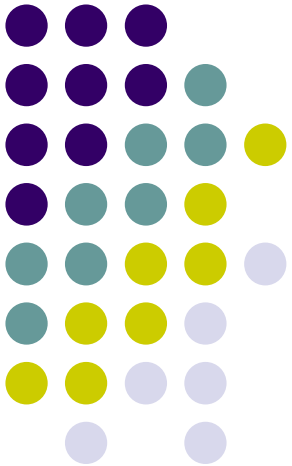
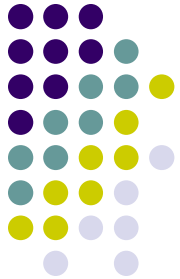
Регулярные выражения

Денис С. Мигинский



Происхождение регулярных выражений



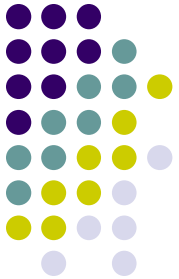
Регулярные выражения – язык, предназначенный для синтаксического разбора текстовых фрагментов. Является развитием wildcard-characters.

Одни из первых реализаций – утилиты sed и grep.

Версии:

- Basic regular expressions (устаревшие)
- POSIX regular expressions
- Perl-compatible regular expressions

Задача



Требуется по известному URL выделить тип протокола, домен, путь

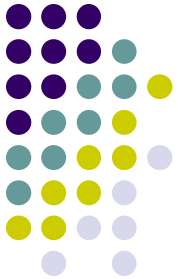
URL=http://ccfit.nsu.ru/~shadow/DT6

Разбор URL

```
url = 'http://ccfit.nsu.ru/~shadow/DT6'
rx = /^(\w+):\/\/([\w\.]+)([\/\w~]+)$/

res = Hash.new
begin
  res['proto'], res['domain'], res['path'] =
    url.match(rx).captures
  res.each{|key, field|
    puts "#{key} = #{field}"
  }
rescue Exception
  puts 'ERROR: incorrect URL'
end
```

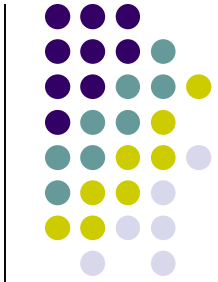
Соответствие строки шаблону



```
 http : / /ccfit.nsu.ru /~shadow/DT6
^(\w+):\/\/([\w\.]+)     ([\/\w~]+)   $
```

^	- начало строки
\w	- произвольная буква/цифра
\w+	- 1+ произвольных букв/цифр
(\w+)	- искомая подстрока
[\w\.]	- любая буква/цифра/'.'
[\w\.]+	- 1+ букв/цифр/'.'
[\/\w~]	- 1+ '\', букв, цифр, '~'
$	- конец строки

Классы символов



[] – класс символов

[^]– дополнение класса символов

\w – буква или цифра [A-Za-z0-9_]

\W – не \w [^\w]

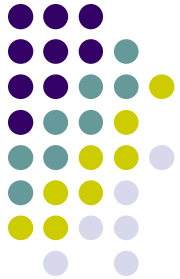
\d – цифра [0-9]

\D – не цифра [^\d]

