Регулярные выражения

https://docs.microsoft.com/ruru/dotnet/standard/basetypes/regular-expressions

Регулярные выражения

System.Text.RegularExpressions

RegExp, RegEx

- формальный язык поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании метасимволов.
- строка-образец (англ. pattern, «шаблон», «маска»), состоящая из символов и метасимволов и задающая правило поиска.

https://docs.microsoft.com/ru-ru/dotnet/standard/base-types/regular-expressions

1) Подключить

```
using System.Text.RegularExpressions;
```

2) Создать экземпляр

```
Regex newReg = new Regex(pattern,RegexOptions.None);
```

RegexOptions option = RegexOptions.;

- IgnoreCase
- ExplicitCapture
- CultureInvariant
- IgnorePatternWhitespace
- Multiline
- None
- RightToLeft
- Singleline
- Compiled

3) Все найденные соответствия в тексте помещаются в тип MatchCollection

```
MatchCollection matches;
```

4) Поместить текст, в котором необходимо произвести поиск:

```
matches= newReg.Matches(textoriginal);
```

в matches появляются все результаты парсинга

5) сколько их

```
int i = matches.Count;
```

6) узнать значение конкретного элемента

```
String s= matches[N].Value;
```

7) Проверка на существование

```
if (newReg.IsMatch(textoriginal))
```

8) Просмотр найденных

```
Regex regex = new Regex(pattern);

        Match match = regex.Match(textoriginal);


        while (match.Success)
        {
            int index = match.Index;
            String findStr = match.Value;
            match = match.NextMatch();

            Group allgroup = match.Groups[1];
        }
```

вернуть найденное соответствие из исходной строки

к следующему совпадению

В шаблоне обращение к группе (одни круглые скобки) - ссылаемся на найденное значение через свойство Groups

Варианты использования

1) Подходит ли строка под регулярное выражение - Regex.IsMatch () - true, false

IsMatch(string input, int startat)

строка, позиция для поиска

IsMatch(string input)

IsMatch(string input, string pattern,
System.Text.RegularExpressions.RegexOptions options)

IsMatch(string input, string pattern)

2) Замена текста

Regex.Replace ()

Replace (string input, string pattern, string replacement)

Replace(string input, string replacement)

Replace(string input, string replacement, int count)

Replace(string input, string pattern, string replacement, System.Text.RegularExpressions.RegexOptions options)

3) Разделение одной строки на массив строк

string[]

Split(string input, string pattern)

Split(string input, int count)

Элементы языка регулярных выражений

- http://msdn.microsoft.com/ruru/library/az24scfc.aspx

- система сжатого описания некоторого множества строк с помощью шаблонов

- Обычные символы (литералы)
- специальные символы (метасимволы)
- Escape-символы
- Классы символов
- Привязки
- Конструкции группирования
- Кванторы
- Конструкции обратных ссылок
- Конструкции изменения
- Подстановки
- Прочие конструкции

Escape-символы

\t – символ табуляции
\r – символ возврата каретки
\n –новая строка
\e – символ escape

специальные символы, задающие команды

Метасимволы

[группа_символов] - один из группы

[^группа_символов] – отрицание

[первый-последний] - диапазон символов

\w - любой алфавитно-цифровой знак

\W - не символ

\d - любая десятичная цифра

\D - не цифра

\s - пробел Unicode \S не пробел Unicode

```
[^aei]   -  laetr   →      l t r
[0-9a-fA-F]
```

Привязки (якоря)

атомарные утверждения нулевой ширины, приводят к успеху или сбою сопоставления, в зависимости от текущей позиции в строке

- ^ или \A Начало строки
- $ или \z Конец строки
- \b Граница слова
- \B Не граница слова
- и т.д.

```
^\d{1}   →   948basjfbs   → 9
\d{3}$   →    basjfb948   →  948
```

Конструкции группирования

- часть выражений регулярных выражений (часть_выражения)

"захвата" и сохранения их во встроенных переменных $1, $2, ..., $9 или имя

(on.) → one ->$1
(?:one) → группировка (не запоминает символы, соответствующие этой группе)

Квантор или множители

количество вхождений предшествующего элемента (знака, группы или класса знаков)

- * пред. шаблон , 0 или любое число раз ({0,}).
- + пред. хотя бы 1 раз ({1,}). «o+z»
- ? пред. 0 или 1 раз {0,1}.

be+ beennn → bee, bent → be

\d* → любое количество цифр, идущих подряд

ru?n → run или rn

- {m,n} - повторений может быть от m до n включительно.
- {m,} m и более повторений.
- {,n} не более n повторений.
- и т.д.

```
\d{3,5}   → 234  2345  23456
```

Конструкции изменения

сопоставление по принципу “либо-либо”

| Соответствует любому элементу, разделенному вертикальной чертой (|)

```
th(e|is|at)
```

Приоритеты

Наименование	**Обозначение**
Круглые скобки (группа)	( )(?:...)
Множители	?, +, *, {m,n}
Последовательность и фиксация	abc,\A, \Z
Дизъюнкция	|

Примеры:

Чтение даты в формате YYYY-MM-DD:

```
(\d{4})-(\d\d)-(\d\d)
```

YYYY - в $1, MM - в $2, DD - в $3

```
\s\d+\.\d+\.
```

Пробел цифры точка цифры точка

@"[0-9]+:[0-9]+:[0-9]+“ ??????????

\w+

Любой текстовый символ, не являющийся пробелом, символом табуляции - слово (цифра или буква)

e-mail:

```
\b\w+([\.\w]+)*\w@\w((\.\w)*\w+)*\.\w{2,3}
```

- ^[a-za-я0-9_]{8,20}$
- \.(?:jp(?:e?g|e|2)|gif|png|tiff?|bmp|ico)$