

Basic Regex patterns

Notebook: Managed Services

Created: 12/20/2017 11:01 AM

Updated: 12/20/2017 2:24 PM

Author: tuan.hoang@episerver.com

Tags: Regex

URL: <https://reaex101.com/>

Basic Regex patterns

- Regex online-testing tool: <https://regex101.com/>

- References:

<http://www.rexegg.com/>

<http://www.rexegg.com/regex-quickstart.html>

<http://www.zytrax.com/tech/web/regex.htm>

<https://www.cheatography.com/davechild/cheat-sheets/regular-expressions/>

Note: Some regex patterns may not work in all environments

A. Basic Patterns

+) Word boundary match

Pattern: `\b`

REGULAR EXPRESSION 4 matches, 23 steps (~0ms)

`/ \b` / g

TEST STRING SWITCH TO UNIT TESTS >

Tuan Hoang

+) Non-word boundary match

Pattern: `\B`

REGULAR EXPRESSION 7 matches, 32 steps (~0ms)

`/ \B` / g

TEST STRING SWITCH TO UNIT TESTS >

Tuan Hoang

+) Word character match

Pattern: `\w`

REGULAR EXPRESSION 12 matches, 25 steps (~0ms)

`/ \w` / g

TEST STRING SWITCH TO UNIT TESTS ▶

TuanHoang123

+) Non-word character match => matches any non-word character (equal to `[^a-zA-Z0-9_]`)

Pattern: `\W`

REGULAR EXPRESSION 1 match, 12 steps (~0ms)

`/ \W` / g

TEST STRING SWITCH TO UNIT TESTS ▶

Tuan Hoang

+) Digit match

Pattern: `\d`

REGULAR EXPRESSION 3 matches, 16 steps (~0ms)

`/ \d` / g

TEST STRING SWITCH TO UNIT TESTS ▶

TuanHoang123

+) Non-digit match

Pattern: `\D`

REGULAR EXPRESSION 9 matches, 22 steps (~0ms)

`/ \D` / g

TEST STRING SWITCH TO UNIT TESTS ▶

TuanHoang123

+) White-space match

Pattern: `\s`

REGULAR EXPRESSION 3 matches, 31 steps (~0ms)

`/ \s` / g

TEST STRING SWITCH TO UNIT TESTS ▶

TuanHoang123 Hoang Anh Tuan

+) Non-whitespace match

Pattern: \S

REGULAR EXPRESSION

24 matches, 52 steps (~0ms)

/ \S

/ g

TEST STRING

SWITCH TO UNIT TESTS

TuanHoang123 Hoang Anh Tuan

+) Quantifiers

Quantifiers

*	0 or more	{3}	Exactly 3
+	1 or more	{3,}	3 or more
?	0 or 1	{3,5}	3, 4 or 5

Add a ? to a quantifier to make it ungreedy.

+) Groups and ranges

Groups and Ranges

`.` Any character except new line (`\n`)

`(a|b)` a or b

`(...)` Group

`(?:...)` Passive (non-capturing) group

`[abc]` Range (a or b or c)

`[^abc]` Not (a or b or c)

`[a-q]` Lower case letter from a to q

`[A-Q]` Upper case letter from A to Q

`[0-7]` Digit from 0 to 7

`\x` Group/subpattern number "x"

Ranges are inclusive.

+) Anchors

Anchors

`^` Start of string, or start of line in multi-line pattern

`\A` Start of string

`$` End of string, or end of line in multi-line pattern

`\Z` End of string

`\b` Word boundary

`\B` Not word boundary

`\<` Start of word

`\>` End of word

+1 Modifiers

Pattern Modifiers

`g` Global match

`i *` Case-insensitive

`m *` Multiple lines

`s *` Treat string as single line

`x *` Allow comments and whitespace in pattern

`e *` Evaluate replacement

`U *` Ungreedy pattern

`*` PCRE modifier

B. Advanced Patterns

1. Named capturing group

Pattern: Hoang (?<name>Tuan) \k<name>

Matched string: Hoang Tuan Tuan

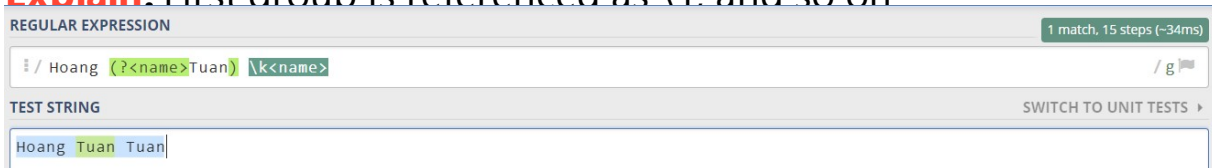
Explain: This pattern will capture the string inside (?<name>...) and hold it in the "name" for back-reference. Later, we can reference to it by \k<name>

2. No-name capturing group

Pattern: (Hoang) (Tuan) \2 \1

Matched string: Hoang Tuan Tuan Hoang

Explain: First group is referenced as \1, and so on



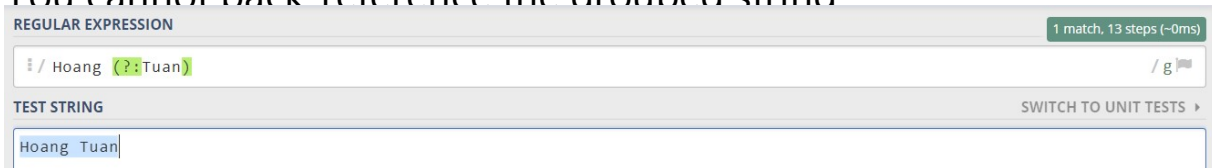
The screenshot shows a regex testing interface. The 'REGULAR EXPRESSION' field contains the pattern `/ Hoang (?<name>Tuan) \k<name>`. The 'TEST STRING' field contains `Hoang Tuan Tuan`. The interface indicates '1 match, 15 steps (~34ms)'. The match is highlighted in the test string, showing 'Hoang' in blue, 'Tuan' in green, and 'Tuan' in blue.

