Regular Expressions

Pieter van den Hombergh

Fontys Hogeschool voor Techniek en Logistiek

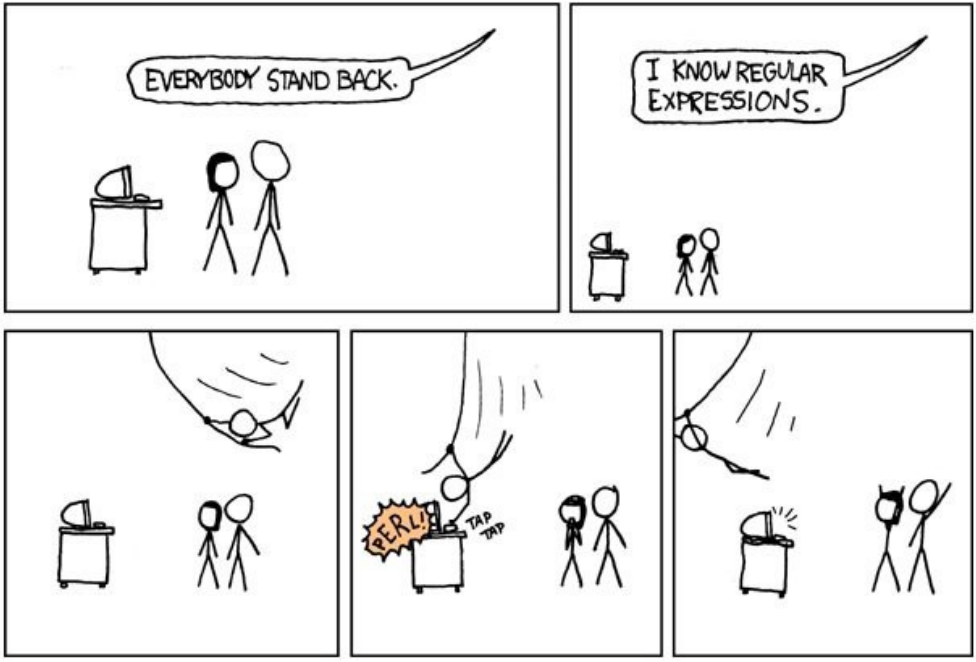
April 13, 2018

Regex?

Regular Expressions are a very power, but also complex tool. There is the saying that:

If you start with a problem that
can be solved with regular expression,
You end up having two problems

having fun with regexes



use case for regexes

Regular Expressions: Meta Characters

- meta characters: `! $ & ( )* + - . : < = > ? [ ] ^ |`
  some regain their literal meaning when preceded by a \ e.g. `\?`.
- A dot (`.`) matches any character
- `*` matches zero or more occurrences of the preceding regexp
- `+` matches one or more occurrences of the preceding regexp
- `?` matches zero or one occurrences of the preceding regexp
- `|` means: either the preceding or the following regexp
- `^` matches the regexp at beginning of a line
- `$` matches the regexp at the end of a line

Meta Characters 2

- **()** groups regexps; groups can be used for further processing.
- `(a|b|c)` matches a or b or c.
- {**n**} matches range of exactly n occurrences of single character
- {**n,**} matches range of n or more occurrences of single character
- {**n,m**} matches range of n up to and including m occurrences of single character

Meta Characters 3

- `[ ]` : character class:
  - `[a-z]` only a lower case character
  - `[a-zA-Z][.?!]` matches any character upper/lowercase followed by `. or ! or ?`
  - `[-+*/]` matches arithmetic operators; - should be at beginning or end!
- excluding class: with `[^..]`
  - `[^0-9]` matches any character except a digit
  - `[^aeiou]` matches anything but vowels

Meta Characters 4

- **A** matches only the beginnig of a string
- **Z** Matches only at the end of the string
- **b** matches the empty string, but only at the beginning or end of a word. A word is defined as a sequence of alphanumeric or underscore characters, so the end of a word is indicated by whitespace or a non-alphanumeric, non-underscore character.
- **B** matches the empty string,but only when it is not at the beginning or end of a word. This is just the opposite of \b
- **d** matches a digit; this is equivalent to the set [0-9]
- **D** matches any non-digit character; this is equivalent to the set [^0-9]

Meta Characters 5

- **s** matches a white space character; this is equivalent to the set `[ \t\n\r\f\v]`
- **S** matches any non-whitespace character; this is equivalent to the set `[^ \t\n\r\f\v]`
- **w** matches any alphanumeric character and the underscore; this is equivalent to the set `[a-zA-Z0-9_]`
- **W** matches any non-alphanumeric character; this is equivalent to the set `[^a-zA-Z0-9_]`
- **$** matches the character '$', because $ is a meta character

Examples: 1

```
static final Pattern COIN = Pattern.compile( "↩
   ^[125]0?$" );

static boolean isCoinFaceNumber( String value )↩
   {
    return COIN.matcher( value ).matches();
}
```

```
            // valid faces
            {"1", true},
            {"2", true},
            {"5", true},
            {"10", true},
            {"20", true},
            {"50", true},
            // phony faces
            {"100", false},
            {"200", false},
            {"25", false},
            {"15", false},
            {"100", false},
            {"250", false},
            {"500", false},
            {"222", false},
            {"251", false},
            {"12", false},
```

Examples: 2

```
static final Pattern BILJET =
        Pattern.compile( "^(50{0,2}|[12]0{1,2})↩
           $" );
```

```
        // valid faces
        {"5", true},
        {"50", true},
        {"500", true},
        {"10", true},
        {"100", true},
        {"20", true},
        {"200", true},
        // phony faces
        {"2", false},
        {"1", false},
        {"1000", false},
        {"502", false},
        {"51", false},
        {"150", false},
        {"250", false},
        {"5000", false},
        {"1250", false},
        {"12", false},
```