Table.php

```php
<?php

namespace dokuwiki\Parsing\Handler;

class Table extends AbstractRewriter
{

    protected $tableCalls = array();
    protected $maxCols = 0;
    protected $maxRows = 1;
    protected $currentCols = 0;
    protected $firstCell = false;
    protected $lastCellType = 'tablecell';
    protected $inTableHead = true;
    protected $currentRow = array('tableheader' => 0, 'tablecell' => 0);
    protected $countTableHeadRows = 0;

    /** @inheritdoc */
    public function finalise()
    {
        $last_call = end($this->calls);
        $this->writeCall(array('table_end',array(), $last_call[2]));

        $this->process();
        $this->callWriter->finalise();
        unset($this->callWriter);
    }

    /** @inheritdoc */
    public function process()
    {
        foreach ($this->calls as $call) {
            switch ($call[0]) {
                case 'table_start':
                    $this->tableStart($call);
                    break;
                case 'table_row':
                    $this->tableRowClose($call);
                    $this->tableRowOpen(array('tablerow_open',$call[1],$call[2]));
                    break;
                case 'tableheader':
                case 'tablecell':
                    $this->tableCell($call);
                    break;
                case 'table_end':
```

```
                    $this->tableRowClose($call);
                    $this->tableEnd($call);
                    break;
                default:
                    $this->tableDefault($call);
                    break;
            }
        }
        $this->callWriter->writeCalls($this->tableCalls);

        return $this->callWriter;
    }

    protected function tableStart($call)
    {
        $this->tableCalls[] =
array('table_open',$call[1],$call[2]);
        $this->tableCalls[] =
array('tablerow_open',array(),$call[2]);
        $this->firstCell = true;
    }

    protected function tableEnd($call)
    {
        $this->tableCalls[] =
array('table_close',$call[1],$call[2]);
        $this->finalizeTable();
    }

    protected function tableRowOpen($call)
    {
        $this->tableCalls[] = $call;
        $this->currentCols = 0;
        $this->firstCell = true;
        $this->lastCellType = 'tablecell';
        $this->maxRows++;
        if ($this->inTableHead) {
            $this->currentRow = array('tablecell' => 0,
'tableheader' => 0);
        }
    }

    protected function tableRowClose($call)
    {
        if ($this->inTableHead && ($this->inTableHead =
$this->isTableHeadRow())) {
            $this->countTableHeadRows++;
        }
        // Strip off final cell opening and anything after it
        while ($discard = array_pop($this->tableCalls)) {
```

```
91.             if ($discard[0] == 'tablecell_open' || $discard[0] ==
    'tableheader_open') {
92.                 break;
93.             }
94.             if (!empty($this->currentRow[$discard[0]])) {
95.                 $this->currentRow[$discard[0]]--;
96.             }
97.         }
98.         $this->tableCalls[] = array('tablerow_close', array(),
    $call[2]);
99.
100.        if ($this->currentCols > $this->maxCols) {
101.            $this->maxCols = $this->currentCols;
102.        }
103.    }
104.
105.    protected function isTableHeadRow()
106.    {
107.        $td = $this->currentRow['tablecell'];
108.        $th = $this->currentRow['tableheader'];
109.
110.        if (!$th || $td > 2) return false;
111.        if (2*$td > $th) return false;
112.
113.        return true;
114.    }
115.
116.    protected function tableCell($call)
117.    {
118.        if ($this->inTableHead) {
119.            $this->currentRow[$call[0]]++;
120.        }
121.        if (!$this->firstCell) {
122.            // Increase the span
123.            $lastCall = end($this->tableCalls);
124.
125.            // A cell call which follows an open cell means an
    empty cell so span
126.            if ($lastCall[0] == 'tablecell_open' || $lastCall[0]
    == 'tableheader_open') {
127.                $this->tableCalls[] =
    array('colspan',array(),$call[2]);
128.            }
129.
