

Re: Splitting a string with Regex and keep the separator

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2007-06/msg00855.html>

- *From:* Jesse Houwing <jesse.houwing@xxxxxxxxxxxxxxxxxxxx>
 - *Date:* Wed, 06 Jun 2007 00:28:29 +0200
-

* nagar@xxxxxxxxxxxxxxxxxxxx wrote, On 5-6-2007 22:06:

One more thing Jesse.
I noticed that the Key(val) is not interpreted correctly if I have two expressions attached to one another.

For example

test#Key(F1) is interpreted correctly
test#Key(F1)#Key(F2) is not

How should I change the expression?
Thanks again.
Andrea

After some careful testing I got the other issues fixed as well. The regex is quite big already. I'll try to explain what's going on where.

First, the regex:

```
\G((?<other>((?!#[^\(]+([\^])+)|.)+)(?<keyval>#(?<key>\w+)\(((?<token>(?)\w+)\s*)+))
```

To extract the fields you can use this:

```
foreach (Match m in ms)
{
if (m.Groups["keyval"].Success)
{
string key = m.Groups["key"].Value;
foreach (Capture c in m.Groups["token"].Captures)
{
string token = c.Value;
}
}
else if (m.Groups["other"].Success)
```

Re: Splitting a string with Regex and keep the separator

```
{  
ManipulateOther(m.Groups["other"].Value);  
}  
}
```

And now for the explanation:

`\G` -> Make sure every new match is directly adjacent to the previous one, so we're not skipping invalid input

`(?<other>)` -> Match the 'other' text into a group named "other"

`((?!#[^\(]+\([^\)]+\)).)+` Match every character that isn't the start of a key/val pair. I'm doing this by looking ahead to see if a keyval structure is found, and if it isn't I add one character to the match (.).

If we're at the end of an "other section" there's two options, either the end of the string, in which case the regex just stops matching, or there's the start of a key/val thingy.

`(?<keyval>)` -> match the whole key/val structure into a group named "keyval"

`#(?<key>\w+)\(` -> match the key and put it in a group named "key". The key comes directly after a "#" and only contains one or more alphanumeric characters (`\w+`) followed by "("

`((?<token>(?!#\w+)\s*)+)` -> Match every token into a group called "token". If this group captures multiple tokens they're added to the group's Captures collection in the order in which they're found. A token is made up of one or more alphanumeric characters (`\w+`). It can be followed by zero or more spaces. The `(?>...)` construction is used to prevent too much backtracking going on. The whole token-followed-by-space can exist multiple times. As the final token will not have a space behind it I used `\s*`.

`)` -> and finally the closing parenthesis.

Keep in mind that if you use the `RegexOptions.IgnorePatternWhitespace`, you can reflow the regex to be easier to read. It's also easier to add comments that way.

```
@"  
(?# Start of the previous match)  
\G  
(  
(?#  
Match any character until you find the start of  
A key/val pair.  
)  
(?<other>((?!#[^\(]+\([^\)]+\)).)+)  
|  
(?#  
Match a key/val pair. Put the keyname in a group  
and every token in another.  
)  
(?<keyval>#(?<key>\w+)\(((?<token>\w+)\s*)+\))  
)  
";
```

One alternative to this whole approach I haven't tested yet, but would work none the less is to only look for

Re: Splitting a string with Regex and keep the separator

the special key/val thingies with only the right subexpression:

```
#(?:<key>\w+)\(((?<token>(?\w+)\s*)+\))
```

And query the start/end location of each match to determine if there were any other characters since the last found match. You can then extract those characters with a substring function. I'm not sure which option is faster, but I would not be surprised if the substring option would work even better, though it would contain more coding.

Jesse Houwing

.