr">待验证的字符串
</param>
43          /// <param name="patternStr">正则表达式字
字符串</param>
44          /// <param name="ifIgnoreCase">匹配时是否
不区分大小写</param>
45          /// <param name="ifValidateWhiteSpace">
是否验证空白字符串</param>
46          /// <returns>是否匹配</returns>
47          public static bool IsMatch(string
inputStr, string patternStr, bool ifIgnoreCase, bool
ifValidateWhiteSpace)
48          {
49              if (!ifValidateWhiteSpace &&
string.IsNullOrEmpty(inputStr))//.NET 4.0 新增
IsNullOrEmpty 方法, 便于对用户做处理
50                  return false;//如果不要求验证空白字
字符串而此时传入的待验证字符串为空白字符串, 则不匹配
51              Regex regex = null;
52              if (ifIgnoreCase)
53                  regex = new Regex(patternStr,
RegexOptions.IgnoreCase);//指定不区分大小写的匹配
54              else
55                  regex = new Regex(patternStr);
56              return regex.IsMatch(inputStr);
57          }
58          #endregion
59
60          #region 验证方法
61          /// <summary>
62          /// 验证数字(double类型)
63          /// [可以包含负号和小数点]
64          /// </summary>
65          /// <param name="input">待验证的字符串
</param>
66          /// <returns>是否匹配</returns>
67          public static bool IsNumber(string
input)
68          {
69              //string pattern = @"^-?\d+$|^(-?
\d+)(\.\d+)?$";
70              //return IsMatch(input, pattern);
71              double d = 0;
72              if (double.TryParse(input, out d))
73                  return true;
```

```
74             else
75                 return false;
76         }
77
78         /// <summary>
79         /// 验证整数
80         /// </summary>
81         /// <param name="input">待验证的字符串
</param>
82         /// <returns>是否匹配</returns>
83         public static bool IsInteger(string
input)
84         {
85             //string pattern = @"^-?\d+$";
86             //return IsMatch(input, pattern);
87             int i = 0;
88             if (int.TryParse(input, out i))
89                 return true;
90             else
91                 return false;
92         }
93
94         /// <summary>
95         /// 验证非负整数
96         /// </summary>
97         /// <param name="input">待验证的字符串
</param>
98         /// <returns>是否匹配</returns>
99         public static bool
IsIntegerNotNagtive(string input)
100         {
101             //string pattern = @"^\d+$";
102             //return IsMatch(input, pattern);
103             int i = -1;
104             if (int.TryParse(input, out i) && i
>= 0)
105                 return true;
106             else
107                 return false;
108         }
109
110         /// <summary>
111         /// 验证正整数
112         /// </summary>
113         /// <param name="input">待验证的字符串
</param>
114         /// <returns>是否匹配</returns>
115         public static bool
IsIntegerPositive(string input)
```

```
116         {
117             //string pattern = @"^[0-9]*
[1-9][0-9]*$";
118             //return.IsMatch(input, pattern);
119             int i = 0;
120             if (int.TryParse(input, out i) && i
>= 1)
121                 return true;
122             else
123                 return false;
124         }
125
126         /// <summary>
127         /// 验证小数
128         /// </summary>
129         /// <param name="input">待验证的字符串
</param>
130         /// <returns>是否匹配</returns>
131         public static bool IsDecimal(string
input)
132         {
133             string pattern = @"^([+]?[1-9]\d*
\.\d+|-?0\.\d*[1-9]\d*)$";
134             return IsMatch(input, pattern);
135         }
136
137         /// <summary>
138         /// 验证只包含英文字母
139         /// </summary>
140         /// <param name="input">待验证的字符串
</param>
141         /// <returns>是否匹配</returns>
142         public static bool
IsEnglishCharacter(string input)
143         {
144             string pattern = @"^[A-Za-z]+$";
145             return IsMatch(input, pattern);
146         }
147
148         /// <summary>
149         /// 验证只包含数字和英文字母
150         /// </summary>
151         /// <param name="input">待验证的字符串
</param>
152         /// <returns>是否匹配</returns>
153         public static bool
IsIntegerAndEnglishCharacter(string input)
154         {
155             string pattern = @"^[0-9A-Za-z]+$";
```



