UNIVERSIDAD DE CÓRDOBA
ESCUELA POLITÉCNICA SUPERIOR

PROYECTO FIN DE CARRERA
INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

Editor colaborativo de código fuente en lenguaje C

Manual de Código

Alumno: Antonio Jesús González León
Director: Cristóbal Romero Morales

Fecha: 3 de septiembre de 2012

D. Cristóbal Romero Morales
Profesor Titular de la Escuela Politécnica Superior
Departamento de Informática y Análisis Numérico
Universidad de Córdoba

CERTIFICA: Que la memoria titulada *"Editor colaborativo de código fuente en lenguaje C"* ha sido realizada por Antonio Jesús González León bajo mi dirección y constituye su Proyecto de Fin de Carrera de Ingeniería Técnica Informática de Gestión.

En Córdoba, a 3 de septiembre de 2012

D. Cristóbal Romero Morales
Director del proyecto

Índice general

Capítulo 1

Introducción

En este manual se describe el código utilizado en cada uno de los ficheros que compone la aplicación desarrollada para realizar un entorno de edición de código colaborativo en lenguaje C para Moodle. Debido a que este Sistema de Administración de Cursos es mantenido aún tanto en su versión 1.* como en la 2.*, este proyecto puede dividirse en varios subproyectos:

- **Módulo para Moodle 1.x**
- **Módulo para Moodle 2.x**
- **Editor colaborativo**

El editor colaborativo es parte común para ambas versiones, pero debido a que está escrito en otro lenguaje y a su vez es un elemento totalmente independiente, válido para cualquier LMS, también es preferible detallarlo por separado. Así, la estructura de ficheros existente será:

- Módulo para Moodle 1.x

  Front-end: create.php, create_form.php, diff.php, edit.php, edit_form.php, history.php, index.php, log.php, log_form.php, mod_form.php, pagelib.php, version.php, view.php, viewversion.php

  Back-end: lib.php, localib.php

- Módulo para Moodle 2.x

  Front-end: admin.php, comments.php, comments_form.php, create.php, create_form.php, diff.php, edit.php, edit_form.php, files.php, filesedit.php, filesedit_form.php, history.php, index.php, log.php, log_form.php, mod_form.php, pagelib.php, version.php, view.php, viewversion.php

  Back-end: instancecomments.php, lib.php, localib.php, map.php, overridelocks.php, prettyview.php, renderer.php, restoreversion.php, search.php

- Editor colaborativo

  Front-end: wikichat.php, wikieditor.php

  Back-end PHP: editorlib.php, editorlibfn.php, unlock.php, chat/post.php

  Back-end JS: codemirror.js, editor.js, highlight.js, lock.js, parseC.js, select.js, stringstream.js, tokenize.js, undo.js, unlock.js, util.js

  Hojas de estilo: Ccolors.css, chat/style.css

1.1. Descripción del código

Los módulos correspondientes a las versiones de Moodle han sido desarrollados en lenguaje PHP, puesto que es el lenguaje que utiliza el sistema e-learning. PHP es un acrónimo recursivo que significa PHP Hypertext Pre-processor, el cual puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. El lenguaje PHP se encuentra instalado en más de 20 millones de sitios web y en un millón de servidores.

Al tratarse de un lenguaje de programación interpretado o *framework* para HTML (HyperText Markup Language) no necesita de compilación, sino que es el propio servidor el que lo interpreta y crea la salida.

La estructura de ficheros es muy simple, donde tenemos un fichero principal (pagelib.php) que es el encargado de la declaración de todas las clases y posteriormente tenemos un fichero por formulario que es el que hace uso de estas clases.

Con respecto al editor, y puesto que se requería que este actuara en tiempo real, se ha decidido utilizar como lenguaje Javascript. JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Al contrario que PHP, se utiliza principalmente en su forma *client-side*, lo que nos permite mejoras en el interfaz y sobretodo dinamismo en base al cliente. Unido a las

ventajas de este lenguaje es que se ha utilizado su biblioteca *JQuery*, la cual nos permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

Todos los archivos Javascript del proyecto se encuentran dentro del directorio **/js**, lo cual nos permite un acceso más directo a ellos.