\s – любой разделитель

. – произвольный символ

Символы позиционирования



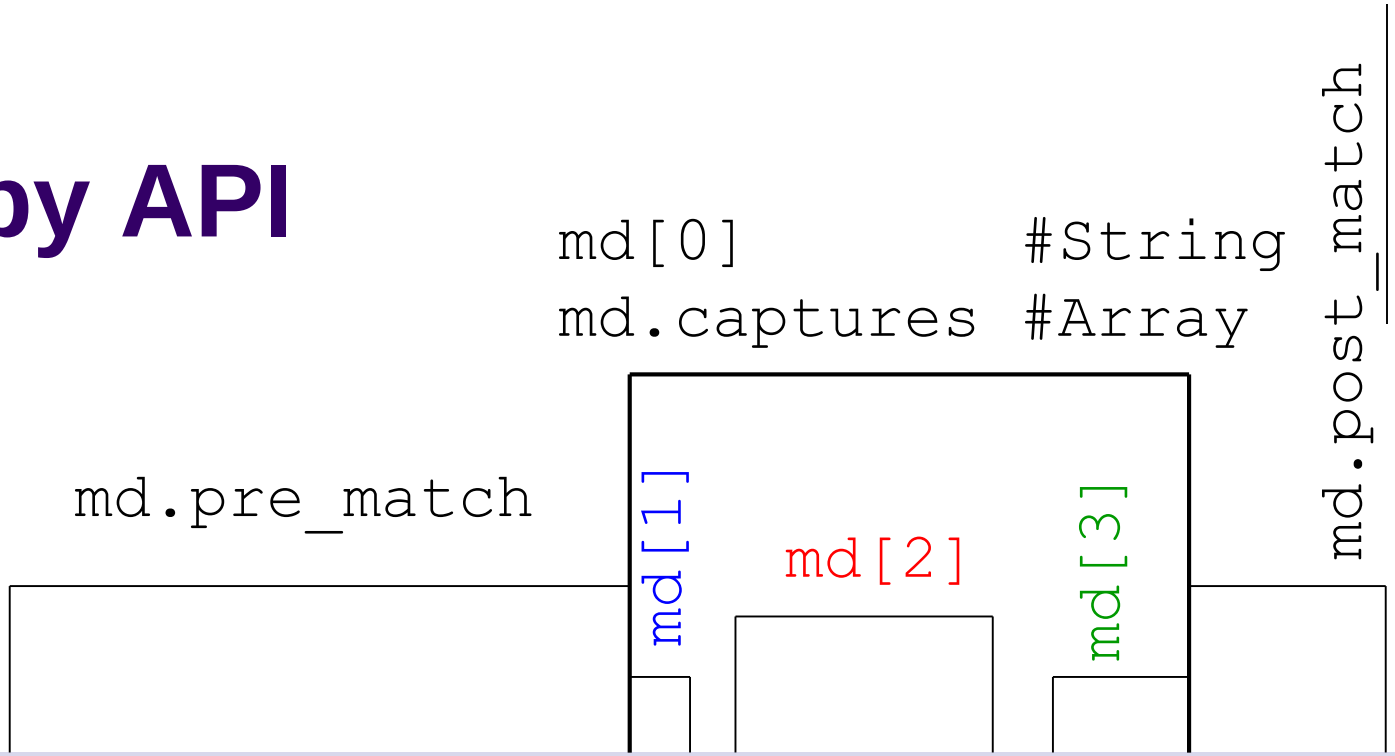
^ - начало строки

$ - конец строки

\b - граница слова

\B – не граница слова

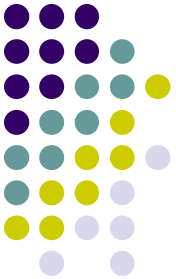
Ruby API




```
md= "Иван IV родился 15 августа 1530 года".
  match(/(\d+)\s+([а-яА-Я]+)\s+(\d+)/)

puts md.class #MatchData
puts md.captures == md[1..3]  #true
puts md[1] == md.captures[0]  #true
```

Ruby API

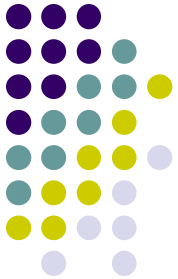


```
String::match(pattern)=>matchdata or nil

Regexp::match(string)=>matchdata or nil

MatchData::
[](i)      =>string
captions   =>array
pre_match  =>string
post_match=>string
```

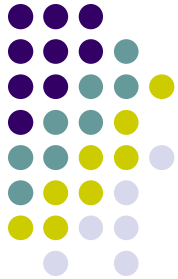
Perl-style Ruby API



```
String::=~(pattern)=>index or nil

md[0]             => $&, $~
md[1..9]          => $1, … $9
md.pre_match      => $`
md.post_match     => $'
```

