### 字符转义

下表列出了转义字符：

|  |  |  |  |
| --- | --- | --- | --- |
| **转义字符** | **描述** | **模式** | **匹配** |
| **\a** | 与报警 (bell) 符 \u0007 匹配。 | \a | "Warning!" + '\u0007' 中的 "\u0007" |
| **\b** | 在字符类中，与退格键 \u0008 匹配。 | [\b]{3,} | "\b\b\b\b" 中的 "\b\b\b\b" |
| **\t** | 与制表符 \u0009 匹配。 | (\w+)\t | "Name\tAddr\t" 中的 "Name\t" 和 "Addr\t" |
| **\r** | 与回车符 \u000D 匹配。（\r 与换行符 \n 不是等效的。） | \r\n(\w+) | "\r\Hello\nWorld." 中的 "\r\nHello" |
| **\v** | 与垂直制表符 \u000B 匹配。 | [\v]{2,} | "\v\v\v" 中的 "\v\v\v" |
| **\f** | 与换页符 \u000C 匹配。 | [\f]{2,} | "\f\f\f" 中的 "\f\f\f" |
| **\n** | 与换行符 \u000A 匹配。 | \r\n(\w+) | "\r\Hello\nWorld." 中的 "\r\nHello" |
| **\e** | 与转义符 \u001B 匹配。 | \e | "\x001B" 中的 "\x001B" |
| **\ nnn** | 使用八进制表示形式指定一个字符（nnn 由二到三位数字组成）。 | \w\040\w | "a bc d" 中的 "a b" 和 "c d" |
| **\x nn** | 使用十六进制表示形式指定字符（nn 恰好由两位数字组成）。 | \w\x20\w | "a bc d" 中的 "a b" 和 "c d" |
| **\c X \c x** | 匹配 X 或 x 指定的 ASCII 控件字符，其中 X 或 x 是控件字符的字母。 | \cC | "\x0003" 中的 "\x0003" (Ctrl-C) |
| **\u nnnn** | 使用十六进制表示形式匹配一个 Unicode 字符（由 nnnn 表示的四位数）。 | \w\u0020\w | "a bc d" 中的 "a b" 和 "c d" |
| **\** | 在后面带有不识别的转义字符时，与该字符匹配。 | \d+[\+-x\\*]\d+\d+[\+-x\\*\d+ | "(2+2) \* 3\*9" 中的 "2+2" 和 "3\*9" |

### 字符类

字符类与一组字符中的任何一个字符匹配。

下表列出了字符类：

|  |  |  |  |
| --- | --- | --- | --- |
| **字符类** | **描述** | **模式** | **匹配** |
| **[character\_group]** | 匹配 character\_group 中的任何单个字符。 | [mn] | "mat" 中的 "m"，"moon" 中的 "m" 和 "n" |
| **[^character\_group]** | 不在 character\_group 中的任何单个字符匹配。 | [^aei] | "avail" 中的 "v" 和 "l" |
| **[ first - last ]** | 字符范围：与从 first 到 last 的范围中的任何单个字符匹配。 | (\w+)\t | "Name\tAddr\t" 中的 "Name\t" 和 "Addr\t" |
| **.** | 通配符：与除 \n 之外的任何单个字符匹配。 | a.e | "have" 中的 "ave"， "mate" 中的 "ate" |
| **\w** | 与任何单词字符匹配。 | \w | "Room#1" 中的 "R"、 "o"、 "m" 和 "1" |
| **\W** | 与任何非单词字符匹配。 | \W | "Room#1" 中的 "#" |
| **\s** | 与任何空白字符匹配。 | \w\s | "ID A1.3" 中的 "D " |
| **\S** | 与任何非空白字符匹配。 | \s\S | "int \_\_ctr" 中的 " \_" |
| **\d** | 与任何十进制数字匹配。 | \d | "4 = IV" 中的 "4" |
| **\D** | 匹配不是十进制数的任意字符。 | \D | "4 = IV" 中的 " "、 "="、 " "、 "I" 和 "V" |

### 定位点

下表列出了定位点：

|  |  |  |  |
| --- | --- | --- | --- |
| **断言** | **描述** | **模式** | **匹配** |
| **^** | 匹配必须从字符串或一行的开头开始。 | ^\d{3} | "567-777-" 中的 "567" |
| **$** | 匹配必须出现在字符串的末尾或出现在行或字符串末尾的 **\n** 之前。 | -\d{4}$ | "8-12-2012" 中的 "-2012" |