3. Non-capturing group

Pattern: Hoang (?:Tuan)

Matched string: Hoang Tuan

You cannot back-reference the grouped string



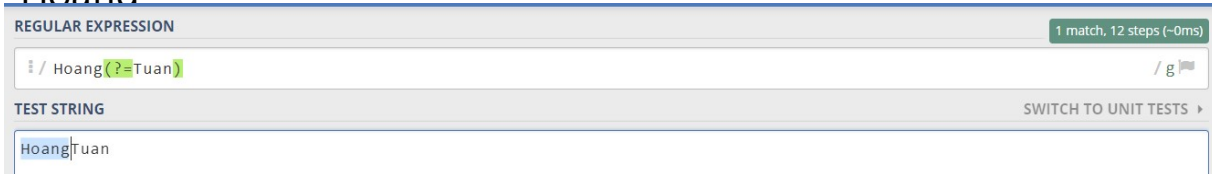
The screenshot shows a regex testing interface. The 'REGULAR EXPRESSION' field contains the pattern `/ Hoang (?:Tuan)`. The 'TEST STRING' field contains `Hoang Tuan`. The interface indicates '1 match, 13 steps (~0ms)'. The match is highlighted in the test string, showing 'Hoang' in blue and 'Tuan' in blue.

4. Positive Lookahead

Pattern: Hoang(?=Tuan)

This pattern will match the string "Hoang" in "HoangTuan" but does not match the string "Hoang" in "HoangHa" & "TuanHoang"

Explain: When found "Hoang", it will look ahead to see if the string "Tuan" exist or not. If "Tuan" exists ahead of "Hoang", it will match "Hoang"

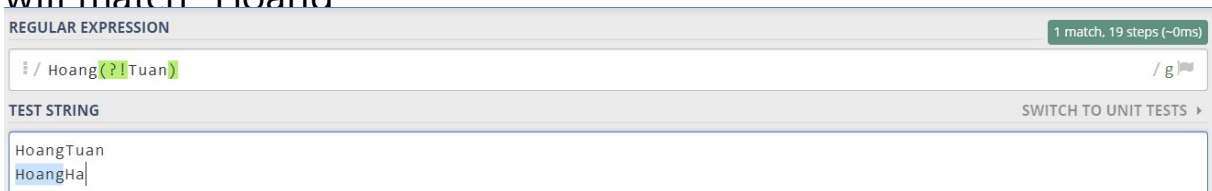


5. Negative Lookahead

Pattern: `Hoang(?!Tuan)`

This pattern will match the string "Hoang" in "HoangHa" but does not match the string "Hoang" in "HoangTuan" & "TuanHoang"

Explain: When found "Hoang", it will look ahead to see if the string "Tuan" exist or not. If "Tuan" DOES NOT exist ahead of "Hoang", it will match "Hoang"



6. Positive Lookbehind

Pattern: `(?<=Tuan)Hoang`

This pattern will match the string "Hoang" in "TuanHoang" but does not match the string "Hoang" in "HaHoang" & "HoangTuan"

Explain: When found "Hoang", it will look behind to see if the string "Tuan" exist or not. If "Tuan" exists behind of "Hoang", it will match "Hoang"

REGULAR EXPRESSION 1 match, 22 steps (~0ms)

/ (?<=Tuan)Hoang /g

TEST STRING SWITCH TO UNIT TESTS ▶

```
TuanHoang
HoangHa
HaHoang
HoangTuan
```

7. Positive Lookbehind

Pattern: (?<!Tuan)Hoang

This pattern will match the string "Hoang" in "HaHoang" but does not match the string "Hoang" in "TuanHoang" & "HoangTuan"

Explain: When found "Hoang", it will look behind to see if the string "Tuan" exist or not. If "Tuan" DOES NOT exist behind of "Hoang", it will match "Hoang"

REGULAR EXPRESSION 3 matches, 38 steps (~0ms)

/ (?<!Tuan)Hoang /g

TEST STRING SWITCH TO UNIT TESTS ▶

```
TuanHoang
HoangHa
HaHoang
HoangTuan
```

8. Greedy & Lazy quantifiers

By default, all quantifiers are greedy, they will try to match as much as possible

To make a quantifier to be lazy (i.e match as few as possible), put ? next to the quantifier

Example:

Greedy version (.+)

REGULAR EXPRESSION 1 match (~1ms)

/.+/

TEST STRING SWITCH TO UNIT TESTS ▶

```
TuanHoang
HoangTuan
```

Lazy version (.+?)

REGULAR EXPRESSION

1 match (~0ms)

~.+?

TEST STRING

SWITCH TO UNIT TESTS ▶

TuanHoang
HoangTuan