

# **REGULAR EXPRESSIONS**

# Определение

Формальный язык поиска и осуществления манипуляций с подстроками в тексте.

Основан на использовании метасимволов.

# СИМВОЛЫ

x ==> СИМВОЛ x

\\ ==> обратный слеш

\xhh ==> СИМВОЛ с кодом U+00hh

\xhhhh ==> СИМВОЛ с кодом U+hhhh

\n ==> перевод строки

\r ==> возврат каретки

\t ==> табуляция

# Символьные классы

Простой класс:

$[abc] \Rightarrow a, b \text{ или } c$

Регулярное выражение:  $[ab]$

Входная строка: accddba

# Символьные классы

Отрицание:

$[^abc]$   $\Rightarrow$  любой символ, кроме a, b, c

Регулярное выражение:  $[^ab]$

Входная строка: acdba

# Символьные классы

Диапазон:

`[a-zA-Z]`  $\Rightarrow$  от a до z или от A до Z

Регулярное выражение: `[a-zA-Z]`

Входная строка: adAcdh

# Символьные классы

Объединение:

$[a-zA-Z]$   $\Rightarrow$  от  $a$  до  $z$  или от  $A$  до  $Z$

Тоже самое, что и  $[a-zA-Z]$

# Символьные классы

Пересечение:

`[a-z&&[def]] ==> d, e или f`

Регулярное выражение: `[a-d&&c-f]`

Входная строка: accdddeab



# Символьные классы

Вычитание:

`[a-z&&[^def]] ==>` от а до с или от g до z

Регулярное выражение: `[a-d&&[^c-f]]`

Входная строка: accdddeab

# Символьные классы Java

Эквиваленты методов класса Character:

\p{javaLowerCase} ~ isLowerCase

\p{javaUpperCase} ~ isUpperCase

\p{javaWhitespace} ~ isWhitespace

Регулярное выражение:

\p{javaUpperCase} \p{javaLowerCase}

Входная строка: Текст

# Предопределенные классы

. ==> любой символ

Регулярное выражение: ...

Входная строка: abcdefgh

Регулярное выражение: ..

Входная строка: abcde

# Предопределенные классы

`\d` ==> цифра, [0-9]

`\D` ==> не цифра, [^\d]

Регулярное выражение: `\d\D`

Входная строка: ab8ab8

# Предопределенные классы

`\s` ==> пробельный символ, [ `\t\n\f\r\x0b`]

`\S` ==> непробельный символ, [`^\s`]

Регулярное выражение: `\s\S`

Входная строка: ab 8 a b8

# Предопределенные классы

`\w ==>` символ слова, `[a-zA-Z_\d]`

`\W ==>` отрицание `\w`, `[^\w]`

Регулярное выражение: `\w\W`

Входная строка: `ab*8&ab8`

# Границы

$\wedge$   $\Rightarrow$  начало строки

Регулярное выражение:  $\wedge ab$

Входная строка:  $ababab$

$\$$   $\Rightarrow$  конец строки

Регулярное выражение:  $ab\$$

Входная строка:  $ababab$

# Границы

`\b` ==> граница слова

`\B` ==> отрицание `\b`

Регулярное выражение: `abc\b`

Входная строка: `abc abcd`



# Границы

$\backslash A \Rightarrow$  начало ввода

Регулярное выражение:  $\backslash Aabc$

Входная строка:

abc abc

abc abc

# Границы

\z ==> конец ввода

Регулярное выражение: **abc\z**

Входная строка:

abc abc

abc abc

# Границы

`\Z`  $\Rightarrow$  конец ввода, как и `\z`, но может включать ограничитель строки

Регулярное выражение: `abc\Z`

Входная строка:

`abc abc`

`abc abc<ограничитель строки>`

# Ограничители строк

'\n' ==> LF (новая строка)

'\r' ==> CR (возврат каретки)

"\r\n" ==> CR+LF

'\u0085' ==> следующая строка

'\u2028' ==> разделитель строки

'\u2029' ==> разделитель параграфа

# Квантификаторы

Квантификатор определяет повторяемость.

Жадный квантификатор определяет максимально возможную подстроку.

Ленивый квантификатор определяет минимально возможную подстроку.

# Квантификаторы

$X?$   $\Rightarrow$  один или ноль раз (жадный)

Регулярное выражение:  $ab?$

Входная строка: aabcabbb

$X??$   $\Rightarrow$  один или ноль раз (ленивый)

Регулярное выражение:  $ab??$

Входная строка: aabcabbb

# Квантификаторы

$X^* \Rightarrow$  ноль или более раз (жадный)

Регулярное выражение:  $ab^*$

Входная строка: aabcabbb

$X^*? \Rightarrow$  ноль или более раз (ленивый)

Регулярное выражение:  $ab^*?$

Входная строка: aabcabbb

# Квантификаторы

$X^+$   $\Rightarrow$  один или более раз (жадный)

Регулярное выражение:  $ab^+$

Входная строка: aabcabbb

$X^+?$   $\Rightarrow$  один или более раз (ленивый)

Регулярное выражение:  $ab^+?$

Входная строка: aabcabbb



# Квантификаторы

$X\{n\} \implies$  ровно  $n$  раз (жадный)  
или (совпадает по результату применения)  
 $X\{n\}? \implies$  ровно  $n$  раз (ленивый)