130.            $this->tableCalls[] =
    array($this->lastCellType.'_close',array(),$call[2]);
131.            $this->tableCalls[] =
    array($call[0].'_open',array(1,null,1),$call[2]);
132.            $this->lastCellType = $call[0];
133.        } else {
134.            $this->tableCalls[] =
```

```
    array($call[0].'_open',array(1,null,1),$call[2]);
135.             $this->lastCellType = $call[0];
136.             $this->firstCell = false;
137.         }
138. 
139.         $this->currentCols++;
140.     }
141. 
142.     protected function tableDefault($call)
143.     {
144.         $this->tableCalls[] = $call;
145.     }
146. 
147.     protected function finalizeTable()
148.     {
149. 
150.         // Add the max cols and rows to the table opening
151.         if ($this->tableCalls[0][0] == 'table_open') {
152.             // Adjust to num cols not num col delimeters
153.             $this->tableCalls[0][1][] = $this->maxCols - 1;
154.             $this->tableCalls[0][1][] = $this->maxRows;
155.             $this->tableCalls[0][1][] =
    array_shift($this->tableCalls[0][1]);
156.         } else {
157.             trigger_error('First element in table call list is not
    table_open');
158.         }
159. 
160.         $lastRow = 0;
161.         $lastCell = 0;
162.         $cellKey = array();
163.         $toDelete = array();
164. 
165.         // if still in tableheader, then there can be no table
    header
166.         // as all rows can't be within <THEAD>
167.         if ($this->inTableHead) {
168.             $this->inTableHead = false;
169.             $this->countTableHeadRows = 0;
170.         }
171. 
172.         // Look for the colspan elements and increment the colspan
    on the
173.         // previous non-empty opening cell. Once done, delete all
    the cells
174.         // that contain colspans
175.         for ($key = 0; $key < count($this->tableCalls); ++$key) {
176.             $call = $this->tableCalls[$key];
177. 
178.             switch ($call[0]) {
```

```
179.                 case 'table_open':
180.                     if ($this->countTableHeadRows) {
181.                         array_splice($this->tableCalls, $key+1, 0,
    array(
182.
    array('tablethead_open', array(), $call[2])));
183.                     }
184.                     break;
185.
186.                 case 'tablerow_open':
187.                     $lastRow++;
188.                     $lastCell = 0;
189.                     break;
190.
191.                 case 'tablecell_open':
192.                 case 'tableheader_open':
193.                     $lastCell++;
194.                     $cellKey[$lastRow][$lastCell] = $key;
195.                     break;
196.
197.                 case 'table_align':
198.                     $prev = in_array($this->tableCalls[$key-1][0],
    array('tablecell_open', 'tableheader_open'));
199.                     $next = in_array($this->tableCalls[$key+1][0],
    array('tablecell_close', 'tableheader_close'));
200.                     // If the cell is empty, align left
201.                     if ($prev && $next) {
202.                         $this->tableCalls[$key-1][1][1] = 'left';
203.
204.                         // If the previous element was a cell
    open, align right
205.                     } elseif ($prev) {
206.                         $this->tableCalls[$key-1][1][1] = 'right';
207.
208.                         // If the next element is the close of an
    element, align either center or left
209.                     } elseif ($next) {
210.                         if
    ($this->tableCalls[$cellKey[$lastRow][$lastCell]][1][1] ==
    'right') {
211.
    $this->tableCalls[$cellKey[$lastRow][$lastCell]][1][1] = 'center';
212.                         } else {
213.
    $this->tableCalls[$cellKey[$lastRow][$lastCell]][1][1] = 'left';
214.                         }
215.                     }
216.
217.                     // Now convert the whitespace back to cdata
218.                     $this->tableCalls[$key][0] = 'cdata';
219.                     break;
```

```
220.
221.                 case 'colspan':
222.                     $this->tableCalls[$key-1][1][0] = false;
223.
