C# Regex Examples

Ask Question

C# Curator Date Feb 10, 2021 1.2m 2 22

Regex in C# defines a regular expression in C#. The Regex class offers methods and properties to parse a large text to find patterns of characters. In this article, you'll learn how to use a Regex class in C#.

Regular Expressions in C#

A regular expression is used to check if a string matches a pattern or not. C# regex also known as C# regular expression or C# regexp is a sequence of characters that defines a pattern. A pattern may consist of literals, numbers, characters, operators, or constructs. The pattern is used to search strings or files to see if matches are found. Regular expressions are often used in input validations, parsing, and finding strings. For example, checking a valid date of birth, social security number, full name where the first and the last names are separated by a comma, finding number of occurrences of a substring, replacing substrings, date formats, valid email formats, a currency format, and so on.

1. C# Regex Class

C# Regex class represents the regular expression engine. It can be used to quickly parse large amounts of text to find specific character patterns; to extract, edit, replace, or delete text substrings; and to add the extracted strings to a collection to generate a report.

In .NET Regex class is defined in the System.Text.RegularExpressions namespace. The Regex class constructor takes a pattern string as a parameter with other optional parameters.

The following code snippet creates a Regex from a pattern. Here pattern is to match a word starting with char 'M'.

```
01. // Create a pattern for a word that starts with letter "M"
```

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


The following code snippet has a long text with author names that needs to be parsed.

```
// Long string
string authors = "Mahesh Chand, Raj Kumar, Mike Gold, Allen O'Neill, Marshal Troll";
```

The Matches method is used to find all the matches in a Regex and returns a MatchCollection.

```
// Get all matches
MatchCollection matchedAuthors = rg.Matches(authors);
```

The following code snippet loops through the matches collection.

```
// Print all matched authors
for (int count = 0; count < matchedAuthors.Count; count++)
Console.WriteLine(matchedAuthors[count].Value);
```

Here is the complete code:

```
// Create a pattern for a word that starts with letter "M"
string pattern = @"\b[M]\w+";
// Create a Regex
Regex rg = new Regex(pattern);

// Long string
string authors = "Mahesh Chand, Raj Kumar, Mike Gold, Allen O'Neill, Marshal Troll";
// Get all matches
MatchCollection matchedAuthors = rg.Matches(authors);
// Print all matched authors
for (int count = 0; count < matchedAuthors.Count; count++)
Console.WriteLine(matchedAuthors[count].Value);
```

In the above example, the code looks for char ‘M’. But what if the word starts with ‘m’. The following code snippet uses RegexOptions.IgnoreCase parameter to make sure that Regex does not look for uppercase or lowercase.

```
// Create a pattern for a word that starts with letter "M"
string pattern = @"\b[m]\w+";
// Create a Regex
Regex rg = new Regex(pattern, RegexOptions.IgnoreCase);
```

The Regex.Replace() method is used to replace a matched string with a new string. The following example finds multiple whiteıg and replaces with a single whitespace.

```
// A long string with ton of white spaces
string badString = "Here is a strig with ton of white space." ;
string CleanedString = Regex.Replace(badString, "\\s+", " ");
Console.WriteLine($"Cleaned String: {CleanedString}");
```

The following code snippet replaces whitespaces with a '-'.

```
string CleanedString = Regex.Replace(badString, "\\s+", "-");
```

3. Replacing multiple white spaces using Regex in C#

The following example uses the regular expression pattern [a-z]+ and the Regex.Split() method to split a string on any uppercase or lowercase alphabetic character.

```
// Spilt a string on alphabetic character
string azpattern = "[a-z]+";
string str = "Asd2323b0900c1234Def5678Ghi9012Jklm";
string[] result = Regex.Split(str, azpattern,
RegexOptions.IgnoreCase, TimeSpan.FromMilliseconds(500));
for (int i = 0; i < result.Length; i++)
{
    Console.Write("'{0}'", result[i]);
    if (i < result.Length - 1)
        Console.Write(", ");
}
```

Regular Expressions in C#

Regular expressions are a pattern matching standard for string parsing and replacement and is a way for a computer user to express how a computer program should look for a specified pattern in text and then what the program is to do when each pattern match is found. Sometimes it is abbreviated "regex". They are a powerful way to find and replace strings that take a defined format.

Here is a simple code example in C# that shows how to use regular expressions.