Регулярное выражение:  $ab\{2\}$  или  $ab\{2\}?$

Входная строка: aabcabbb

# Квантификаторы

$X\{n,\}$   $\Rightarrow$  не менее  $n$  раз (жадный)

Регулярное выражение:  $ab\{2,\}$

Входная строка:  $aabcabbb$

$X\{n,\}?$   $\Rightarrow$  не менее  $n$  раз (ленивый)

Регулярное выражение:  $ab\{2,\}?$

Входная строка:  $aabcabbb$

# Квантификаторы

$X\{n,m\} \Rightarrow$  от  $n$  до  $m$  раз (жадный)

Регулярное выражение:  $ab\{1,2\}$

Входная строка:  $a\underline{ab}c\underline{abb}b$

$X\{n,m\}? \Rightarrow$  от  $n$  до  $m$  раз (ленивый)

Регулярное выражение:  $ab\{1,2\}?$

Входная строка:  $a\underline{ab}c\underline{ab}bb$

# Сверхжадные квантификаторы

При поиске в строке **aab** с помощью рег. выражения **a+b** шаги анализатора:

**a+** ==> **a** (соответствует)

**a+** ==> **aa** (соответствует)

**a+** ==> **aab** (не соответствует)

откат назад (возврат **b**) к последнему соответствию (**aa**) и проверка **a+b**:

**a+b** ==> **aab** (соответствует)

# Сверхжадные квантификаторы

Сверхжадный квантификатор действует как жадный, но никогда не откатывается назад.

`a++ ==> a` (соответствует)

`a++ ==> aa` (соответствует)

`a++ ==> aab` (не соответствует)

Последний символ ввода (b) прочтен, соответствие не найдено.

# Сверхжадные квантификаторы

Чтобы сделать жадный квантификатор сверхжадным достаточно добавить + справа от квантификатора:

$X? \implies X?+$        $X\{n\} \implies X\{n\}+$   
 $X^* \implies X^*+$        $X\{n,\} \implies X\{n,\}+$   
 $X+ \implies X++$        $X\{n,m\} \implies X\{n,m\}+$

Сверхжадные квантификаторы работают как правило быстрее, чем жадные.

# Логические операции

$XY \implies X \text{ за которым следует } Y \text{ (AND)}$

$X|Y \implies X \text{ или } Y \text{ (OR)}$

Приоритет AND выше чем OR.

Регулярное выражение:  $aa|b$

Входная строка: aabcabbb

# Группы

Выражение в круглых скобках - группа.

Каждая группа имеет номер.

Группы нумеруются слева направо, начиная с единицы (номер может быть больше 9)

Чтобы группа не нумеровалась, она должна начинаться с (?:



# Группы

(A)(B(C)(?:D))

(A) ==> группа номер 1

(B(C)(?:D)) ==> группа номер 2

(C) ==> группа номер 3

(?:D) ==> группа без номера

# Группы

Группы могут быть использованы по номеру в регулярном выражении с помощью синтаксиса: \НОМЕР\_ГРУППЫ

Регулярное выражение: (aab)\W\1

Входная строка: aab aab

# Экранирование символов

Для представления специальных  
СИМВОЛОВ:

\ . | + \* ?  
[ ] ( ) { }  
^ \$

используют экранирование с помощью  
обратного слеша:

\\ \. \\ \+ \\* \?  
\[ \] \( \) \{ \}  
\\^ \\\$

# Экранирование символов

Для указания диапазона экранирования  
можно использовать **\Q** и/или **\E**

**\Q==>** начало диапазона

**\E ==>** окончание диапазона

Регулярное выражение: **\Q\(\*\E(a)\1**

Входная строка: ab**\(\*aa**

# Упреждающий просмотр вперед

Позитивный: (?=X)

Регулярное выражение: a(?=b)

Входная строка: abacab

Негативный: (?!X)

Регулярное выражение: a(?!b)

Входная строка: abacab

# Просмотр назад

Позитивный: (?<=X)

Регулярное выражение: (?<=b)a

Входная строка: abacab

Негативный: (?<!X)

Регулярное выражение: (?<!b) a

Входная строка: abacab

# Режимы

Влияют на работу регулярных выражений.  
Каждый режим имеет буквенный код.

- ◉ COMMENTS ==> x
- ◉ CASE\_INSENSITIVE ==> i
- ◉ UNIX\_LINES ==> d
- ◉ DOTALL ==> s
- ◉ UNICODE\_CASE ==> u
- ◉ MULTILINE ==> m

# Режимы

Чтобы включить режим, достаточно предварить регулярное выражение комбинацией: (?КОД\_РЕЖИМА).

(?m)    (?s)

Если нужно включить сразу несколько режимов, можно писать несколько кодов:

(?iu)



# Режимы

COMMENTS ==> x

Режим комментариев. Пробельные символы игнорируются, после символа # можно писать комментарий к рег. выражению.

Регулярное выражение: (?x)a bc #comment

Входная строка: abcab

# Режимы

CASE\_INSENSITIVE ==> i

Игнорирует регистр символов.

UNIX\_LINES ==> d

Разделитель строк только CR (\r)

DOTALL ==> s

Точка (.) может включать \n

# Режимы

UNICODE\_CASE ==> u

Игнорирует регистр символов.

MULTILINE ==> m

Многострочный режим (по умолчанию \$ - конец ввода).