Regular Expressions

Text pattern matching.

Ben Simo

Ben@QualityFrog.com
http://www.QuestioningSoftware.com

Regular Expression Support

Test Tools
- WinRunner
- QuickTest Professional
- LoadRunner

Utilities
- sed
- grep
- egrep

Scripting/Programming Languages
- Perl
- Ruby
- Java
- JavaScript
- C#
- VBScript

Text/Code Editors
- vi
- V-Edit
- Visual Studio

Support in HP/Mercury Tools

WinRunner
- Describe GUI objects
- Static text and edit checkpoints
- Script code

QTP
- Describe GUI objects
- Text checkpoints
- Script code

LoadRunner
- Web GUI (Click & Script) protocol functions
- Terminal Emulator protocol functions

Simplest

"frog"

- Matches "frog", "bullfrog", and "tree frog"; but not "Frog"

"2008"

- Matches "2008", "120083", and "Copyright 2008"; but not "2,008"

Metacharacters: ^

^ Matches the beginning

"^Frog"

- Matches "Froggy went a courting", but not "Quality Frog"

"^2008"

- Matches "20081 Bananas", but not "Copyright 2008"

** The ^ has a different meaning when used in a character class definition.*

** Not supported by WinRunner.*

Metacharacters: $

$ Matches the end

"frog$"

- Matches "frog", "bullfrog", and "tree frog"; but not "froggy" or "The frog sat on a log."

"2008$"

- Matches "Copyright 2008" and "12008"; but not "20081" or "2008."

** Not supported by WinRunner.*

Metacharacters: .

. Matches any single character

“.at”

- Matches “cat”, “rat”, “bat”, “goat”, and “gnat”

“b.t”

- Matches “bat”, “bet”, “but”, and “b t”; but not “boat” or “boot”

“200.”

- Matches “200.”, “2007”, “2008”, and “200A”; but not “200”

Metacharacters: *

* Matches zero or more occurrences of the preceding character

  “20*5”

  - Matches “2005”, “20005”, “2000000000000000000000005”, “25”; but not “2ABC5” or “2006”

  “20.*5”

  - Matches “2005”, “20005”, “205”, “20ABC5”, and “20 times 5”; but not “25” or “2006”

  - *The "*" is greedy.*
    *For example: "25.*5" will match "25 / 5 = 5"*
    instead of just "25 / 5".

Metacharacters: \

\ Escape character. Identifies the following character as an ordinary character.*

"\$2005"

- Matches "$2005", "$20050", and "Total: $2005"

"\$2005$"

- Matches "$2005" and "Total: $2005"; but not "$20050"

"2005\.*"

- Matches "2005." and "2005….."

"20\..*05"

- Matches "20.05" and "20.ABC05"

** In most cases.*

Metacharacter: ?

? Matches zero or one occurrence of the preceding character or regular expression.

"Spee?d"

- Matches "Sped" and "Speed"; but not "Speeed"

"25.*?5"

- Forces otherwise greedy * to match just "25 / 5" instead of the entire text "25 / 5 = 5"

** Not supported by WinRunner.*

Metacharacter: +

+ Matches one or more occurrences of the previous character.

  “20+5”

  - Matches “2005” and “20005”, but not “25”

  “20.+5”

  - Matches “2005” and “2015”, but not “205”

** Not supported by WinRunner.*

Metacharacters: [ ]

[ ] Defines a character class.

"b[aou]t"

- Matches "bat", "bot", and "but"; but not "bet"

"200[5-9]"

- Matches "2005", "2006", "2007", and "2009"; but not "2004"

"[A-Za-z]"

- Matches any letter but no numbers or symbols

[^] Defines an exclusion-based character class.

"[^409A-Z]

- Matches any character except 4, 0 ,9, and upper case letters

Metacharacters: { } *or* \{ \}

\{ \} Matches a specific number or range of instances of the previous character

"A[0-9]\{3\}$"

- Matches "A123", but not "A1234"

"A[0-9]\{2,3\}"

- Matches "A12" and "A123", but not "A1" or "A1234"

** Some implementations treat "{" and "}" as metacharacters. Others treat "{" and "}" as normal characters and "\{" and "\}" as metacharacters.*

** Not supported by WinRunner.*

Metacharacters: ( ) *or* \(\)

\(\) Treats the expression between \(and \) as a group.

"\(the \)\(the \)*"

- Matches one or more instances of the word "the" followed by a space

"\$\([1-9]\),\([0-9]\)"

- Matches "$1,000", "$9,999", and "$2,394"; but not "$10,100" or "$1000"

"\([^,]*\),"

- Matches zero or more non-comma characters followed by a comma

** Some implementations treat "(" and ")" as metacharacters. Others treat "(" and ")" as normal characters and "\(" and "\)" as metacharacters.*

** Not supported by WinRunner.*

Metacharacter: |

| Or. Use to match one of *two* expressions.

"his|her"

- Matches "the dog ate his homework" and "the dog ate her homework"; but not "the dog ate their homework"

"199[0-9]|200[0-9]"

- Matches years 1990 through 2009.

