c# regex matches example

52

8

Am trying to get values using following text, any thoughts this can be done with Regex?

Input

> Lorem ipsum dolor sit %download%#456 amet, consectetur adipiscing %download%#3434 elit. Duis non nunc nec mauris feugiat porttitor. Sed tincidunt blandit dui a viverra%download%#298. Aenean dapibus nisl %download%#893434 id nibh auctor vel tempor velit blandit.

Output

```
456
3434
298
893434
```

Thanks in advance.

c# regex

edited Jan 10 '18 at 8:37
nkr
2,779 5 26 36

asked Jan 19 '11 at 21:29
Sha Le
438 2 7 10

6 Answers

59

So you're trying to grab numeric values that are preceded by the token "%download%#"?

Try this pattern:

That should work. I don't think `#` or `%` are special characters in .NET Regex, but you'll have to either escape the backslash like `\\` or use a verbatim string for the whole pattern:

```
var regex = new Regex(@"(?<=%download%#)\d+");
return regex.Matches(strInput);
```

Tested here: http://rextester.com/BLYCC16700

NOTE: The lookbehind assertion `(?<=...)` is important because you don't want to include `%download%#` in your results, only the numbers after it. However, your example appears to require it before each string you want to capture. The lookbehind group will make sure it's there in the input string, but won't include it in the returned results. More on lookaround assertions here.

edited Sep 9 '13 at 15:19

answered Jan 19 '11 at 21:32

Justin Morgan
23.3k 11 64 95

Perfect. You have an extra \ in the code snippet. Just curious, could you please explain me or direct to me some article how all these works. Thanks. – Sha Le Jan 19 '11 at 21:46

1 The extra '\' is intentional. When you pass the pattern around in the C# code, the doubled '\' lets the CLR know that the '\' is part of the regex pattern, not a special character in the C# string such as '\n' or '\t' (hope that makes sense). For an excellent regex reference and tutorial, check out regular-expressions.info . – Justin Morgan Jan 19 '11 at 21:52

1 totally un-needed lookahead, use named groups! – Firoso Jan 19 '11 at 21:52

1 syntactic clarity and extensibility, especially for larger expressions. Especially if you use strongly named groups and use resources for group names. – Firoso Jan 19 '11 at 22:06

2 To get rid of backquoting, prefix your string with an @, like so: Regex regex = new Regex(@"(?<=%download%#)\d+"); – ashes999 Jan 19 '11 at 23:08

38

All the other responses I see are fine, but C# has support for named groups!

I'd use the following code:

```
const string input = "Lorem ipsum dolor sit %download%#456 amet, consectetur adipiscing
%download%#3434 elit. Duis non nunc nec mauris feugiat porttitor. Sed tincidunt blandit
dui a viverra%download%#298. Aenean dapibus nisl %download%#893434 id nibh auctor vel
```

```
{
    Regex expression = new Regex(@"%download%#(?<Identifier>[0-9]*)");
    var results = expression.Matches(input);
    foreach (Match match in results)
    {
        Console.WriteLine(match.Groups["Identifier"].Value);
    }
}
```

The code that reads: `(?<Identifier>[0-9]*)` specifies that `[0-9]*` 's results will be part of a named group that we index as above: `match.Groups["Identifier"].Value`

edited Jan 10 '18 at 8:39
nkr
2,779 5 26 36

answered Jan 19 '11 at 21:50
Firoso
3,871 9 39 86

1 No need to name them, just use @"%download%#(\d*)" and you'll have the result in match.Groups[1].Value – Perdi Estaquel Oct 16 '18 at 22:40

4

```
public void match2()
{
    string input = "%download%#893434";
    Regex word = new Regex(@"\d+");
    Match m = word.Match(input);
    Console.WriteLine(m.Value);
}
```

answered Aug 31 '13 at 5:43
mohan
41 1

2

It looks like most of post here described what you need here. However - something you might need more complex behavior - depending on what you're parsing. In your case it might be so that you won't need more complex parsing - but it depends what information you're extracting.

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

public class Info
{
    public String Identifier;
    public char nextChar;
};

class testRegex {

    const string input = "Lorem ipsum dolor sit %download%#456 amet, consectetur
adipiscing %download%#3434 elit. " +
    "Duis non nunc nec mauris feugiat porttitor. Sed tincidunt blandit dui a
viverra%download%#298. Aenean dapibus nisl %download%#893434 id nibh auctor vel tempor
velit blandit.";

    static void Main(string[] args)
    {
        Regex regex = new Regex(@"%download%#(?<Identifier>[0-9]*)(?<nextChar>.)(?
<thisCharIsNotNeeded>.)");
        List<Info> infos = new List<Info>();

        foreach (Match match in regex.Matches(input))
        {
            Info info = new Info();
            for( int i = 1; i < regex.GetGroupNames().Length; i++ )
            {
                String groupName = regex.GetGroupNames()[i];

                FieldInfo fi = info.GetType().GetField(regex.GetGroupNames()[i]);

                if( fi != null ) // Field is non-public or does not exists.
                    fi.SetValue( info, Convert.ChangeType(
match.Groups[groupName].Value, fi.FieldType));
            }
            infos.Add(info);
        }

        foreach ( var info in infos )
        {
            Console.WriteLine(info.Identifier + " followed by '" +
```

```
};
```

This mechanism uses C# reflection to set value to class. group name is matched against field name in class instance. Please note that Convert.ChangeType won't accept any kind of garbage.

If you want to add tracking of line / column - you can add extra Regex split for lines, but in order to keep for loop intact - all match patterns must have named groups. (Otherwise column index will be calculated incorrectly)

This will results in following output:

```
456 followed by ' '
3434 followed by ' '
298 followed by '.'
893434 followed by ' '
```

answered Nov 21 '14 at 11:36

TarmoPikaro
2,111 1 19 34

0

This pattern should work:

```
#\d

foreach(var match in System.Text.RegularExpressions.RegEx.Matches(input, "#\d"))
{
    Console.WriteLine(match.Value);
}
```

(I'm not in front of Visual Studio, but even if that doesn't compile as-is, it should be close enough to tweak into something that works).

answered Jan 19 '11 at 21:31

Adam Robinson
152k 27 256 323

0

```
Regex regex = new Regex("%download#(\\d+?)%", RegexOptions.SingleLine);
Matches m = regex.Matches(input);
```

I think will do the trick (not tested).

edited Jan 19 '11 at 21:47

answered Jan 19 '11 at 21:35

signetro

676 5 15