```
156             return IsMatch(input, pattern);
157         }
158
159         /// <summary>
160         /// 验证只包含汉字
161         /// </summary>
162         /// <param name="input">待验证的字符串
</param>
163         /// <returns>是否匹配</returns>
164         public static bool
IsChineseCharacter(string input)
165         {
166             string pattern = @"^[\u4e00-
\u9fa5]+$";
167             return IsMatch(input, pattern);
168         }
169
170         /// <summary>
171         /// 验证数字长度范围 (数字前端的0计长度)
172         /// [若要验证固定长度, 可传入相同的两个长度数
值]
173         /// </summary>
174         /// <param name="input">待验证的字符串
</param>
175         /// <param name="lengthBegin">长度范围起始
值 (含) </param>
176         /// <param name="lengthEnd">长度范围结束值
(含) </param>
177         /// <returns>是否匹配</returns>
178         public static bool
IsIntegerLength(string input, int lengthBegin, int
lengthEnd)
179         {
180             //string pattern = @"^d{" +
lengthBegin + "," + lengthEnd + "}";
181             //return IsMatch(input, pattern);
182             if (input.Length >= lengthBegin &&
input.Length <= lengthEnd)
183             {
184                 int i;
185                 if (int.TryParse(input, out i))
186                     return true;
187                 else
188                     return false;
189             }
190             else
191                 return false;
192         }
193
```

```
194          /// <summary>
195          /// 验证字符串包含内容
196          /// </summary>
197          /// <param name="input">待验证的字符串
</param>
198          /// <param name="withEnglishCharacter">
是否包含英文字母</param>
199          /// <param name="withNumber">是否包含数字
</param>
200          /// <param name="withChineseCharacter">
是否包含汉字</param>
201          /// <returns>是否匹配</returns>
202          public static bool
IsStringInclude(string input, bool withEnglishCharacter,
bool withNumber, bool withChineseCharacter)
203          {
204              if (!withEnglishCharacter &&
!withNumber && !withChineseCharacter)
205                  return false; //如果英文字母、数字和
汉字都没有, 则返回false
206              StringBuilder patternString = new
StringBuilder();
207              patternString.Append("^[" );
208              if (withEnglishCharacter)
209                  patternString.Append("a-zA-Z");
210              if (withNumber)
211                  patternString.Append("0-9");
212              if (withChineseCharacter)
213                  patternString.Append(@"\u4E00-
\u9FA5");
214              patternString.Append("]+$");
215              return IsMatch(input,
patternString.ToString());
216          }
217
218          /// <summary>
219          /// 验证字符串长度范围
220          /// [若要验证固定长度, 可传入相同的两个长度数
值]
221          /// </summary>
222          /// <param name="input">待验证的字符串
</param>
223          /// <param name="lengthBegin">长度范围起始
值 (含) </param>
224          /// <param name="lengthEnd">长度范围结束值
(含) </param>
225          /// <returns>是否匹配</returns>
226          public static bool IsStringLength(string
input, int lengthBegin, int lengthEnd)
```

```
227         {
228             //string pattern = @"^.{ " +
lengthBegin + "," + lengthEnd + "}$";
229             //return.IsMatch(input, pattern);
230             if (input.Length >= lengthBegin &&
input.Length <= lengthEnd)
231                 return true;
232             else
233                 return false;
234         }
235
236         /// <summary>
237         /// 验证字符串长度范围 (字符串内只包含数字和/或
英文字母)
238         /// [若要验证固定长度, 可传入相同的两个长度数
值]
239         /// </summary>
240         /// <param name="input">待验证的字符串
</param>
241         /// <param name="lengthBegin">长度范围起始
值 (含) </param>
242         /// <param name="lengthEnd">长度范围结束值
(含) </param>
243         /// <returns>是否匹配</returns>
244         public static bool
IsStringLengthOnlyNumberAndEnglishCharacter(string
input, int lengthBegin, int lengthEnd)
245         {
246             string pattern = @"^[0-9a-zA-z]{ " +
lengthBegin + "," + lengthEnd + "}$";
247             return IsMatch(input, pattern);
248         }
249
250         /// <summary>
251         /// 验证字符串长度范围
252         /// [若要验证固定长度, 可传入相同的两个长度数
值]
253         /// </summary>
254         /// <param name="input">待验证的字符串
</param>
255         /// <param name="withEnglishCharacter">
是否包含英文字母</param>
256         /// <param name="withNumber">是否包含数字
</param>
257         /// <param name="withChineseCharacter">
是否包含汉字</param>
258         /// <param name="lengthBegin">长度范围起始
值 (含) </param>
259         /// <param name="lengthEnd">长度范围结束值
```