Con respecto a los ficheros PHP necesarios para el editor colaborativo, también se ha considerado crear un directorio para cada uno que los represente. Por ello los ficheros para el chat se encuentran dentro del directorio **/chat**, mientras que los ficheros necesarios para el propio editor de código se encuentran en el directorio **/editors**.

Por último, se ha considerado útil almacenar todas las hojas de estilo de modo conjunto, dentro del directorio **/css**.

Capítulo 2

Módulo para Moodle 1.x

2.1. Front-End

2.1.1. create.php

```
<?php

require_once('../../config.php');
require_once(dirname(__FILE__) . '/create_form.php');
require_once($CFG->dirroot . '/mod/wikicode/lib.php');
require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
require_once($CFG->dirroot . '/mod/wikicode/pagelib.php');

// this page accepts two actions: new and create
// 'new' action will display a form contains page title and page format
// selections
// 'create' action will create a new page in db, and redirect to
// page editing page.
$action = optional_param('action', 'new', PARAM_TEXT);
// The title of the new page, can be empty
$title = optional_param('title', '', PARAM_TEXT);
$wid = optional_param('wid', 0, PARAM_INT);
$swid = optional_param('swid', 0, PARAM_INT);
$gid = optional_param('gid', 0, PARAM_INT);
$uid = optional_param('uid', 0, PARAM_INT);

// 'create' action must be submitted by moodle form
// so sesskey must be checked
if ($action == 'create') {
    if (!confirm_sesskey()) {
        print_error('invalidsesskey');
    }
}

```

```
30  if (!empty($swid)) {
31      $subwiki = wikicode_get_subwiki($swid);
32          $wid = $subwiki->wikiid;
33
34      if (!$wiki = wikicode_get_wiki($subwiki->wikiid)) {
35          print_error('invalidwikiid', 'wikicode');
36      }
37
38  } else {
39      $subwiki = wikicode_get_subwiki_by_group($wid, $gid, $uid);
40
41      if (!$wiki = wikicode_get_wiki($wid)) {
42          print_error('invalidwikiid', 'wikicode');
43      }
44
45  }
46
47  if (!$cm = get_coursemodule_from_instance('wikicode', $wiki->id)) {
48      print_error('invalidcoursemoduleid', 'wikicode');
49  }
50
51  $course = get_record('course', 'id', $cm->course);
52
53  require_login($course->id, true, $cm);
54
55  add_to_log($course->id, 'createpage', 'createpage', 'view.php?id=' . $cm->id, $wiki
        ->id);
56
57  $wikipage = new page_wikicode_create($wiki, $subwiki, $cm);
58
59  if (!empty($swid)) {
60      $wikipage->set_gid($subwiki->groupid);
61      $wikipage->set_uid($subwiki->userid);
62      $wikipage->set_swid($swid);
63          $wikipage->set_wid($wid);
64  } else {
65      $wikipage->set_wid($wid);
66      $wikipage->set_gid($gid);
67      $wikipage->set_uid($uid);
68  }
69
70  // set page action, and initialise moodle form
71
72  $wikipage->set_action($action);
73
74  switch ($action) {
75  case 'create':
76      $wikipage->create_page($title);
77      break;
78  case 'new':
79      if ((int)$wiki->forceformat == 1 && !empty($title)) {
80          $wikipage->create_page($title);
81      } else {
82          // create link from moodle navigation block without pagetitle
```

```
83          $wikipage->print_header($cm, $course);
84
85          // new page without page title
86          $wikipage->print_content($title);
87      }
88      print_footer($course);
89      break;
90  }
```