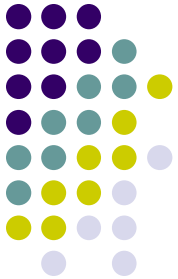
Повторы



#x – произвольный символ или класс

x – ровно один символ
x+ – как минимум один символ
x+? – как минимум один символ («нежадный»)
x* – любое количество символов
x*? – любое количество символов («нежадный»)
x? – не более одного символа
x{m,n} – от **m** до **n** символов
x{m,} – не менее **m** символов
x{m} – ровно **m** символов

Жадные и нежадные регулярные выражения



#обе группы жадные

```
"12345".match(/(\d+)(\d+)/)
```

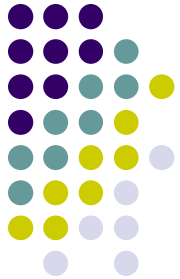
#2-ая группа жадная

```
"12345".match(/(\d+?)(\d+)/)
```

#обе группы нежадные

```
"12345".match(/(\d+?)(\d+?)/)
```

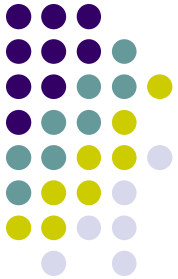
Опции регулярных выражений



```
/rx/imx

i – игнорировать регистр
m – разбирать текст как многострочный
      с m ^, $ срабатывают на \n,
      в противном случае только в
      начале и конце текста
x – расширенное регулярное выражение
```

Расширенные регулярные выражение



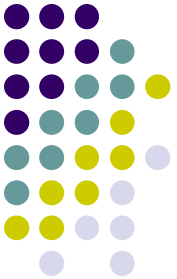
#обычное регулярное выражение

```
rx = /^(\w+):\/\/([\w\.]+)([\/\w~]+)$/
```

#расширенное регулярное выражение:
#игнорируются разделители, допускаются комментарии

```
rx = /^
      (\w+)          #protocol
      :\/\/          #://
      ([\w\.]+)      #domain
      ([\/\w~]+)     #path
      $/x
```

Замена на основе регулярных выражений



```
str="plum is purple, apple is green"
colour='purple|green'

puts str.sub(/#{colour}/, "coloured")
```

#plum is coloured, apple is green

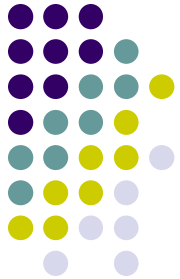
```
puts str.gsub(/#{colour}/, "coloured")
```

#plum is coloured, apple is coloured

```
puts str.sub(   /(#{colour})(.*)\1/,
                '\3\2\1')
```

#plum is green, apple is purple

Другие операции на основе регулярных выражений



#возвращает все подстроки, удовлетворяющие шаблону (regex или строка)

```
String::scan(pattern)[{|match,...| block}]
     =>array
```

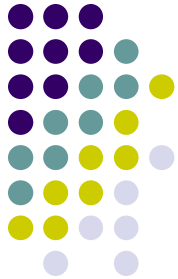
#разделяет исходную строку на подстроки по шаблону (regex или строка)

```
String::split(pattern[,limit])=>array
```

#выделяет первое совпадение с шаблоном

```
String::slice(regex)=>string
String::[](regex)   =>string
```

Задача 3



Вариант 1:
Написать программу, разбирающую журнал событий (например, журнал web-сервера, syslog и т.д.) и вычисляющий ежемесячную статистику в зависимости от содержимого выбранного журнала (например, обращения с различных IP-адресов, ошибки различных типов, аутентификацию пользователей и т.д.)

Вариант 2:
По детализации звонков (предварительно преобразованным в текстовый формат) своего сотового оператора за несколько месяцев сделать расчет финансовых затрат по нескольким различным тарифам. Допускается использовать не все параметры тарифа, только наиболее важные (например, абонентская плата, стоимость звонков, дифференцированная по различным видам номеров).