

Re: RegEx problem

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2007-06/msg04573.html>

- *From:* Jesse Houwing <jesse.houwing@xxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Thu, 28 Jun 2007 21:23:11 +0200
-

* Martin# wrote, On 28-6-2007 18:40:

Hello,

First, very good and detailed answer! (Got a positive rate from me)

Thank you :)

But I would prefer the string.Split solution that you also presented.
A quick test with a loop and two timestamps will show you why!

I hadn't tested, but my guess is that it's a major difference. Regex can do beautiful things, but isn't the best tool for every problem. As I said before: I'd prefer this solution over the regex one. It's both easier to read, and faster. The only problem is that it doesn't validate the input while the regex would do that for you.

I'm not sure if a int.TryParse would impact the loop you tried enough to make it slower than a regex though, my guess is that it's still faster than a regex.

Jesse

All the best,

and to you.

Jesse

Martin

"Jesse Houwing" wrote:

Re: RegEx problem

* jac wrote, On 28-6-2007 17:26:

Hi,

I have problems with following code and don't find the bug :

```
// Set [8,9,54]
ArrayList aArray = new ArrayList();
regStr = new Regex(@"(?:\d+[,])*(\d+)");
if(text != null && regStr.IsMatch(text))
{
    Match m = regStr.Match(text);
    GroupCollection groups = m.Groups;
    number = 0;
    for(int i=1;i < groups.Count;i++)
    {
        foreach(Capture c in groups[i].Captures)
        {
            aArray.Add(c.Value.ToString());
            number++;
        }
    }
}
```

[8,9] : thats working in my aArray I have 8 and 9

[16,5] : OK I have 16 and 5

[16,34] : That is nok I have 3 items in my array 16 and 3 and 4

[16] : that s is nok I have 2 items in my array 1 and 6

Why m.groups has 3 groups for [16,34]? The same for [16] why m.groups has 2 groups.

I think it must be the last part of my regex expression (\d+). This is one group even if there are more numbers in it. How can I solve this?

Thanks in advance,
jac

```
\[(?<number>\d+)(?:,(?<number>\d+))*\]
```

should do the trick. Currently there are too many options as both the , as well as the whole first group are optional (which they're not).

The new expression reads

Re: RegEx problem

find a [
find a number (one or more digits)
optionally find a comma followed by a number
repeat optional group if possible
find a]

both number are captured in the same named group, which makes it easier to extract the values:

```
Match m = regStr.Match(text);  
foreach (Capture c in m.Groups["number"].Captures)  
{  
    aArray.Add(c.Value);  
}
```

```
number = aArray.Count;
```

Optionally you could also do a `string.Split` with '[' , ',' and ']' as separator characters which would probably be faster as well. You can instruct `string.Split` to ignore empty groups.

```
string[] results = "[16,23,1]".Split(new char[] { ',', '[', ']' },  
StringSplitOptions.RemoveEmptyEntries);  
int number = results.Length;
```

I'd prefer this solution over the regex one.

Jesse