```
(含) </param>
260          /// <returns>是否匹配</returns>
261          public static bool
IsStringLengthByInclude(string input, bool
withEnglishCharacter, bool withNumber, bool
withChineseCharacter, int lengthBegin, int lengthEnd)
262          {
263              if (!withEnglishCharacter &&
!withNumber && !withChineseCharacter)
264                  return false; //如果英文字母、数字和
汉字都没有, 则返回false
265              StringBuilder patternString = new
StringBuilder();
266              patternString.Append("^[";
267              if (withEnglishCharacter)
268                  patternString.Append("a-zA-Z");
269              if (withNumber)
270                  patternString.Append("0-9");
271              if (withChineseCharacter)
272                  patternString.Append(@"\u4E00-
\u9FA5");
273              patternString.Append("]" +
lengthBegin + "," + lengthEnd + "}$");
274              return IsMatch(input,
patternString.ToString());
275          }
276
277          /// <summary>
278          /// 验证字符串字节数长度范围
279          /// [若要验证固定长度, 可传入相同的两个长度数
值; 每个汉字为两个字节长度]
280          /// </summary>
281          /// <param name="input">待验证的字符串
</param>
282          /// <param name="lengthBegin">长度范围起始
值 (含) </param>
283          /// <param name="lengthEnd">长度范围结束值
(含) </param>
284          /// <returns></returns>
285          public static bool
IsStringByteLength(string input, int lengthBegin, int
lengthEnd)
286          {
287              //int byteLength =
Regex.Replace(input, @"[\x00-\xff]", "ok").Length;
288              //if (byteLength >= lengthBegin &&
byteLength <= lengthEnd)
289                  //{
290                      //    return true;
291                  }
```

```
291             //}
292             //return false;
293             int byteLength =
Encoding.Default.GetByteCount(input);
294             if (byteLength >= lengthBegin &&
byteLength <= lengthEnd)
295                 return true;
296             else
297                 return false;
298         }
299
300         /// <summary>
301         /// 验证日期
302         /// </summary>
303         /// <param name="input">待验证的字符串
</param>
304         /// <returns>是否匹配</returns>
305         public static bool IsDateTime(string
input)
306         {
307             DateTime dt;
308             if (DateTime.TryParse(input, out
dt))
309                 return true;
310             else
311                 return false;
312         }
313
314         /// <summary>
315         /// 验证固定电话号码
316         /// [3位或4位区号; 区号可以用小括号括起来; 区号
可以省略; 区号与本地号间可以用减号或空格隔开; 可以有3位数的分机号,
分机号前要加减号]
317         /// </summary>
318         /// <param name="input">待验证的字符串
</param>
319         /// <returns>是否匹配</returns>
320         public static bool
IsTelePhoneNumber(string input)
321         {
322             string pattern = @"^(((0\d{2}|0
\d{2}))[- ]?\d{8}|((0\d{3}|0\d{3}))[- ]?\d{7})
(-\d{3})?$$";
323             return IsMatch(input, pattern);
324         }
325
326         /// <summary>
327         /// 验证手机号码
328         /// [可匹配"(+86)013325656352", 括号可以省
```

略, +号可以省略, (+86)可以省略, 11位手机号前的0可以省略; 11位手机号第二位数可以是3、4、5、8中的任意一个]

