

ParallelRegex.php

```
1. <?php
2. /**
3.  * Lexer adapted from Simple Test:
4.  * http://sourceforge.net/projects/simpletest/
5.  * For an intro to the Lexer see:
6.  *
7.  * https://web.archive.org/web/20120125041816/http://www.phppatterns.com/docs/develop/simple\_test\_lexer\_notes
8.  *
9.  * @author Marcus Baker http://www.lastcraft.com
10. */
11.
12. namespace dokuwiki\Parsing\Lexer;
13.
14. /**
15.  * Compounded regular expression.
16.  * Any of the contained patterns could match and when one does
17.  * it's label is returned.
18.  */
19. class ParallelRegex
20. {
21.     /** @var string[] patterns to match */
22.     protected $patterns;
23.     /** @var string[] labels for above patterns */
24.     protected $labels;
25.     /** @var string the compound regex matching all patterns */
26.     protected $regex;
27.     /** @var bool case sensitive matching? */
28.     protected $case;
29.
30.     /**
31.      * Constructor. Starts with no patterns.
32.      *
33.      * @param boolean $case True for case sensitive, false
34.      * for insensitive.
35.      */
36.     public function __construct($case)
37.     {
38.         $this->case = $case;
39.         $this->patterns = array();
40.         $this->labels = array();
41.         $this->regex = null;
42.     }
43.
44.     /**
45.      * Adds a pattern with an optional label.
46.      *
47.      * @param string $pattern The regular expression to match.
48.      * @param string $label The label to return when this pattern matches.
49.      * @return void
50.      */
51.     public function add($pattern, $label = null)
52.     {
53.         $this->patterns[] = $pattern;
54.         $this->labels[] = $label;
55.         $this->regex = null;
56.     }
57.
58.     /**
59.      * Returns the compound regular expression.
60.      *
61.      * @return string The compound regular expression.
62.      */
63.     public function getRegex()
64.     {
65.         if ($this->regex === null)
66.         {
67.             $this->regex = implode('|', $this->patterns);
68.             if ($this->case === false)
69.             {
70.                 $this->regex = '(?i)' . $this->regex;
71.             }
72.         }
73.         return $this->regex;
74.     }
75.
76.     /**
77.      * Returns the label for the first matching pattern.
78.      *
79.      * @param string $text The text to match.
80.      * @return string The label for the first matching pattern, or null if no match.
81.      */
82.     public function match($text)
83.     {
84.         $regex = $this->getRegex();
85.         preg_match($regex, $text, $matches);
86.         if ($matches)
87.         {
88.             return $this->labels[0];
89.         }
90.         return null;
91.     }
92.
93.     /**
94.      * Returns all matching labels.
95.      *
96.      * @param string $text The text to match.
97.      * @return array An array of labels for all matching patterns.
98.      */
99.     public function matchAll($text)
100.     {
101.         $regex = $this->getRegex();
102.         preg_match_all($regex, $text, $matches);
103.         $labels = array();
104.         foreach ($matches as $match)
105.         {
106.             $labels[] = $this->labels[0];
107.         }
108.         return $labels;
109.     }
110. }
111.
112. // Example usage
113. $lexer = new ParallelRegex(false);
114. $lexer->add('/\d+/', 'digits');
115. $lexer->add('/\w+/', 'word');
116. $lexer->add('/\s+/', 'space');
117.
118. $text = '123 abc 456';
119. $label = $lexer->match($text);
120. echo "Matched: " . $label . "\n";
121.
122. $labels = $lexer->matchAll($text);
123. print_r($labels);
124.
125. // Output:
126. // Matched: digits
127. // Array (
128. //     0 => digits
129. //     1 => word
130. //     2 => digits
131. // )
132.
133. // Note: The order of labels depends on the order of patterns in the lexer.
134.
135. // License: MIT
136. // Copyright (c) 2005-2010 Marcus Baker
137. // All rights reserved.
138. // Permission is granted to anyone to use this software for any purpose,
139. // provided the original author and source are credited.
```

```
45.     * @param mixed     $pattern Perl style regex. Must be UTF-8
46.     *
    (, )
47.     *
    lose their meaning unless they
48.     *
    form part of a lookahead or
49.     *
    lookbehind assertion.
50.     * @param bool|string $label Label of regex to be returned
51.     *
    on a match. Label must be ASCII
52.     */
53.     public function addPattern($pattern, $label = true)
54.     {
55.         $count = count($this->patterns);
56.         $this->patterns[$count] = $pattern;
57.         $this->labels[$count] = $label;
58.         $this->regex = null;
59.     }
60.
61.     /**
62.     * Attempts to match all patterns at once against a string.
63.     *
64.     * @param string $subject String to match against.
65.     * @param string $match First matched portion of
66.     *
    subject.
67.     * @return bool|string False if no match found, label
    if label exists, true if not
68.     */
69.     public function apply($subject, &$match)
70.     {
71.         if (count($this->patterns) == 0) {
72.             return false;
73.         }
74.         if (! preg_match($this->getCompoundedRegex(), $subject,
    $matches)) {
75.             $match = "";
