

Re: Expert script (.bat) writers help needed (strip double-quote from string)

Source:

http://www.tech-archive.net/Archive/WinXP/microsoft.public.windowsxp.help_and_support/2004-09/0147.html

From: David Candy (david_at_mvps.org)

Date: 08/31/04

Date: Wed, 1 Sep 2004 04:08:31 +1000

I thought the for command would help but haven't found a way to specify " as a delimiter. And a search of several hundred posts using the delims command didn't find anyone who has tried.

This is one that doesn't work as it assumes no delimiters
for /f "tokens=1,2,3 delims="" %m in ("dog" cat") do echo %m and %n

^" doesn't help.

Therefore,

1. Write a vbs/jscript script (I've attached one that search and replaces files – you can modify it to write the result to the console).

ReplaceRegExp Filename SearchString ReplaceString. Quotes are required for spaces. Use 0xnn for wierd characters. Wildcard list is below..

2. See the set command for extracting sub strings. Something like

```
if "%TempVar:~count,1%" neq "" set TempVar2=tempVar2%TempVar:~Count,1%  
where count is incremented for each character.
```

Visual Basic Scripting Edition

Pattern Property

See Also

Global Property | IgnoreCase Property

Applies To: RegExp Object

Requirements

Version 2

Sets or returns the regular expression pattern being searched for.

object.Pattern [= "searchstring"]Arguments

object

Required. Always a RegExp object variable.

searchstring

Optional. Regular string expression being searched for. May include any of the regular expression characters defined in the table in the Settings section.

Settings

Special characters and sequences are used in writing patterns for regular expressions. The following table describes and gives an example of the characters and sequences that can be used.

Character Description

`\` Marks the next character as either a special character or a literal. For example, `"n"` matches the character `"n"`. `"\n"` matches a newline character. The sequence `"\\"` matches `"\"` and `"\"` matches `"("`.

`^` Matches the beginning of input.

`$` Matches the end of input.

`*` Matches the preceding character zero or more times. For example, `"zo*"` matches either `"z"` or `"zoo"`.

`+` Matches the preceding character one or more times. For example, `"zo+"` matches `"zoo"` but not `"z"`.

`?` Matches the preceding character zero or one time. For example, `"a?ve?"` matches the `"ve"` in `"never"`.

`.` Matches any single character except a newline character.

`(pattern)` Matches pattern and remembers the match. The matched substring can be retrieved from the resulting Matches collection, using `Item [0]...[n]`. To match parentheses characters `()`, use `"\"` or `"\"`.

`x|y` Matches either `x` or `y`. For example, `"z|wood"` matches `"z"` or `"wood"`. `"(z|w)oo"` matches `"zoo"` or `"wood"`.

`{n}` `n` is a nonnegative integer. Matches exactly `n` times. For example, `"o{2}"` does not match the `"o"` in `"Bob,"` but matches the first two `o`'s in `"foooooo"`.

`{n,}` `n` is a nonnegative integer. Matches at least `n` times. For example, `"o{2,}"` does not match the `"o"` in `"Bob"` and matches all the `o`'s in `"foooooo."` `"o{1,}"` is equivalent to `"o+"`. `"o{0,}"` is equivalent to `"o*"`.

`{n,m}` `m` and `n` are nonnegative integers. Matches at least `n` and at most `m` times. For example, `"o{1,3}"` matches the first three `o`'s in `"foooooo."` `"o{0,1}"` is equivalent to `"o?"`.

`[xyz]` A character set. Matches any one of the enclosed characters. For example, `"[abc]"` matches the `"a"` in `"plain"`.

`[^xyz]` A negative character set. Matches any character not enclosed. For example, `"[^abc]"` matches the `"p"` in `"plain"`.

`[a-z]` A range of characters. Matches any character in the specified range. For example, `"[a-z]"` matches any lowercase alphabetic character in the range `"a"` through `"z"`.

`[^m-z]` A negative range characters. Matches any character not in the specified range. For example, `"[m-z]"` matches any character not in the range `"m"` through `"z"`.