```
329          /// </summary>
330          /// <param name="input">待验证的字符串
</param>
331          /// <returns>是否匹配</returns>
332          public static bool
IsMobilePhoneNumber(string input)
333          {
334              string pattern = @"^((\+)?86|
((\+)?86)?0?1[3458]\d{9})$";
335              return IsMatch(input, pattern);
336          }
337
338          /// <summary>
339          /// 验证电话号码 (可以是固定电话号码或手机号
码)
340          /// [固定电话: [3位或4位区号; 区号可以用小括号
括起来; 区号可以省略; 区号与本地号间可以用减号或空格隔开; 可以有3位
数的分机号, 分机号前要加减号]]
341          /// [手机号码: [可匹配"+86)013325656352",
括号可以省略, +号可以省略, (+86)可以省略, 手机号前的0可以省略; 手
机号第二位数可以是3、4、5、8中的任意一个]]
342          /// </summary>
343          /// <param name="input">待验证的字符串
</param>
344          /// <returns>是否匹配</returns>
345          public static bool IsPhoneNumber(string
input)
346          {
347              string pattern = @"^((\+)?86|
((\+)?86)?0?1[3458]\d{9})$|^(((0\d2|0\d{2})[- ]?)?\d{8}|
((0\d3|0\d{3})[- ]?)?\d{7})(-\d{3})?$";
348              return IsMatch(input, pattern);
349          }
350
351          /// <summary>
352          /// 验证邮政编码
353          /// </summary>
354          /// <param name="input">待验证的字符串
</param>
355          /// <returns>是否匹配</returns>
356          public static bool IsZipCode(string
input)
357          {
358              //string pattern = @"^\d{6}$";
359              //return IsMatch(input, pattern);
360              if (input.Length != 6)
361                  return false;
```

```
362             int i;
363             if (int.TryParse(input, out i))
364                 return true;
365             else
366                 return false;
367         }
368
369         /// <summary>
370         /// 验证电子邮箱
371         ///  [@字符前可以包含字母、数字、下划线和点号; @
372         字符后可以包含字母、数字、下划线和点号; @字符后至少包含一个点号且点
373         号不能是最后一个字符; 最后一个点号后只能是字母或数字]
374         /// </summary>
375         /// <param name="input">待验证的字符串
376     </param>
377         /// <returns>是否匹配</returns>
378     public static bool IsEmail(string input)
379     {
380         /// 邮箱名以数字或字母开头; 邮箱名可由字
381         母、数字、点号、减号、下划线组成; 邮箱名 (@前的字符) 长度为3~18个
382         字符; 邮箱名不能以点号、减号或下划线结尾; 不能出现连续两个或两个以
383         上的点号、减号。
384         //string pattern = @"^[a-zA-Z0-9]((?
385         <!(\.\.|\--)) [a-zA-Z0-9\._-]) {1,16} [a-zA-Z0-9]@([0-9a-zA-
386         Z] [0-9a-zA-Z-] {0,62} \.)+ ([0-9a-zA-Z] [0-9a-zA-Z-] {0,62})
387         \. ? | ((25 [0-5] | 2 [0-4] \d | [01] ? \d \d ?) \. ) {3} (25 [0-5] | 2 [0-4]
388         \d | [01] ? \d \d ?) $";
389         string pattern = @"^([\w-\.] +)@([\w-
390         \.] +) (\.[a-zA-Z0-9] +) $";
391         return IsMatch(input, pattern);
392     }
393
394     /// <summary>
395     /// 验证网址 (可以匹配IPv4地址但没对IPv4地址进
396     行格式验证; IPv6暂时没做匹配)
397     /// [允许省略"://"; 可以添加端口号; 允许层级;
398     允许传参; 域名中至少一个点号且此点号前要有内容]
399     /// </summary>
400     /// <param name="input">待验证的字符串
401 </param>
402     /// <returns>是否匹配</returns>
403     public static bool IsURL(string input)
404     {
405         /// 每级域名由字母、数字和减号构成 (第一个
406         字母不能是减号), 不区分大小写, 单个域长度不超过63, 完整的域名全长
407         不超过256个字符。在DNS系统中, 全名是以一个点"."来结束的, 例如
408         "www.nit.edu.cn."。没有最后的那个点则表示一个相对地址。
409         /// 没有例如"http://"的前缀, 没有传参的匹
410         配
```