76.             return false;
77.         }
78.
79.         $match = $matches[0];
80.         $size = count($matches);
81.         // FIXME this could be made faster by storing the labels
    as keys in a hashmap
82.         for ($i = 1; $i < $size; $i++) {
83.             if ($matches[$i] && isset($this->labels[$i - 1])) {
84.                 return $this->labels[$i - 1];
85.             }
86.         }
87.         return true;
88.     }
89.
```

```
90.     /**
91.      * Attempts to split the string against all patterns at once
92.      *
93.      * @param string $subject      String to match against.
94.      * @param array $split        The split result: array
    containing, pre-match, match & post-match strings
95.      * @return boolean           True on success.
96.      *
97.      * @author Christopher Smith <chris@jalakai.co.uk>
98.      */
99.     public function split($subject, &$split)
100.    {
101.        if (count($this->patterns) == 0) {
102.            return false;
103.        }
104.
105.        if (! preg_match($this->getCompoundedRegex(), $subject,
    $matches)) {
106.            if (function_exists('preg_last_error')) {
107.                $err = preg_last_error();
108.                switch ($err) {
109.                    case PREG_BACKTRACK_LIMIT_ERROR:
110.                        msg('A PCRE backtrack error occurred. Try
    to increase the pcre.backtrack_limit in php.ini', -1);
111.                        break;
112.                    case PREG_RECURSION_LIMIT_ERROR:
113.                        msg('A PCRE recursion error occurred. Try
    to increase the pcre.recursion_limit in php.ini', -1);
114.                        break;
115.                    case PREG_BAD_UTF8_ERROR:
116.                        msg('A PCRE UTF-8 error occurred. This
    might be caused by a faulty plugin', -1);
117.                        break;
118.                    case PREG_INTERNAL_ERROR:
119.                        msg('A PCRE internal error occurred. This
    might be caused by a faulty plugin', -1);
120.                        break;
121.                }
122.            }
123.
124.            $split = array($subject, "", "");
125.            return false;
126.        }
127.
128.        $idx = count($matches)-2;
129.        list($pre, $post) =
    preg_split($this->patterns[$idx].$this->getPerlMatchingFlags(),
    $subject, 2);
130.        $split = array($pre, $matches[0], $post);
131.
132.        return isset($this->labels[$idx]) ? $this->labels[$idx] :
```

```
    true;
133.     }
134.
135.     /**
136.      * Compounds the patterns into a single
137.      * regular expression separated with the
138.      * "or" operator. Caches the regex.
139.      * Will automatically escape (, ) and / tokens.
140.      *
141.      * @return null|string
142.      */
143.     protected function getCompoundedRegex()
144.     {
145.         if ($this->regex == null) {
146.             $sCnt = count($this->patterns);
147.             for ($i = 0; $i < $sCnt; $i++) {
148.                 /*
149.                  * decompose the input pattern into "(", "(?",
150.                  * "[...]", "[...]..", "[^]..", "[...[:...:]]..",
151.                  * "\x"...
152.                  * elements.
153.                  */
154.                 preg_match_all('/\\\\\.|' .
155.                                 '\\(?:|' .
156.                                 '\\[()|]' .
157.                                 '\\[\\^?\\]?(?:\\\\\.|\\[:[\\^]]*:\\|\\[\\^\\\\\\)]*\\|)' .
158.                                 '\\^[()\\\\\\]+/' ,
159.                                 $this->patterns[$i], $elts);
160.
161.                 $pattern = "";
162.                 $level = 0;
163.
164.                 foreach ($elts[0] as $elt) {
165.                     /*
166.                      * for "(", ")" remember the nesting level,
167.                      * only to the non-"(? " ones.
168.                      */
169.                     switch ($elt) {
170.                         case '(':
171.                             $pattern .= '\\(';
172.                             break;
173.                         case ')':
174.                             if ($level > 0)
175.                                 $level--; /* closing (? */
176.                             else $pattern .= '\\)';
```

```
176.         $pattern .= ')';
177.         break;
178.     case '(':
179.         $level++;
180.         $pattern .= '(';
181.         break;
182.     default:
183.         if (substr($elt, 0, 1) == '\\')
184.             $pattern .= $elt;
185.         else $pattern .= str_replace('/',
'\', $elt);
186.     }
187. }
188. $this->patterns[$i] = "($pattern)";
189. }
190. $this->regex = "/" . implode("|", $this->patterns) .
"/" . $this->getPerlMatchingFlags();
191. }
192. return $this->regex;
193. }
194.
195. /**
196.  * Accessor for perl regex mode flags to use.
197.  * @return string      Perl regex flags.
198.  */
199. protected function getPerlMatchingFlags()
200. {
201.     return ($this->case ? "msS" : "msSi");
202. }
203. }
```

From:

<https://wwoss.ru/> - worldwide open-source software

Permanent link:

<https://wwoss.ru/doku.php?id=wiki:xref:dokuwiki:inc:parsing:lexer:parallelregex.php>

Last update: 2025/01/16 19:43

