

Regular Expression

[Regex](#), [RegExp](#), [Regular Expression](#), [Tips](#), [Java](#)

일부 내용은 [손에 잡히는 정규 표현식](#)에서 발췌

기본 개념

Metacharacters

Metacharacter	설명	비고
.	문자 1개 일치	
+	문자 1개 이상 일치	{1,} 와 같음
*	문자 0개 이상 일치 (문자가 없거나 여러 개 일치)	{0,} 와 같음
?	문자 0개 혹은 1개 일치 (문자가 없거나 1개 일치)	{0,1} 와 같음
Metacharacter	설명	
	OR	
\	Metacharacter를 일반 문자로 인식	

범위

Metacharacter	설명	비고
[]	문자 집합 구성원 중 하나와 일치	AND가 아닌 OR 조건이다!
[^]	문자 집합 구성원을 제외하고 일치	
-	범위 정의	
Metacharacter	설명	비고
\d	10진수 숫자 일치	[0-9] 와 같음
\D	\d와 반대로 일치	[^0-9] 와 같음
\x	16진수 숫자 일치	[0-9a-fA-F] 와 같음
\0	8진수 숫자 일치	
\w	영문, 숫자, _ 일치	[a-zA-Z0-9_] 와 같음
\W	\w와 반대로 일치	[^a-zA-Z0-9_] 와 같음

공백 문자

Metacharacter	설명	
[\b]	Backspace	
Metacharacter	설명	
\f	페이지 넘김(form feed)	
\n	줄바꿈 (new line)	
\r	Carrige Retrun (커서/캐럿을 그 줄의 맨 앞으로 이동)	
\t	탭	
\v	수직 탭	
Metacharacter	설명	비고
\s	모든 공백 문자와 일치	[\f\n\r\t\v] 와 같다

Metacharacter	설명	비고
\S	\s와 반대로 일치	[^\f\n\r\t\v] 와 같다

Quantifier

수량자. {MIN,MAX} (MAX는 생략 가능)

과하게 일치하는 상황 방지

- Greedy Qualifier : 최대로 일치
- Lazy Qualifier : 최소로 일치

Greedy Qualifier	Lazy Qualifier
*	*?
+	+?
{n,}	{n,}?

예시

```
(?i)<b>.*</b>
```

```
<b>ak</b> and <b>hi</b>
```

```
(?i)<b>.*?</b>
```

```
<b>ak</b> and <b>hi</b>
```

위치 지정

Metacharacter	설명	비고
^	전체 문자열의 시작과 일치	일부 언어는 \A. Java, JavaScript는 미지원
\$	전체 문자열의 끝과 일치	일부 언어는 \Z. Java, JavaScript는 미지원
(?m)	다중행 모드. ^와 \$를 행별로 인식	일부 언어는 미지원. Java, Javascript는 지원
Metacharacter	설명	비고
\b	단어 경계와 일치	boundary
\B	\b 반대로 일치	

Lookaround

일치하는 영역을 제외하고, 일치하는 영역 기준으로 이전 혹은 이후 값을 반환

Lookahead

전방 탐색

- Positive : ?=
- Negative : ?!

.+(?=)

<http://www.google.com>

<https://mail.google.com>

<ftp://ftp.xxx.xyz>

Backreference

괄호로 감싼 하위 표현식을 참조하는 정규 표현식. \$1~n 으로 표현. (일부 언어는 \$대신 /을 사용하기도 한다.)

변수와 비슷.

예시

```
String eg1 = "Hello, xxx@xxx.com is my email address.";
String result1 = eg1.replaceAll("(\\w+([\\w.]*@[\\w.]+\\.\\w+)", "<a href=\"\\$1\">\\$1</href>");
System.out.println(result1); // Hello, <a href="ben@forta.com">ben@forta.com</href> is my email address.

String eg2 = "031-123-1234";
String result2 = eg2.replaceAll("(\\d{3})(-)(\\d{3})(-)(\\d{4})", "($1) $3-$5");
System.out.println(result2); // (031) 123-1234
```

Lookbehind

후방 탐색.

- Positive : ?<=
- Negative : ?<!

(?<=\\\$)[0-9.]+

A11 : \$23.34

B22 : \$5.31

Total items found : 2

Java

<https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html>

```

Pattern p = Pattern.compile("^REGULAR_EXPRESSION$",
Pattern.CASE_INSENSITIVE); // regex, pattern
Matcher m = p.matcher(STRING);
boolean b = m.matches();

```

Pattern Constant

Pattern.CASE_INSENSITIVE	(?i)
Pattern.COMMENTS	(?x)
Pattern.MULTILINE	(?m)
Pattern.DOTALL	(?s)
Pattern.LITERAL	None
Pattern.UNICODE_CASE	(?u)
Pattern.UNIX_LINES	(?d)

POSIX character classes (US-ASCII only)

p{Lower}	A lower-case alphabetic character: [a-z]
p{Upper}	An upper-case alphabetic character: [A-Z]
p{ASCII}	All ASCII: [\x00-\x7F]
p{Alpha}	An alphabetic character: [\p{Lower}\p{Upper}]
p{Digit}	A decimal digit: [0-9]
p{Alnum}	An alphanumeric character: [\p{Alpha}\p{Digit}]
p{Punct}	Punctuation: One of !"#\$%&'()*+,-./:;<=>?@[\]^_`{ }~
p{Graph}	A visible character: [\p{Alnum}\p{Punct}]
p{Print}	A printable character: [\p{Graph}\x20]
p{Blank}	A space or a tab: [\t]
p{Cntrl}	A control character: [\x00-\x1F\x7F]
p{XDigit}	A hexadecimal digit: [0-9a-fA-F]
p{Space}	A whitespace character: [\t\n\x0B\f\r]

활용

전화번호

```
^(02|0[3-6]{1}[1-5]{1})-[0-9]{3,4}-?[0-9]{4}$ // 지역번호 - xxx(x) - xxxx  
^(15(44|77|88|99)|1644)-?[0-9]{4}$ // 15xx/1644-xxxx
```

소괄호 안 문자(**Parameter**) 추출

```
String str = "(int a, int b)";  
  
Pattern p = Pattern.compile("\\((.*?)\\)");  
Matcher m = p.matcher(str);  
  
while(m.find())  
    System.out.println(m.group(1));
```

From:

<http://ledyx.xyz/> - **Ledyx WIKI**

Permanent link:

http://ledyx.xyz/doku.php?id=regular_expression&rev=1665925895

Last update: **2022/10/16 13:11**