```
393             //string pattern = @"^([0-9a-zA-Z][0-9a-zA-Z-]{0,62}\.)+([0-9a-zA-Z][0-9a-zA-Z-]{0,62})\.$";
394
395             //string pattern =
396             @"^(((file|gopher|news|nntp|telnet|http|ftp|https|ftp|sftp)://)|(www\.))+([a-zA-Z0-9\._-]+\.[a-zA-Z]{2,6})|([0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}))(/[a-zA-Z0-9\&%\_\.\/-~]*)?$";
397             string pattern = @"^([a-zA-Z]+://)?([\w-\.]+\.[a-zA-Z0-9]+)(:\d{0,5})?/?([\w-/*]*\.[a-zA-Z]*)\???([\w-]*=[\w%]*&?)*$";
398             return IsMatch(input, pattern);
399         }
400
401         /// <summary>
402         /// 验证IPv4地址
403         /// [第一位和最后一位数字不能是0或255; 允许用0补位]
404         /// </summary>
405         /// <param name="input">待验证的字符串
406         </param>
407         /// <returns>是否匹配</returns>
408         public static bool IsIPv4(string input)
409         {
410             //string pattern =
411             @"^(25[0-4]|2[0-4]\d|[01]?\d{2}|[1-9])\. (25[0-5]|2[0-4]\d|[01]?\d?\d)\. (25[0-5]|2[0-4]\d|[01]?\d?\d)\. (25[0-4]|2[0-4]\d|[01]?\d{2}|[1-9])$";
412             //return IsMatch(input, pattern);
413             string[] IPs = input.Split('.');
414             if (IPs.Length != 4)
415                 return false;
416             int n = -1;
417             for (int i = 0; i < IPs.Length; i++)
418             {
419                 if (i == 0 || i == 3)
420                 {
421                     if (int.TryParse(IPs[i], out n) && n > 0 && n < 255)
422                         continue;
423                     else
424                         return false;
425                 }
426                 else
427                 {
428                     if (int.TryParse(IPs[i], out n) && n >= 0 && n <= 255)
429                         continue;
430                 }
431             }
432             return false;
433         }
434     }
435 }
```



```
427             else
428                 return false;
429         }
430     }
431     return true;
432 }
433
434     /// <summary>
435     /// 验证IPv6地址
436     /// [可用于匹配任何一个合法的IPv6地址]
437     /// </summary>
438     /// <param name="input">待验证的字符串
</param>
439     /// <returns>是否匹配</returns>
440     public static bool IsIPv6(string input)
441     {
442         string pattern = @"^\s*((([0-9A-Fa-f]{1,4}:){7}([0-9A-Fa-f]{1,4}:)|((([0-9A-Fa-f]{1,4}:){6}(:[0-9A-Fa-f]{1,4}|((25[0-5]|2[0-4]\d|1\d\d|[1-9]?
\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?d))){3})|:))|
((([0-9A-Fa-f]{1,4}:){5}((([0-9A-Fa-f]{1,4}){1,2})|:
((25[0-5]|2[0-4]\d|1\d\d|[1-9]?d)(\.(25[0-5]|2[0-4]
\d|1\d\d|[1-9]?d))){3})|:))|((([0-9A-Fa-f]{1,4}){4}((:
[0-9A-Fa-f]{1,4}){1,3})|(:[0-9A-Fa-f]{1,4})?:
((25[0-5]|2[0-4]\d|1\d\d|[1-9]?d)(\.(25[0-5]|2[0-4]
\d|1\d\d|[1-9]?d))){3}))|:))|((([0-9A-Fa-f]{1,4}){3}(((
[0-9A-Fa-f]{1,4}){1,4})|(:[0-9A-Fa-f]{1,4}){0,2}:
((25[0-5]|2[0-4]\d|1\d\d|[1-9]?d)(\.(25[0-5]|2[0-4]
\d|1\d\d|[1-9]?d))){3}))|:))|((([0-9A-Fa-f]{1,4}){2}(((
[0-9A-Fa-f]{1,4}){1,5})|(:[0-9A-Fa-f]{1,4}){0,3}:
((25[0-5]|2[0-4]\d|1\d\d|[1-9]?d)(\.(25[0-5]|2[0-4]
\d|1\d\d|[1-9]?d))){3}))|:))|((([0-9A-Fa-f]{1,4}){1}(((
[0-9A-Fa-f]{1,4}){1,6})|(:[0-9A-Fa-f]{1,4}){0,4}:
((25[0-5]|2[0-4]\d|1\d\d|[1-9]?d)(\.(25[0-5]|2[0-4]
\d|1\d\d|[1-9]?d))){3}))|:))|(:((( [0-9A-Fa-f]{1,4})
{1,7})|(:[0-9A-Fa-f]{1,4}){0,5}:((25[0-5]|2[0-4]\d|1\d
\d|[1-9]?d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?
\d))){3}))|:))) (%.+)?\s*$";
443         return IsMatch(input, pattern);
444     }
445
446     /// <summary>
447     /// 身份证上数字对应的地址
448     /// </summary>
449     ///enum IDAddress
450     ///{
451     ///    北京 = 11, 天津 = 12, 河北 = 13, 山西
= 14, 内蒙古 = 15, 辽宁 = 21, 吉林 = 22, 黑龙江 = 23, 上海 =
31, 江苏 = 32, 浙江 = 33,
```

