

Lists.php

```
1. <?php
2.
3. namespace dokuwiki\Parsing\Handler;
4.
5. class Lists extends AbstractRewriter
6. {
7.     protected $listCalls = array();
8.     protected $listStack = array();
9.
10.    protected $initialDepth = 0;
11.
12.    const NODE = 1;
13.
14.    /** @inheritdoc */
15.    public function finalise()
16.    {
17.        $last_call = end($this->calls);
18.        $this->writeCall(array('list_close', array(),
19.                                $last_call[2]));
20.
21.        $this->process();
22.        $this->callWriter->finalise();
23.        unset($this->callWriter);
24.    }
25.
26.    /** @inheritdoc */
27.    public function process()
28.    {
29.        foreach ($this->calls as $call) {
30.            switch ($call[0]) {
31.                case 'list_item':
32.                    $this->listOpen($call);
33.                    break;
34.                case 'list_open':
35.                    $this->listStart($call);
36.                    break;
37.                case 'list_close':
38.                    $this->listEnd($call);
39.                    break;
40.                default:
41.                    $this->listContent($call);
42.                    break;
43.            }
44.        }
45.
46.        $this->callWriter->writeCalls($this->listCalls);
47.        return $this->callWriter;
```

```
48.     }
49.
50.     protected function listStart($call)
51.     {
52.         $depth = $this->interpretSyntax($call[1][0], $listType);
53.
54.         $this->initialDepth = $depth;
55.         //          array(list type, current depth, index
of current listitem_open)
56.         $this->listStack[] = array($listType, $depth, 1);
57.
58.         $this->listCalls[] =
array('list'.$listType.'_open',array(),$call[2]);
59.         $this->listCalls[] =
array('listitem_open',array(1),$call[2]);
60.         $this->listCalls[] =
array('listcontent_open',array(),$call[2]);
61.     }
62.
63.
64.     protected function listEnd($call)
65.     {
66.         $closeContent = true;
67.
68.         while ($list = array_pop($this->listStack)) {
69.             if ($closeContent) {
70.                 $this->listCalls[] =
array('listcontent_close',array(),$call[2]);
71.                 $closeContent = false;
72.             }
73.             $this->listCalls[] =
array('listitem_close',array(),$call[2]);
74.             $this->listCalls[] = array('list'.$list[0].'_close',
array(), $call[2]);
75.         }
76.     }
77.
78.     protected function listOpen($call)
79.     {
80.         $depth = $this->interpretSyntax($call[1][0], $listType);
81.         $send = end($this->listStack);
82.         $key = key($this->listStack);
83.
84.         // Not allowed to be shallower than initialDepth
85.         if ($depth < $this->initialDepth) {
86.             $depth = $this->initialDepth;
87.         }
88.
89.         if ($depth == $send[1]) {
90.             // Just another item in the list...
```

```
91.         if ($listType == $end[0]) {
92.             $this->listCalls[] =
array('listcontent_close',array(),$call[2]);
93.             $this->listCalls[] =
array('listitem_close',array(),$call[2]);
94.             $this->listCalls[] =
array('listitem_open',array($depth-1),$call[2]);
95.             $this->listCalls[] =
array('listcontent_open',array(),$call[2]);
96.
97.             // new list item, update list stack's index into
current listitem_open
98.             $this->listStack[$key][2] =
count($this->listCalls) - 2;
99.
100.            // Switched list type...
101.            } else {
102.                $this->listCalls[] =
array('listcontent_close',array(),$call[2]);
103.                $this->listCalls[] =
array('listitem_close',array(),$call[2]);
104.                $this->listCalls[] =
array('list'.$end[0].'_close', array(), $call[2]);
105.                $this->listCalls[] =
array('list'.$listType.'_open', array(), $call[2]);
106.                $this->listCalls[] = array('listitem_open',
array($depth-1), $call[2]);
107.                $this->listCalls[] =
array('listcontent_open',array(),$call[2]);
108.
109.                array_pop($this->listStack);
110.                $this->listStack[] = array($listType, $depth,
count($this->listCalls) - 2);
111.            }
112.            } elseif ($depth > $end[1]) { // Getting deeper...
113.                $this->listCalls[] =
array('listcontent_close',array(),$call[2]);
114.                $this->listCalls[] = array('list'.$listType.'_open',
array(), $call[2]);
115.                $this->listCalls[] = array('listitem_open',
array($depth-1), $call[2]);
116.                $this->listCalls[] =
array('listcontent_open',array(),$call[2]);
117.
118.                // set the node/leaf state of this item's parent
listitem_open to NODE
119.                $this->listCalls[$this->listStack[$key][2]][1][1] =
self::NODE;
120.
121.                $this->listStack[] = array($listType, $depth,
count($this->listCalls) - 2);
```

```
122.         } else { // Getting shallower ( $depth < $end[1] )
123.             $this->listCalls[] =
array('listcontent_close',array(),$call[2]);
124.             $this->listCalls[] =
array('listitem_close',array(),$call[2]);
125.             $this->listCalls[] =
array('list'.$end[0].'_close',array(),$call[2]);
126.
127.             // Throw away the end - done
128.             array_pop($this->listStack);
129.
130.             while (1) {
131.                 $end = end($this->listStack);
132.                 $key = key($this->listStack);
133.
134.                 if ($end[1] <= $depth) {
135.                     // Normalize depths
136.                     $depth = $end[1];
137.
138.                     $this->listCalls[] =
array('listitem_close',array(),$call[2]);
139.
140.                     if ($end[0] == $listType) {
141.                         $this->listCalls[] =
array('listitem_open',array($depth-1),$call[2]);
142.                         $this->listCalls[] =
array('listcontent_open',array(),$call[2]);
143.
144.                         // new list item, update list stack's
index into current listitem_open
145.                         $this->listStack[$key][2] =
count($this->listCalls) - 2;
146.                         } else {
147.                             // Switching list type...
148.                             $this->listCalls[] =
array('list'.$end[0].'_close', array(), $call[2]);
149.                             $this->listCalls[] =
array('list'.$listType.'_open', array(), $call[2]);
150.                             $this->listCalls[] =
array('listitem_open', array($depth-1), $call[2]);
151.                             $this->listCalls[] =
array('listcontent_open',array(),$call[2]);
152.
153.                             array_pop($this->listStack);
154.                             $this->listStack[] = array($listType,
$depth, count($this->listCalls) - 2);
155.                         }
156.
157.                         break;
158.
```

```
159.          // Haven't dropped down far enough yet.... (
    $end[1] > $depth )
160.          } else {
161.              $this->listCalls[] =
    array('listitem_close',array(),$call[2]);
162.              $this->listCalls[] =
    array('list'.$end[0].'_close',array(),$call[2]);
163.
164.              array_pop($this->listStack);
165.          }
166.      }
167.  }
168.  }
169.
170.  protected function listContent($call)
171.  {
172.      $this->listCalls[] = $call;
173.  }
174.
175.  protected function interpretSyntax($match, & $type)
176.  {
177.      if (substr($match, -1) == '*') {
178.          $type = 'u';
179.      } else {
180.          $type = 'o';
181.      }
182.      // Is the +1 needed? It used to be count(explode(...))
183.      // but I don't think the number is seen outside this
    handler
184.      return substr_count(str_replace("\t", ' ', $match), ' ');
185.  }
186. }
```

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

Permanent link:
<https://wwoss.ru/doku.php?id=wiki:xref:dokuwiki:inc:parsing:handler:lists.php>

Last update: **2025/01/16 22:01**