2.1.2. create_form.php

```
1   <?php
2
3   require_once($CFG->libdir.'/formslib.php');
4
5   class mod_wikicode_create_form extends moodleform {
6
7        function definition() {
8           global $CFG;
9           $mform =& $this->_form;
10
11          $formats = $this->_customdata['formats'];
12          $defaultformat = $this->_customdata['defaultformat'];
13          $forceformat = $this->_customdata['forceformat'];
14
15          $mform->addElement('header', 'general', get_string('createpage', 'wikicode'
                ));
16
17          $textoptions = array();
18          if (!empty($this->_customdata['disable_pagetitle'])) {
19              $textoptions = array('readonly'=>'readonly');
20          }
21          $mform->addElement('text', 'pagetitle', get_string('newpagetitle', '
                wikicode'), $textoptions);
22
23          if ($forceformat) {
24              $mform->addElement('hidden', 'pageformat', $defaultformat);
25          } else {
26              $mform->addElement('static', 'format', get_string('format', 'wikicode')
                    );
27              foreach ($formats as $format) {
28                  if ($format == $defaultformat) {
29                      $attr = array('checked'=>'checked');
30                  }else if (!empty($forceformat)){
31                      $attr = array('disabled'=>'disabled');
32                  } else {
33                      $attr = array();
34                  }
35                  $mform->addElement('radio', 'pageformat', '', get_string('format'.
                        $format, 'wikicode'), $format, $attr);
36              }
```

```
37          }
38
39          //hiddens
40          $mform->addElement('hidden', 'action');
41          $mform->setDefault('action', 'create');
42
43          $this->add_action_buttons(false, get_string('createpage', 'wikicode'));
44      }
45  }
```

2.1.3. diff.php

```
1   <?php
2
3   require_once('../../config.php');
4
5   require_once($CFG->dirroot . '/mod/wikicode/lib.php');
6   require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
7   require_once($CFG->dirroot . '/mod/wikicode/pagelib.php');
8
9   require_once($CFG->dirroot . '/mod/wikicode/diff/difflib.php');
10  require_once($CFG->dirroot . '/mod/wikicode/diff/diff_nwiki.php');
11
12  $pageid = required_param('pageid', PARAM_TEXT);
13  $compare = required_param('compare', PARAM_INT);
14  $comparewith = required_param('comparewith', PARAM_INT);
15
16  if (!$page = wikicode_get_page($pageid)) {
17      print_error('incorrectpageid', 'wikicode');
18  }
19
20  if (!$subwiki = wikicode_get_subwiki($page->subwikiid)) {
21      print_error('incorrectsubwikiid', 'wikicode');
22  }
23
24  if (!$wiki = wikicode_get_wiki($subwiki->wikiid)) {
25      print_error('incorrectwikiid', 'wikicode');
26  }
27
28  if (!$cm = get_coursemodule_from_instance('wikicode', $wiki->id)) {
29      print_error('invalidcoursemodule');
30  }
31
32  $course = get_record('course', 'id', $cm->course);
33
34  if ($compare >= $comparewith) {
35      print_error("A page version can only be compared with an older version.");
36  }
37
38  require_login($course->id, true, $cm);
39  add_to_log($course->id, "wikicode", "diff", "diff.php?id=$cm->id", "$wiki->id");
```

```
40
41  $wikipage = new page_wikicode_diff($wiki, $subwiki, $cm);
42
43  $wikipage->set_page($page);
44  $wikipage->set_comparison($compare, $comparewith);
45
46  $wikipage->print_header($cm, $course);
47
48  $wikipage->print_content();
49
50  print_footer($course);
```