```
452          //    安徽 = 34, 福建 = 35, 江西 = 36, 山东
= 37, 河南 = 41, 湖北 = 42, 湖南 = 43, 广东 = 44, 广西 =
45, 海南 = 46, 重庆 = 50, 四川 = 51,
453          //    贵州 = 52, 云南 = 53, 西藏 = 54, 陕西
= 61, 甘肃 = 62, 青海 = 63, 宁夏 = 64, 新疆 = 65, 台湾 =
71, 香港 = 81, 澳门 = 82, 国外 = 91
454          //}
455
456          /// <summary>
457          /// 验证一代身份证号 (15位数)
458          /// [长度为15位的数字; 匹配对应省份地址; 生日能
正确匹配]
459          /// </summary>
460          /// <param name="input">待验证的字符串
</param>
461          /// <returns>是否匹配</returns>
462          public static bool IsIDCard15(string
input)
463          {
464              //验证是否可以转换为15位整数
465              long l = 0;
466              if (!long.TryParse(input, out l) ||
l.ToString().Length != 15)
467              {
468                  return false;
469              }
470              //验证省份是否匹配
471              //1~6位为地区代码, 其中1、2位数为各省级政
府的代码, 3、4位数为地、市级政府的代码, 5、6位数为县、区级政府代
码。
472              string address =
"11,12,13,14,15,21,22,23,31,32,33,34,35,36,37,41,42,43,4
4,45,46,50,51,52,53,54,61,62,63,64,65,71,81,82,91,";
473              if
(!address.Contains(input.Remove(2) + ","))
474              {
475                  return false;
476              }
477              //验证生日是否匹配
478              string birthdate =
input.Substring(6, 6).Insert(4, "/").Insert(2, "/");
479              DateTime dt;
480              if (!DateTime.TryParse(birthdate,
out dt))
481              {
482                  return false;
483              }
484              return true;
485          }
```

```
486
487         /// <summary>
488         /// 验证二代身份证号 (18位数, GB11643-1999标准)
489         /// [长度为18位; 前17位为数字, 最后一位(校验码)可以为大小写x; 匹配对应省份地址; 生日能正确匹配; 校验码能正确匹配]
490         /// </summary>
491         /// <param name="input">待验证的字符串
</param>
492         /// <returns>是否匹配</returns>
493         public static bool IsIDCard18(string
input)
494         {
495             ///验证是否可以转换为正确的整数
496             long l = 0;
497             if (!long.TryParse(input.Remove(17),
out l) || l.ToString().Length != 17 ||
!long.TryParse(input.Replace('x', '0').Replace('X',
'0'), out l))
498             {
499                 return false;
500             }
501             ///验证省份是否匹配
502             ///1~6位为地区代码, 其中1、2位数为各省级政府的代码, 3、4位数为地、市级政府的代码, 5、6位数为县、区级政府代码。
503             string address =
"11,12,13,14,15,21,22,23,31,32,33,34,35,36,37,41,42,43,4
4,45,46,50,51,52,53,54,61,62,63,64,65,71,81,82,91,";
504             if
(!address.Contains(input.Remove(2) + ","))
505             {
506                 return false;
507             }
508             ///验证生日是否匹配
509             string birthdate =
input.Substring(6, 8).Insert(6, "/").Insert(4, "/");
510             DateTime dt;
511             if (!DateTime.TryParse(birthdate,
out dt))
512             {
513                 return false;
514             }
515             ///校验码验证
516             ///校验码:
517             /// (1) 十七位数字本体码加权求和公式
518             /// S = Sum(Ai * Wi), i = 0, ... , 16
, 先对前17位数字的权求和
```