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
06.
07. namespace RegularExpression1                                                    Ask Question
08. {
09.     class Program
10.     {
11.         static void Main(string[] args)
12.         {
13.             Regex r = new Regex(@"^\+?\d{0,2}\-?\d{4,5}\-?\d{5,6}");
14.             //class Regex Repesents an immutable regular expression.
15.
16.             string[] str = { "+91-9678967101", "9678967101", "+91-9678-967101", "+91-96789-67101", "+919678967101" };
17.             //Input strings for Match valid mobile number.
18.             foreach (string s in str)
19.             {
20.                 Console.WriteLine("{0} {1} a valid mobile number.", s,
21.                 r.IsMatch(s) ? "is" : "is not");
22.                 //The IsMatch method is used to validate a string or
23.                 //to ensure that a string conforms to a particular pattern.
24.             }
25.         }
26.     }
27. }
```

Here is a detailed explaination of Regular Expressions and how to use them in C# and .NET:
Regular Expressions in C#

4. Regex for Email Validation in C#

For validating multiple emails, we can use the following regular expressions. We are separating emails by using delimiter ';'

^((\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*)\s*[;]{0,1}\s*)+$

If you want to use delimiter ',' then use this

^((\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*)\s*[,]{0,1}\s*)+$

and if you want to use both delimiter ',' and ';' then use this

^((\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*)\s*[;,.]{0,1}\s*)+$

Learn more here: Regex for Multiple Email Validation

5. Validating User Input With Regular Expressions in C#

This article explains how to use Regular expressions (the Regx class of the System.Text.RegularExpressions namespace) in C# and .NET.

We can use Regex.Match method that takes an input and regex and returns success if

```
if (!Regex.Match(firstNameTextBox.Text, "^[A-Z][a-zA-Z]*$").Success) {}
    if (!Regex.Match(addressTextBox.Text, @"^[0-9]+\s+([a-zA-Z]+|[a-zA-Z]+\s[a-zA-Z]+)$").Success)

if (!Regex.Match(cityTextBox.Text, @"^([a-zA-Z]+|[a-zA-Z]+\s[a-zA-Z]+)$").Success)
     if (!Regex.Match(stateTextBox.Text, @"^([a-zA-Z]+|[a-zA-Z]+\s[a-zA-Z]+)$").Success)
```

```
if (!Regex.Match(zipCodeTextBox.Text, @"^\d{5}$").Success)
{
    if (!Regex.Match(phoneTextBox.Text, @"^[1-9]\d{2}-[1-9]\d{2}-\d{4}$").Success)
```

Learn more here:

Validating User Input with Regular Expressions

6. Split String using Regex.split (Regular Expression) in C#

In this post, we will learn about how to split the string using Regex in c#.

In this post, we will learn how to split the string using RegEx in C#. Regex splits the string based on a pattern. It handles a delimiter specified as a pattern. This is why Regex is better than string.Split. Here are some examples of how to split a string using Regex in C#. Let's start coding.

For use, Regex adds the below namespaces for spliting a string.

```
using System;
using System.Text.RegularExpressions;
using System.Collections.Generic;
```

Example 1

```
string Text = "1 One, 2 Two, 3 Three is good.";

string[] digits = Regex.Split(Text, @"\D+");

foreach (string value in digits)
{
    int number;
    if (int.TryParse(value, out number))
    {
        Console.WriteLine(value);
    }
}
```

The above code splits the string using \D+ and loops through check number and print.

Learn more here:

Split String using Regex in C#

7. Replace Special Characters from string using Regex in C#

In this blog I'll tell you about How to replace Special Characters Using Regex in C#.

If you are having a string with special characters and want's to remove/replace them then you can use regex for that.

Use this code:

```
Regex.Replace(your String, @"[^0-9a-zA-Z]+", "")
```

This code will remove all of the special characters but if you doesn't want to remove some of the special character for e.g. comma "," and colon ":" then make changes like this:

```
Regex.Replace(Your String, @"[^0-9a-zA-Z:,]+", "")
```

Similarly you can make changes according to your requirement.

Note:

Ask Question

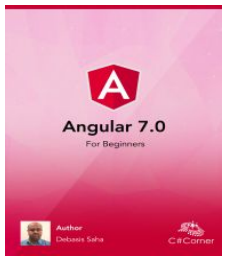
Further readings and References

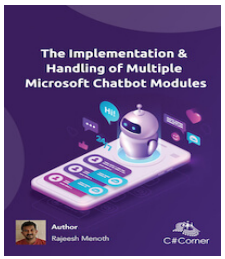
If you're new to regular expressions, I recommend reading this article Introduction to Regular Expressions.

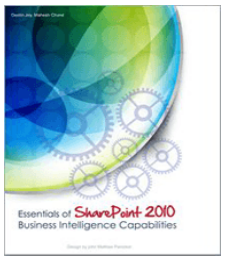
Here is one more article: Regular Expressions Usage in C#

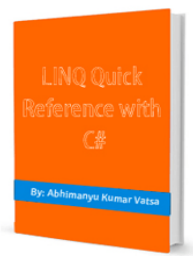
C# Regex | CSharp Regex | Regular Expressions

OUR BOOKS









C# Curator TOP 100

This is a C# Corner community account used by curators.

https://www.c-sharpcorner.com/members/puran-mehra

60	43.5m	1
22	2	

Type your comment here and press Enter Key (Minimum 10 characters)

2143 15 0

Ask Question

0 0 Reply

Nice useful article Sir.

Sourav Kumar Das

761 2.5k 147.8k

Nov 23, 2019

0 0 Reply

FEATURED ARTICLES

F# Records

How to Use Change Tokens In .NET 7

Microsoft Teams Incoming Webhook Integration With ASP.NET Core

Static Abstract Interface Members In C# 11 And Curiously Recurring Template Pattern

Learn C# 8.0

CHALLENGE YOURSELF