2.1.4. edit.php

```
1   <?php
2
3   require_once('../../config.php');
4
5   require_once($CFG->dirroot . '/mod/wikicode/lib.php');
6   require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
7   require_once($CFG->dirroot . '/mod/wikicode/pagelib.php');
8
9
10  $pageid = required_param('pageid', PARAM_INT);
11  $contentformat = optional_param('contentformat', '', PARAM_ALPHA);
12  $option = optional_param('editoption', '', PARAM_TEXT);
13  $section = optional_param('section', "", PARAM_TEXT);
14  $version = optional_param('version', -1, PARAM_INT);
15  $attachments = optional_param('attachments', 0, PARAM_INT);
16  $deleteuploads = optional_param('deleteuploads', 0, PARAM_RAW);
17  $compiled = optional_param('compiled', 0, PARAM_INT);
18
19  $newcontent = '';
20  if (!empty($newcontent) && is_array($newcontent)) {
21      $newcontent = $newcontent['text'];
22  }
23
24  if (!empty($option) && is_array($option)) {
25      $option = $option['editoption'];
26  }
27
28  if (!$page = wikicode_get_page($pageid)) {
29      print_error('incorrectpageid', 'wikicode');
30  }
31
32  if (!$subwiki = wikicode_get_subwiki($page->subwikiid)) {
33      print_error('incorrectsubwikiid', 'wikicode');
34  }
35
36  if (!$wiki = wikicode_get_wiki($subwiki->wikiid)) {
37      print_error('incorrectwikiid', 'wikicode');
```

```
38  }
39
40  if (!$cm = get_coursemodule_from_instance('wikicode', $wiki->id)) {
41      print_error('invalidcoursemodule');
42  }
43
44  $course = get_record('course', 'id', $cm->course);
45
46  if (!empty($section) && !$sectioncontent = wikicode_get_section_page($page,
        $section)) {
47      print_error('invalidsection', 'wikicode');
48  }
49
50  require_login($course, true, $cm);
51
52  $context = get_context_instance(CONTEXT_MODULE, $cm->id);
53  require_capability('mod/wikicode:editpage', $context);
54
55  add_to_log($course->id, 'wikicode', 'edit', "edit.php?id=$cm->id", "$wiki->id");
56
57  if ($option == get_string('save', 'wikicode')) {
58      if (!confirm_sesskey()) {
59          print_error(get_string('invalidsesskey', 'wikicode'));
60      }
61      $wikipage = new page_wikicode_save($wiki, $subwiki, $cm);
62      $wikipage->set_page($page);
63      $wikipage->set_newcontent($newcontent);
64      $wikipage->set_upload(true);
65  } else {
66      if ($option == 'Compile' or $option == 'Download') {
67          if (!confirm_sesskey()) {
68              print_error(get_string('invalidsesskey', 'wikicode'));
69          }
70          $wikipage = new page_wikicode_compile($wiki, $subwiki, $cm);
71          $wikipage->set_page($page);
72                  $wikipage->set_download(($option == 'Download'));
73      }
74          else {
75          if ($option == get_string('cancel')) {
76              //delete lock
77              wikicode_delete_locks($page->id, $USER->id, $section);
78
79              redirect($CFG->wwwroot . '/mod/wikicode/view.php?pageid=' . $pageid);
80          } else {
81              $wikipage = new page_wikicode_edit($wiki, $subwiki, $cm);
82              $wikipage->set_page($page);
83              $wikipage->set_upload($option == get_string('upload', 'wikicode'));
84                          $wikipage->set_compiled($compiled);
85          }
86      }
87
88      if (has_capability('mod/wikicode:overridelock', $context)) {
89          $wikipage->set_overridelock(true);
90      }
```

```
91  }
92
93  if ($version >= 0) {
94      $wikipage->set_versionnumber($version);
95  }
96
97  if (!empty($section)) {
98      $wikipage->set_section($sectioncontent, $section);
99  }
100
101 if (!empty($attachments)) {
102     $wikipage->set_attachments($attachments);
103 }
104
105 if (!empty($deleteuploads)) {
106     $wikipage->set_deleteuploads($deleteuploads);
107 }
108
109 if (!empty($contentformat)) {
110     $wikipage->set_format($contentformat);
111 }
112
113 $wikipage->print_header($cm, $course);
114
115 $wikipage->print_content();
116
117 print_footer($course);
```