```
519          //Ai:表示第i位置上的身份证号码数字值
520          //Wi:表示第i位置上的加权因子
521          //Wi: 7 9 10 5 8 4 2 1 6 3 7 9 10 5
522          // (2) 计算模
523          //Y = mod(S, 11)
524          // (3) 通过模得到对应的校验码
525          //Y: 0 1 2 3 4 5 6 7 8 9 10
526          //校验码: 1 0 X 9 8 7 6 5 4 3 2
527          string[] arrVarifyCode =
528          ("1,0,x,9,8,7,6,5,4,3,2").Split(',');
529          string[] Wi =
530          ("7,9,10,5,8,4,2,1,6,3,7,9,10,5,8,4,2").Split(',');
531          char[] Ai =
532          input.Remove(17).ToCharArray();
533          int sum = 0;
534          for (int i = 0; i < 17; i++)
535          {
536              sum += int.Parse(Wi[i]) *
537              int.Parse(Ai[i].ToString());
538          }
539          int y = -1;
540          Math.DivRem(sum, 11, out y);
541          if (arrVarifyCode[y] !=
542          input.Substring(17, 1).ToLower())
543          {
544              return false;
545          }
546          return true;
547      }
548  }
549  /// <summary>
550  /// 验证身份证号 (不区分一二代身份证号)
551  /// </summary>
552  /// <param name="input">待验证的字符串
553  </param>
554  /// <returns>是否匹配</returns>
555  public static bool IsIDCard(string
556  input)
557  {
558      if (input.Length == 18)
559          return IsIDCard18(input);
560      else if (input.Length == 15)
561          return IsIDCard15(input);
562      else
563          return false;
564  }
565  }
566  /// <summary>
```

```
560          /// 验证经度
561          /// </summary>
562          /// <param name="input">待验证的字符串
</param>
563          /// <returns>是否匹配</returns>
564          public static bool IsLongitude(string
input)
565          {
566              ///范围-180~180, 小数位数必须是1到5位
567              //string pattern = @"^[-\+]?((1[0-7]
\d{1}|0?\d{1,2})\.\d{1,5}|180\.\d{1,5})$";
568              //return IsMatch(input, pattern);
569              float lon;
570              if (float.TryParse(input, out lon)
&& lon >= -180 && lon <= 180)
571                  return true;
572              else
573                  return false;
574          }
575
576          /// <summary>
577          /// 验证纬度
578          /// </summary>
579          /// <param name="input">待验证的字符串
</param>
580          /// <returns>是否匹配</returns>
581          public static bool IsLatitude(string
input)
582          {
583              ///范围-90~90, 小数位数必须是1到5位
584              //string pattern = @"^[-\+]?([0-8]?
\d{1}\.\d{1,5}|90\.\d{1,5})$";
585              //return IsMatch(input, pattern);
586              float lat;
587              if (float.TryParse(input, out lat)
&& lat >= -90 && lat <= 90)
588                  return true;
589              else
590                  return false;
591          }
592          #endregion
593      }
```

注：感谢 Sam Xiao 对RegEx类的实例用法作出补充

- 感谢你的阅读。如果你觉得这篇文章对你有帮助或者有启发，就请推荐一下吧~你的精神支持是博主强大的写作动力。欢迎转载！

- 博主的文章没有高度、深度和广度，只是凑字数。由于博主的水平不高（其实是个菜B），不足和错误之处在所难免，希望大家能够批评指出。
- 欢迎加入.NET 从入门到精通技术讨论群→[523490820](#) 期待你的加入
- 不舍得打乱，就永远学不会复原。被人嘲笑梦想，才更有实现的价值。
- 我的博客：<http://www.cnblogs.com/zhangxiaoyong/>

分类: [C#,前端](#)

[好文要顶](#)[关注我](#)[收藏该文](#)

潇十一郎

关注 - 37

粉丝 - 117

[+加关注](#)

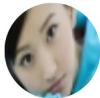
« 上一篇: [可输入的下拉框（简易实现）](#)

» 下一篇: [PowerDesigner（数据建模）使用大全](#)

posted @ 2016-11-08 15:41 [潇十一郎](#) 阅读(25812) 评论(23) [编辑](#) [收藏](#)

评论列表

#1楼 2016-11-08 15:47 xiao99



专业啊

[支持\(1\)](#) [反对\(0\)](#)

#2楼[楼主] 2016-11-08 15:51 潇十一郎

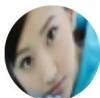


@ xiao99

一起加油，你名字里面也有个xiao? me too

[支持\(0\)](#) [反对\(0\)](#)

#3楼 2016-11-08 16:31 xiao99



@ 潇十一郎

我姓肖

[支持\(0\)](#) [反对\(0\)](#)

#4楼 2016-11-09 00:31 杰哥很忙



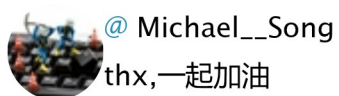
支持(1) 反对(0)

#5楼 2016-11-09 09:04 Michael_Song



支持(1) 反对(0)

#6楼[楼主] 2016-11-09 09:06 潇十一郎



支持(0) 反对(0)

#7楼[楼主] 2016-11-09 09:06 潇十一郎



支持(0) 反对(0)

#8楼 2016-11-09 12:56 Sam Xiao



再配上我这一段，就完美了。

