1. C# Regex Class

In C#, the Regex class provides methods and properties to work with regular expressions. Here are some key features and functionalities of C# regex:

* Pattern matching: C# regex allows you to define patterns using special characters and constructs to match specific sequences of characters within a string.
* Searching and replacing: You can use regex to search for patterns within a string and replace them with other text or manipulate the matched portions of the string.
* Validation: Regex is commonly used for validating user input. You can define a pattern that input must match, and then check if the input string conforms to that pattern.
* Splitting strings: Regex can be used to split strings based on a specified pattern. For example, using regex, you can split a comma-separated list of values into individual elements.
* Capturing groups: Regex allows you to define capturing groups, which are portions of the pattern enclosed in parentheses. These groups can be used to extract specific parts of a matched string.

The 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 the pattern is to match a word starting with char ‘M’.

// Create a pattern for a word that starts with the letter "M"

string pattern = @"\b[M]\w+";

// Create a Regex

Regex rg = new Regex(pattern);

C#

Copy

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

// Long string

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

C#

Copy

The Matches method finds all the matches in a Regex and returns a MatchCollection.

// Get all matches

MatchCollection matchedAuthors = rg.Matches(authors);

C#

Copy

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);

C#

Copy

Here is the complete code:

// Create a pattern for a word that starts with the 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);

C#

Copy

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 ensure 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);

C#

Copy

2. Replacing multiple white spaces using Regex

The Regex.Replace() method replaces a matched string with a new one. The following example finds multiple whitespaces in a string and replaces them with a single whitespace.

// A long string with a 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}");

C#

Copy

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

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

C#

Copy

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(", ");

}

C#

Copy

Regular Expressions

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

using System.Text;

using System.Text.RegularExpressions;

namespace RegularExpression1

{

class Program

{

static void Main(string[] args)

{

Regex r = new Regex(@"^\+?\d{0,2}\-?\d{4,5}\-?\d{5,6}");

//class Regex represents an immutable regular expression.

string[] str = { "+91-9678967101", "9678967101", "+91-9678-967101", "+91-96789-67101", "+919678967101" };

//Input strings for Match valid mobile number.

foreach (string s in str)

{

Console.WriteLine("{0} {1} a valid mobile number.", s,

r.IsMatch(s) ? "is" : "is not");

//The IsMatch method is used to validate a string or

//to ensure that a string conforms to a particular pattern.

}

}

}

}

C#

Copy

Here is a detailed tutorial on Regular Expressions and how to use them in C# and .NET:  [Regular Expressions In C#](https://www.c-sharpcorner.com/article/regular-expressions-in-C-Sharp/).

4. Email Validation in C# with Regex

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

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

So by using the above regular expression, you can validate a single email as well as multiple emails.

Learn more here: [Regex for Multiple Email Validation](https://www.c-sharpcorner.com/blogs/regex-for-multiple-email-validation1).

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#.

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)

C#

Copy

if (!Regex.Match(zipCodeTextBox.Text, @"^\d{5}$").Success)

{

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

C#

Copy

Learn more here: [Validating User Input with Regular Expressions](https://www.c-sharpcorner.com/UploadFile/87b416/validating-user-input-with-regular-expressions/)

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

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

Here 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 splitting a string.

using System;

using System.Text.RegularExpressions;

using System.Collections.Generic;

C#

Copy

**Example 1**

Split digits from a string using Regex.

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);

}

}

C#

Copy

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

Learn more here: [Split String using Regex in C#](https://www.c-sharpcorner.com/blogs/split-string-using-regexsplit-regular-expression-in-c-sharp)

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

Learn how to replace Special Characters Using Regex in C#.

You can use regex if you have a string with special characters and want to remove/replace them.

Use this code:

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

C#

Copy

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

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

C#

Copy

Similarly, you can make changes according to your requirement.

**Note:**

Please note that regular expressions do not solve every tiny string parsing. If you need simple parsing provided by the String class or other classes, try to use those.