2.1.5. edit_form.php

```
1   <?php
2
3   if (!defined('MOODLE_INTERNAL')) {
4       die('Direct access to this script is forbidden.');    ///  It must be included
            from a Moodle page
5   }
6
7   require_once($CFG->dirroot . '/mod/wikicode/editors/wikieditor.php');
8   require_once($CFG->dirroot . '/mod/wikicode/chat/wikichat.php');
9   require_once($CFG->dirroot . '/lib/formslib.php');
10
11  class mod_wikicode_edit_form extends moodleform {
12
13      function definition() {
14                  global $CFG, $USER;
15          $mform =& $this->_form;
16
17          $version = $this->_customdata['version'];
18          $format  = $this->_customdata['format'];
19          $tags    = !isset($this->_customdata['tags'])?"":$this->_customdata['tags'
                ];
```

```
20
21          if ($format != 'html') {
22              $contextid  = $this->_customdata['contextid'];
23              $filearea   = $this->_customdata['filearea'];
24              $fileitemid = $this->_customdata['fileitemid'];
25          }
26
27          if (isset($this->_customdata['pagetitle'])) {
28              $pagetitle = get_string('editingpage', 'wikicode', $this->_customdata['
                    pagetitle']);
29          } else {
30              $pagetitle = get_string('editing', 'wikicode');
31          }
32
33          //editor
34          $mform->addElement('header', 'general', $pagetitle);
35
36          $fieldname = get_string('format' . $format, 'wikicode');
37          if ($format != 'html') {
38              // Use wiki editor
39                          $buttoncommands=array();
40                          $buttoncommands[] =& $mform->createElement('button','
                                editoption','Unlock', array('id' => 'btnunlock', 'class
                                ' => 'btnunlock'));
41                          $buttoncommands[] =& $mform->createElement('button','
                                editoption','Refresh', array('id' => 'btnref', 'class'
                                => 'btnref'));
42                          $buttoncommands[] =& $mform->createElement('submit', '
                                editoption', 'Save', array('id' => 'save'));
43                          $mform->addGroup($buttoncommands, 'editoption', 'Actions:',
                                 '', true);
44              $mform->addElement('wikicodeeditor', 'newcontent', $fieldname, array('
                    cols' => 150, 'rows' => 30, 'Wiki_format' => $format, 'files'=>
                    $files));
45          } else {
46              $mform->addElement('editor', 'newcontent_editor', $fieldname, null,
                    page_wikicode_edit::$attachmentoptions);
47          }
48
49                  //chat
50                  $mform->addElement('header','chat','Chat');
51                  $mform->addElement('wikicodechat', 'wikicodechat', null, array('
                        itemid'=>$fileitemid));
52
53                  //compiler
54                  $mform->addElement('header','compiler', 'Compiler');
55                  $mform->addElement('textarea', 'textCompiler', '', 'wrap="virtual"
                        rows="3" cols="100" readonly="readonly" ');
56
57                  $buttonarray=array();
58                  $buttonarray[] =& $mform->createElement('submit','editoption','
                        Compile', array('id' => 'compile'));
59                  $buttonarray[] =& $mform->createElement('submit','editoption','
                        Download', array('id' => 'compile'));
```

```
60                  $mform->addGroup($buttonarray, 'editoption', 'Options:', '', true);
61
62          //hiddens
63          if ($version >= 0) {
64              $mform->addElement('hidden', 'version');
65              $mform->setDefault('version', $version);
66          }
67
68          $mform->addElement('hidden', 'contentformat');
69          $mform->setDefault('contentformat', $format);
70
71                  $mform->addElement('hidden', 'insert');
72                  $mform->setDefault('insert', 1);
73
74      }
75
76  }
```

2.1.6. history.php

```
1   <?php
2   require_once('../../config.php');
3
4   require_once($CFG->dirroot . '/mod/wikicode/lib.php');
5   require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
6   require_once($CFG->dirroot . '/mod/wikicode/pagelib.php');
7
8   $pageid = required_param('pageid', PARAM_TEXT);
9   $paging = optional_param('page', 0, PARAM_INT);
10  $allversion = optional_param('allversion', 0, PARAM_INT);
11
12  if (!$page = wikicode_get_page($pageid)) {
13      print_error('incorrectpageid', 'wikicode');
14  }
15
16  if (!$subwiki = wikicode_get_subwiki($page->subwikiid)) {
17      print_error('incorrectsubwikiid', 'wikicode');
18  }
19
20  if (!$wiki = wikicode_get_wiki($subwiki->wikiid)) {
21      print_error('incorrectwikiid', 'wikicode');
22  }
23
24  if (!$cm = get_coursemodule_from_instance('wikicode', $wiki->id)) {
25      print_error('invalidcoursemodule');
26  }
27
28  $course = get_record('course', 'id', $cm->course);
29
30  require_login($course->id, true, $cm);
31  $context = get_context_instance(CONTEXT_MODULE, $cm->id);
```

```
32  require_capability('mod/wikicode:viewpage', $context);
33  add_to_log($course->id, 'wikicode', 'history', 'history.php?id=' . $cm->id, $wiki->
        id);
34
35  /// Print the page header
36  $wikipage = new page_wikicode_history($wiki, $subwiki, $cm);
37
38  $wikipage->set_page($page);
39  $wikipage->set_paging($paging);
40  $wikipage->set_allversion($allversion);
41
42  $wikipage->print_header($cm, $course);
43  $wikipage->print_content();
44
45  print_footer($course);
```