```
1 //提取[]的值
2 string pattern1 = @"(?is)(?<=\[)(.*)(?=\])";
3 string result1 = new Regex(pattern1).Match("s
4 //提取()的值
5 string pattern2 = @"(?is)(?<=\()(.*)(?=\))";
6 string result2 = new Regex(pattern2).Match("s
7 //提取{}的值
8 string pattern3 = @"(?is)(?<=\{)(.*)(?=\})";
9 string result3 = new Regex(pattern3).Match("s
```

[刷单软件](#)

[阿里云客户端](#)

支持(1) 反对(0)

#9楼[楼主] 2016-11-09 13:13 潇十一郎



@ Sam Xiao

哈哈，锦上添花

支持(0) 反对(0)

#10楼 2016-11-09 13:38 EarnestMen



不错。点个赞

支持(1) 反对(0)

#11楼 2016-11-09 13:48 懒得安分



辛苦，支持下！

支持(1) 反对(0)

#12楼[楼主] 2016-11-09 13:49 潇十一郎



@ 懒得安分

多谢，共同学习~

支持(0) 反对(0)

#13楼 2016-11-09 14:12 郭小宅



幸苦了

支持(1) 反对(0)

#14楼 2016-11-09 14:18 寒@鹏



ajaxFileUpload

支持(1) 反对(0)

#15楼[楼主] 2016-11-09 15:48 潇十一郎



@ 郭小宅

thx，一起学习，也当个总结，日后免得到处找

支持(0) 反对(0)

#16楼[楼主] 2016-11-09 15:48 潇十一郎



@ 寒@鹏

兄台有何高招？

支持(0) 反对(0)

#17楼 2016-11-09 15:52 一瓢清水



我姓肖+1

支持(0) 反对(0)

#18楼 2016-11-09 16:48 寒@鹏



@ 潇十一郎

引用

@寒@鹏

兄台有何高招?

标示 mark O(∩_∩)O哈! 谢谢了

支持(0) 反对(0)

#19楼 2017-02-17 14:41 Sam Xiao



0\d2|0\d{2} 匹配固定电话那儿的区号的时候, 这样写的作用是什么啊?

单独要 0\d2 或者 0\d{2} 一部分不就行了吗, 为什么要用|符号把这两部分连接起来呢?

支持(0) 反对(0)

#20楼 2017-05-08 15:29 反骨仔



收藏下, 点击你的“首页”发现加载好慢, 等的时间好久

支持(0) 反对(0)

#21楼[楼主] 2017-05-08 15:46 潇十一郎



@ 反骨仔 (二五仔)

哈哈, 是, 我故意设置了三秒延时为了展示首页那个动画效果, 哈哈

支持(0) 反对(0)

#22楼 2018-05-09 18:44 活着1



我就试了一下楼主符号的例子, 发现就有错, 伤不起啊。

支持(0) 反对(0)

#23楼 2018-06-18 21:58 The--One

@ Sam Xiao



友

新人请教一下，正则表达式中的 (? is) 什么意思啊?

支持(0) 反对(0)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

【推荐】超50万VC++源码: 大型组态工控、电力仿真CAD与GIS源码库!

【推荐】如何快速搭建人工智能应用?

【大赛】2018首届“顶天立地”AI开发者大赛



Copyright ©2018 潇十一郎

友情链接: [SOJSON在线工具](#) [潇十一郎](#)