** Not supported by WinRunner.*

Metacharacters: Words

\< Beginning of word

"\<the"

- Matches "the" and "theology", but not "otherwise" or "together"

\> End of word

"art\>"

- Matches "art" and "smart"; but not "artist"

\b Word boundary

"\bclass\b"

- Matches "class"; but not "classified", "declassified", or "subclass"

\B Not a word boundary. The opposite of \b.

"\Bclass\B"

- Matches "declassified"; but not "class", "classified", or "subclass"

** Not supported by WinRunner, QuickTest Professional, or LoadRunner.*

Metacharacters

^	Matches the beginning
$	Matches the end
.	**Matches any single character**
*****	**Matches zero or more occurrences of the preceding character**
****	**Escape character.**
?	Matches zero or one occurrence of the preceding character
+	Matches one or more occurrences of the previous character
[ ]	**Defines a character class**
[^]	**Defines an exclusion-based character class**
\{ \}	Matches a specific number or range of instances of the previous character
\(\)	Treats the expression between \(and \) as a group
| Or.	Use to match one of two expressions
\<	Matches the beginning of a word
\>	Matches the end of a word
\b	Word boundary
\B	Not a word boundary

Examples: Varying Spaces

"Froggy went a courting"

- Matches "Froggy went a courting, he did ride"

"Froggy *went *a *courting"

- Matches "Froggy went a courting" and "Froggywentacourting"

"Froggy *went *a *courting"

- Matches "Froggy went a courting" and "Froggy went a courting", but not "Froggywentacourting"

Examples: Number Format

"[0-9][0-9]*\.[0-9][0-9][^0-9]"

- Matches "1.29", "1.29%", and "1234.55"; but not "1.299" or ".29"

"[0-9][0-9]*\.[0-9][0-9][^0-9%]"

- Matches "1.29" and "1234.55"; but not "1.29%", "1.299", or ".29"

"0x[0-9A-F][0-9A-F] "

- Matches "0x01 " and "0xFF ", but not "0x000" or "0xAG"

Examples: Text Validation

"risk_profile.*1\.gif"

- Matches any text that contains "risk_profile" before "1.gif".

"Copyright . 200[78] Quality Frog\. All rights reserved\."

- Matches "Copyright © 2007 Quality Frog. All rights reserved." and "Copyright © 2008 Quality Frog. All rights reserved."

"Showing: [0-9][0-9]*-[0-9][0-9]* of [0-9][0-9].* records"

- Matches "Showing: 1-10 of 100 records" and "Showing: 0-0 of 100 records"; but not "Showing: – of records" or "Showing: 1- of 1 records".

"A[LKRSZ|C[AOT]|D[CE]|F[LM]|G[AU]|HI|I[ADLN]|K[SY]|LA|M[AFRHINOPST]|N[CDEHJMVY]|O[HKR]|P[ARW]|RI|S[CD]|T[NX]|UT|V[AIT]|W[AIVY]"

- Matches any valid 2-letter US postal state or territory name abbreviation.

Examples: vi

- `:%s/ */ /g`
  - Change 1 or more spaces into a single space
- `:%s/ *$//`
  - Remove all spaces from the end of the line
- `:%s/^/ /`
  - Insert a space at the beginning of every line
- `:%s/[Ss]&[Pp]/Standard & Poor's/g`
  - Replace all occurrences of text “S&P” (regardless of case) with “Standard & Poor’s”
- :%s/\([^,]*\), \(.*\)/\2 \1/
  - Changes the format of a list of names in last-comma-space-first format to first-space-last format. E.g.,: “Doe, John” becomes “John Doe”

Examples: sed

- `sed 's/^$/d' funds.txt`
  - Displays all non-empty lines from file funds.txt
- `sed 's/^[ \t]*$/d' funds.txt > funds2.txt`
  - Copy all lines that do not contain only whitespace (spaces and tabs) from file funds.txt to funds2.txt
- `sed 's/"//g' funds.txt`
  - Displays file funds.txt with all quotation marks removed.
- `sed 's/Copyright 200[0-7]/Copyright 2008/g' funds.txt > funds2007.txt`
  - Copies funds.txt to funds2007.txt: updating all copyright statements for years 2000 through 2007 with "2008".
- `sed 's/^$/d;s/"//g' funds.txt`
  - Displays all non-empty lines from file funds.txt with quotation marks removed.

Examples: grep and egrep

- `grep "^12:[4-5][0-9]:[0-5][0-9]" log.txt`
  - Display all lines of log.txt that start with a timestamp for times between 12:40:00 and 12:59:59.
- `egrep "([Ee]xception|[Ee]rror)" log.txt`
  - Display all lines of log.txt that contain “Exception”, “exception”, “Error”, or “error”.
- `egrep "[Ee]rror|^ |\*$" log.txt`
  - Display all lines of log.txt that contain “Error” or “error”, or start with a space, or end with an asterisk.
- `tail -f log.txt | egrep "[Ee]xception|^[ \t][ \t]*[^ \n\t]"`
  - Display new lines added to log.txt that contain “Exception”, or “exception”, or start with one or more space or tab characters followed by a character that is not a space or tab.

Ben Simo

Ben@QualityFrog.com

http://www.QuestioningSoftware.com