2.1.7. index.php

```
1   <?php
2
3       require_once("../../config.php");
4       require_once("lib.php");
5
6       $id = required_param('id', PARAM_INT);   // course
7
8       if (! $course = get_record("course", "id", $id)) {
9           error("Course ID is incorrect");
10      }
11
12      require_course_login($course);
13
14      add_to_log($course->id, "wikicode", "view all", "index.php?id=$course->id", "")
            ;
15
16
17  /// Get all required strings
18
19      $strwikicodes = get_string("modulenameplural", "wikicode");
20      $strwikicode  = get_string("modulename", "wikicode");
21
22
23  /// Print the header
24      $navlinks = array();
25      $navlinks[] = array('name' => $strwikicodes, 'link' => "index.php?id=$course->
            id", 'type' => 'activity');
26      $navigation = build_navigation($navlinks);
27
28      print_header_simple("$strwikicodes", "", $navigation, "", "", true, "", navmenu
            ($course));
29
30  /// Get all the appropriate data
```

```
31
32      if (! $wikicodes = get_all_instances_in_course("wikicode", $course)) {
33          notice(get_string('thereareno', 'moodle', $strwikicodes), "../../course/
                view.php?id=$course->id");
34          die;
35      }
36
37  /// Print the list of instances (your module will probably extend this)
38
39      $timenow = time();
40      $strname  = 'Nombre';
41      $strsummary = get_string('summary');
42      $strtype = 'Tipo';
43      $strlastmodified = 'Creacion';
44      $strweek  = get_string('week');
45      $strtopic  = get_string('topic');
46
47      if ($course->format == "weeks") {
48          $table->head  = array ($strweek, $strname, $strsummary, $strtype,
                $strlastmodified);
49          $table->align = array ('CENTER', 'LEFT', 'LEFT', 'LEFT', 'LEFT');
50      } else if ($course->format == "topics") {
51          $table->head  = array ($strtopic, $strname, $strsummary, $strtype,
                $strlastmodified);
52          $table->align = array ('CENTER', 'LEFT', 'LEFT', 'LEFT', 'LEFT');
53      } else {
54          $table->head  = array ($strname, $strsummary, $strtype, $strlastmodified);
55          $table->align = array ('LEFT', 'LEFT', 'LEFT', 'LEFT');
56      }
57
58      foreach ($wikicodes as $wikicode) {
59          if (!$wikicode->visible) {
60              //Show dimmed if the mod is hidden
61              $link = '<a class="dimmed" href="view.php?id='.$wikicode->coursemodule.
                    '">'.format_string($wikicode->name,true).'</a>';
62          } else {
63              //Show normal if the mod is visible
64              $link = '<a href="view.php?id='.$wikicode->coursemodule.'">'.
                    format_string($wikicode->name,true).'</a>';
65          }
66
67          $timmod = '<span class="smallinfo">'.userdate($wikicode->timemodified).'</
                span>';
68          $summary = '<div class="smallinfo">'.$wikicode->firstpagetitle.'</div>';
69
70          $site = get_site();
71
72          $wtype = '<span class="smallinfo">'.$wikicode->wikimode.'</span>';
73
74          if ($course->format == "weeks" or $course->format == "topics") {
75              $table->data[] = array ($wikicode->section, $link, $summary, $wtype,
                    $timmod);
76          } else {
77              $table->data[] = array ($link, $summary, $wtype, $timmod);
```

```
78          }
79      }
80
81      echo "<br />";
82
83      print_table($table);
84
85  /// Finish the page
86
87      print_footer($course);
```