### 分组构造

分组构造描述了正则表达式的子表达式，通常用于捕获输入字符串的子字符串。

下表列出了分组构造：

|  |  |  |  |
| --- | --- | --- | --- |
| **分组构造** | **描述** | **模式** | **匹配** |
| **( subexpression )** | 捕获匹配的子表达式并将其分配到一个从零开始的序号中。 |  |  |
| **(?<name >subexpression)** | 将匹配的子表达式捕获到一个命名组中。 |  |  |

### 限定符

下表列出了限定符：

|  |  |  |  |
| --- | --- | --- | --- |
| **限定符** | **描述** | **模式** | **匹配** |
| **\*** | 匹配上一个元素零次或多次。 | \d\*\.\d | ".0"、 "19.9"、 "219.9" |
| **+** | 匹配上一个元素一次或多次。 | "be+" | "been" 中的 "bee"， "bent" 中的 "be" |
| **?** | 匹配上一个元素零次或一次。 | "rai?n" | "ran"、 "rain" |
| **{ n }** | 匹配上一个元素恰好 n 次。 | ",\d{3}" | "1,043.6" 中的 ",043"， "9,876,543,210" 中的 ",876"、 ",543" 和 ",210" |
| **{ n ,}** | 匹配上一个元素至少 n 次。 | "\d{2,}" | "166"、 "29"、 "1930" |
| **{ n , m }** | 匹配上一个元素至少 n 次，但不多于 m 次。 | "\d{3,5}" | "166"， "17668"， "193024" 中的 "19302" |
| **\*?** | 匹配上一个元素零次或多次，但次数尽可能少。 | \d\*?\.\d | ".0"、 "19.9"、 "219.9" |
| **+?** | 匹配上一个元素一次或多次，但次数尽可能少。 | "be+?" | "been" 中的 "be"， "bent" 中的 "be" |
| **??** | 匹配上一个元素零次或一次，但次数尽可能少。 | "rai??n" | "ran"、 "rain" |
| **{ n }?** | 匹配前导元素恰好 n 次。 | ",\d{3}?" | "1,043.6" 中的 ",043"， "9,876,543,210" 中的 ",876"、 ",543" 和 ",210" |
| **{ n ,}?** | 匹配上一个元素至少 n 次，但次数尽可能少。 | "\d{2,}?" | "166"、 "29" 和 "1930" |
| **{ n , m }?** | 匹配上一个元素的次数介于 n 和 m 之间，但次数尽可能少。 | "\d{3,5}?" | "166"， "17668"， "193024" 中的 "193" 和 "024" |

## Regex 类

Regex 类用于表示一个正则表达式。

下表列出了 Regex 类中一些常用的方法：

|  |  |
| --- | --- |
| **序号** | **方法 & 描述** |
| 1 | **public bool IsMatch( string input )** 指示 Regex 构造函数中指定的正则表达式是否在指定的输入字符串中找到匹配项。 |
| 2 | **public bool IsMatch( string input, int startat )** 指示 Regex 构造函数中指定的正则表达式是否在指定的输入字符串中找到匹配项，从字符串中指定的开始位置开始。 |
| 3 | **public static bool IsMatch( string input, string pattern )** 指示指定的正则表达式是否在指定的输入字符串中找到匹配项。 |
| 4 | **public MatchCollection Matches( string input )** 在指定的输入字符串中搜索正则表达式的所有匹配项。 |
| 5 | **public string Replace( string input, string replacement )** 在指定的输入字符串中，把所有匹配正则表达式模式的所有匹配的字符串替换为指定的替换字符串。 |
| 6 | **public string[] Split( string input )** 把输入字符串分割为子字符串数组，根据在 Regex 构造函数中指定的正则表达式模式定义的位置进行分割。 |

## 实例 1

下面的实例匹配了以 'S' 开头的单词：

string str = "A Thousand Splendid Suns";

MatchCollection mc = Regex.Matches(str , @"\bS\S\*");

foreach (Match m in mc)

{

Console.WriteLine(m);

}

## 实例 3

下面的实例替换掉多余的空格：

string input = "Hello World ";

string pattern = "\\s+";

string replacement = " ";

Regex rgx = new Regex(pattern);

string result = rgx.Replace(input, replacement);

## 示例4

匹配json中的url

MatchCollection previewUrlMatch = Regex.Matches(args.Html, @"""preview\_url"":""(?<href>[^""\s]\*)""");

foreach (Match previewUrl in previewUrlMatch)

{

Console.WriteLine(previewUrl.Groups["href"].Value);

}