`\b` Matches a word boundary, that is, the position between a word and a space. For example, `"er\b"` matches the `"er"` in `"never"` but not the `"er"` in `"verb"`.

`\B` Matches a non-word boundary. `"ea*r\B"` matches the `"ear"` in `"never early"`.

`\d` Matches a digit character. Equivalent to `[0-9]`.

`\D` Matches a non-digit character. Equivalent to `[^0-9]`.

`\f` Matches a form-feed character.

`\n` Matches a newline character.

`\r` Matches a carriage return character.

`\s` Matches any white space including space, tab, form-feed, etc. Equivalent to `"[\f\n\r\t\v]"`.

`\S` Matches any nonwhite space character. Equivalent to `"[^ \f\n\r\t\v]"`.

`\t` Matches a tab character.

`\v` Matches a vertical tab character.

`\w` Matches any word character including underscore. Equivalent to `"[A-Za-z0-9_]"`.

`\W` Matches any non-word character. Equivalent to `"[^A-Za-z0-9_]"`.

`\num` Matches num, where num is a positive integer. A reference back to remembered matches. For example, `"(.)\1"` matches two consecutive identical characters.

`\n` Matches n, where n is an octal escape value. Octal escape values must be 1, 2, or 3 digits long. For example, `"\11"` and `"\011"` both match a tab character. `"\0011"` is the equivalent of `"\001"` & `"1"`. Octal escape values must not exceed 256. If they do, only the first two digits comprise the expression. Allows ASCII codes to be used in regular expressions.

`\xn` Matches n, where n is a hexadecimal escape value. Hexadecimal escape values must be exactly two digits long. For example, `"\x41"` matches "A". `"\x041"` is equivalent to `"\x04"` & `"1"`. Allows ASCII codes to be used in regular expressions.

Remarks

The following code illustrates the use of the Pattern property.

Function RegExpTest(patrn, strng)

```
Dim regEx, Match, Matches ' Create variable.
Set regEx = New RegExp ' Create a regular expression.
regEx.Pattern = patrn ' Set pattern.
regEx.IgnoreCase = True ' Set case insensitivity.
regEx.Global = True ' Set global applicability.
Set Matches = regEx.Execute(strng) ' Execute search.
For Each Match in Matches ' Iterate Matches collection.
    RetStr = RetStr & "Match found at position "
    RetStr = RetStr & Match.FirstIndex & ". Match Value is "
    RetStr = RetStr & Match.Value & "." & vbCrLf
Next
RegExpTest = RetStr
End Function
MsgBox(RegExpTest("is.", "IS1 is2 IS3 is4"))
```

See Also

Global Property | IgnoreCase Property

Applies To: RegExp Object

© 2001 Microsoft Corporation. All rights reserved.

Build: Topic Version 5.6.9309.1546

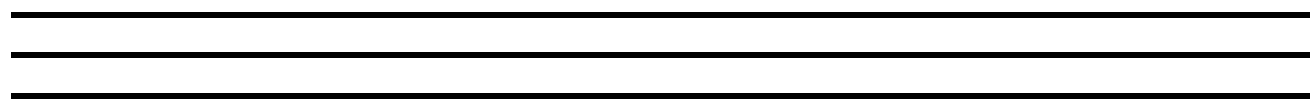
--

<http://www.newmatilda.com/home/default.asp>

```
"James" <anonymous@discussions.microsoft.com> wrote in message news:32e101c48f1b$26b3c0e0$a601280
> Unfortunately, this only removes "surrounding" double-
> quotes. Ie, if string were to contain quotes within the
> text ("Hello "cruel" world"), only the outside quotes are
> removed with the inclusion of a ~
>
> :( Nice try thou
>
> >-----Original Message-----
```

microsoft.public.windowsxp.help_and_support: Re: Expert script (.bat) writers help needed (strip double-quote from str

```
> >The same way you did I - %%~J and %%~K.  
> >  
> >--  
> >-----  
> >'Not happy John! Defending our democracy',  
> >http://www.smh.com.au/articles/2004/06/29/1088392635123.ht  
> ml  
> >  
>
```



- application/octet-stream attachment: [ReplaceRegExp.vbs](#)