Regular Expression

by

Raj Nath Patel
CDAC Mumbai

Regular Expression

- A regular expression (regex or regexp for short) is a special text string for describing a search pattern
- Topics
  - Literal vs Metacharacters
  - Character Classes or Character Sets
  - Shorthand Character Classes
  - The Dot Matches (Almost) Any Character
  - Anchors

Regular Expression contd…

- Topics contd…
  - Alternation
  - Repetition
    - Greedy and Lazy Repetition
  - Word boundaries
  - Grouping and Capturing

Literals V Metacharacters

- Metacharacters are the characters which are having special meaning in any particular scenario
- In the case of REGEX we have following 12 metacharacters
  - backslash \, caret ^, dollar sign $, period or dot ., vertical bar or pipe symbol |, question mark ?, asterisk or star *, plus sign +, opening parenthesis (, closing parenthesis), opening square bracket [, opening curly brace {
  - If you want to use any of these characters as a literal in a regex, you need to escape them with a backslash
- Other than metacharacters all are literal

Character Classes and Set

- A "character class" matches only one out of several characters
  - Eg: **gr[ae]y** will match either **gray** or **grey**
- You can use a hyphen inside a character class to specify a range of characters
  - **[0-9]** will matches a single digit between **0 and 9**
- Typing a caret after the opening square bracket negates the character class
  - **q[^x]** matches **qu** in question. It does not match **Iraq**, WHY?

Shorthand Character Classes

- The actual characters matched by the shorthands depends on the software you're using
  - **\w** Matches word characters
  - **\W** Matches nonword characters
  - **\s** Matches whitespace. Equivalent to [\t\n\r\f]
  - **\S** Matches nonwhitespace
  - **\d** Matches digits. Equivalent to [0-9]
  - **\D** Matches nondigits

The Dot Matches (Almost) Any Character

- The dot matches a single character, except line break characters
- Some applications have a **"dot matches all"** or **"single line"** mode that makes the dot match any single character, including line breaks
  - Eg: **gr.y** matches **gray, grey, gr%y,** etc.
- Use dot only when no other option
  - Often, a character class or negated character class is faster and more precise

Anchors

- “^” Matches beginning of a String or line
  - Eg: **^(dog)** matches **dogs and cats** but not cats and dogs
- “$” Matches end of a String or line
  - Eg: **cats$** matches **dogs and cats** but not cats and dogs

Alternation

- Alternation is the regular expression equivalent of "or"
  - Eg: **cat|dog** matches **cat** in **About cats and dogs**
    - If the regex is applied again, it matches **dog**
- Alternation has the lowest precedence of all regex operators

Repetition

- “?” question mark
  - It tells the engine to attempt to match the preceding token zero times or once
  - Eg: **colou?r** matches **colour** or **color**
- You can make several tokens optional by grouping them together using parentheses
  - Eg: **Nov(ember)?** matches **Nov and November**
- Make the **question mark lazy** by putting a second question mark after the first
  - Apply **Feb 23(rd)?** and **Feb 23(rd)??** to the string **Today is Feb 23rd, 2003** ?

Repetition contd…

- “*” asterisk or star
  - Tells the engine to attempt to match the preceding token zero or more times
  - Eg: colo**u*** r matches color, colour, colouur, colouuur
- “+” plus
  - Tells the engine to attempt to match the preceding token once or more
  - Eg: Eg: colo**u+**r matches colour, colouur, colouuur

Repetition contd…

- Use curly braces to specify a specific amount of repetition : **re{min,max}**
  - **re{n}** Matches exactly n number of occurrences of preceding expression
    - **[1-9][0-9]{3}** matches a number between **1000 and 9999**
  - **re{n,}** Matches n or more occurrences of preceding expression
    - **[1-9][0-9]{3,}** matches a number **greater than 1000**
  - **re{n, m}** Matches at least n and at most m occurrences of preceding expression
    - **[1-9][0-9]{2,4}** matches a number between **100 and 99999**

Word Boundaries

- **\b** allows you to perform a **"whole words only"** search
  - **\bact\b** applying to the **act of character** matches only **act** not the char**act**er
- There are three different positions that qualify as word boundaries:
  - Before the first character in the string, if the first character is a **word character**
  - After the last character in the string, if the last character is a word character
  - Between two characters in the string, where one is a word character and the other is **not a word character**

Word Boundaries contd...

- "word characters" vs "non-word characters"
  - Exactly which characters are **word characters** depends on the regex flavor you're working with
  - In most of the cases **\w** are the characters that are treated as word characters
  - **JAVA** is an exception, which supports **Unicode** for \b not for \w

Grouping and Capturing

- Place parentheses around multiple tokens to group them together
- You can then apply a quantifier to the group
  - Eg: **Set(Value)?** matches **Set or SetValue**
- Parentheses create a capturing group
  - Above example has **one** group
  - After the match, group number **one** contains **nothing** if Set was matched
  - It contains **Value** if **SetValue** was matched

Thank you!
??