224.                     for ($i = $key-2; $i >= $cellKey[$lastRow][1];
    $i--) {
225.                         if ($this->tableCalls[$i][0] ==
    'tablecell_open' ||
226.                             $this->tableCalls[$i][0] ==
    'tableheader_open'
227.                         ) {
228.                             if (false !==
    $this->tableCalls[$i][1][0]) {
229.                                 $this->tableCalls[$i][1][0]++;
230.                                 break;
231.                             }
232.                         }
233.                     }
234.
235.                     $toDelete[] = $key-1;
236.                     $toDelete[] = $key;
237.                     $toDelete[] = $key+1;
238.                     break;
239.
240.                 case 'rowspan':
241.                     if ($this->tableCalls[$key-1][0] == 'cdata') {
242.                         // ignore rowspan if previous call was
    cdata (text mixed with :::)
243.                         // we don't have to check next call as
    that wont match regex
244.                         $this->tableCalls[$key][0] = 'cdata';
245.                     } else {
246.                         $spanning_cell = null;
247.
248.                         // can't cross thead/tbody boundary
249.                         if (!$this->countTableHeadRows ||
    ($lastRow-1 != $this->countTableHeadRows)) {
250.                             for ($i = $lastRow-1; $i > 0; $i--) {
251.                                 if
    ($this->tableCalls[$cellKey[$i][$lastCell]][0] == 'tablecell_open'
    ||
252.
    $this->tableCalls[$cellKey[$i][$lastCell]][0] ==
    'tableheader_open'
253.                                 ) {
254.                                     if
    ($this->tableCalls[$cellKey[$i][$lastCell]][1][2] >= $lastRow -
    $i) {
255.                                         $spanning_cell = $i;
256.                                         break;
```

```
                                    }
                                }
                            }
                        }
                        if (is_null($spanning_cell)) {
                            // No spanning cell found, so convert this cell to
                            // an empty one to avoid broken tables
                            $this->tableCalls[$key][0] = 'cdata';
                            $this->tableCalls[$key][1][0] = '';
                            break;
                        }
$this->tableCalls[$cellKey[$spanning_cell][$lastCell]][1][2]++;

                        $this->tableCalls[$key-1][1][2] = false;

                        $toDelete[] = $key-1;
                        $toDelete[] = $key;
                        $toDelete[] = $key+1;
                    }
                    break;

                case 'tablerow_close':
                    // Fix broken tables by adding missing cells
                    $moreCalls = array();
                    while (++$lastCell < $this->maxCols) {
                        $moreCalls[] = array('tablecell_open', array(1, null, 1), $call[2]);
                        $moreCalls[] = array('cdata', array(''), $call[2]);
                        $moreCalls[] = array('tablecell_close', array(), $call[2]);
                    }
                    $moreCallsLength = count($moreCalls);
                    if ($moreCallsLength) {
                        array_splice($this->tableCalls, $key, 0, $moreCalls);
                        $key += $moreCallsLength;
                    }

                    if ($this->countTableHeadRows == $lastRow) {
                        array_splice($this->tableCalls, $key+1, 0, array(
                            array('tablethead_close', array(), $call[2])));
                    }
                    break;
            }
        }

```

```
300.         // condense cdata
301.         $cnt = count($this->tableCalls);
302.         for ($key = 0; $key < $cnt; $key++) {
303.             if ($this->tableCalls[$key][0] == 'cdata') {
304.                 $ckey = $key;
305.                 $key++;
306.                 while ($this->tableCalls[$key][0] == 'cdata') {
307.                     $this->tableCalls[$ckey][1][0] .=
    $this->tableCalls[$key][1][0];
308.                     $toDelete[] = $key;
309.                     $key++;
310.                 }
311.                 continue;
312.             }
313.         }
314.
315.         foreach ($toDelete as $delete) {
316.             unset($this->tableCalls[$delete]);
317.         }
318.         $this->tableCalls = array_values($this->tableCalls);
319.     }
320. }
```