2.1.8. log.php

```
1   <?php
2
3   require_once('../../config.php');
4
5   require_once($CFG->dirroot . '/mod/wikicode/lib.php');
6   require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
7   require_once($CFG->dirroot . '/mod/wikicode/pagelib.php');
8
9   $pageid = required_param('pageid', PARAM_INT);
10  $section = optional_param('section', "", PARAM_TEXT);
11  $version = optional_param('version', -1, PARAM_INT);
12
13  if (!$page = wikicode_get_page($pageid)) {
14      print_error('incorrectpageid', 'wikicode');
15  }
16
17  if (!$subwiki = wikicode_get_subwiki($page->subwikiid)) {
18      print_error('incorrectsubwikiid', 'wikicode');
19  }
20
21  if (!$wiki = wikicode_get_wiki($subwiki->wikiid)) {
22      print_error('incorrectwikiid', 'wikicode');
23  }
24
25  if (!$cm = get_coursemodule_from_instance('wikicode', $wiki->id)) {
26      print_error('invalidcoursemodule');
27  }
28
29  $course = get_record('course', 'id', $cm->course);
30
31  if (!empty($section) && !$sectioncontent = wikicode_get_section_page($page,
        $section)) {
32      print_error('invalidsection', 'wikicode');
33  }
34
35  require_login($course, true, $cm);
36
37  $context = get_context_instance(CONTEXT_MODULE, $cm->id);
```

```
38  require_capability('mod/wikicode:editpage', $context);
39
40  add_to_log($course->id, 'wikicode', 'log', "log.php?id=$cm->id", "$wiki->id");
41
42  $wikipage = new page_wikicode_log($wiki, $subwiki, $cm);
43
44  $wikipage->set_page($page);
45
46  $wikipage->print_header($cm, $course);
47
48  $wikipage->print_content();
49
50  print_footer($course);
```

2.1.9. log_form.php

```
1   <?php
2
3   if (!defined('MOODLE_INTERNAL')) {
4       die('Direct access to this script is forbidden.');    ///  It must be included
            from a Moodle page
5   }
6
7   require_once ($CFG->dirroot.'/lib/formslib.php');
8
9   class mod_wikicode_log_form extends moodleform {
10
11      function definition() {
12          global $CFG, $USER, $DB;
13
14          $mform =& $this->_form;
15
16          $version = $this->_customdata['version'];
17          $format  = $this->_customdata['format'];
18          $tags    = !isset($this->_customdata['tags'])?"":$this->_customdata['tags'
                ];
19
20          if ($format != 'html') {
21              $contextid  = $this->_customdata['contextid'];
22              $filearea   = $this->_customdata['filearea'];
23              $fileitemid = $this->_customdata['fileitemid'];
24          }
25
26          if (isset($this->_customdata['pagetitle'])) {
27              $pagetitle = get_string('logpage', 'wikicode', $this->_customdata['
                    pagetitle']);
28          } else {
29              $pagetitle = get_string('loging', 'wikicode');
30          }
31
32                  //Time
```

```
33                  $time = $this->_customdata['page']->timeendedit - $this->
                        _customdata['page']->timestartedit;
34                  $seconds = $time % 60;
35                  $time = ($time - $seconds) / 60;
36                  $minutes = $time % 60;
37                  $hours = ($time - $minutes) / 60;
38
39                  //Stats
40                  $attr = array('size' => '75', 'readonly' => 1);
41                  $mform->addElement('header','stats', 'Stats');
42                  $attr['value'] = $hours . " hours, " . $minutes . " minutes, " .
                        $seconds . " seconds";
43                  $mform->addElement('text', 'timeedit', 'Editing Time', $attr);
44                  //$mform->addHelpButton('timeedit', 'timeedit', 'wikicode');
45                  $attr['value'] = $this->_customdata['page']->errorcompile;
46                  $mform->addElement('text', 'errorscompilation', 'Compilation Errors
                        ', $attr);
47                  //$mform->addHelpButton('errorscompilation', 'errorscompilation', '
                        wikicode');
48
49
50          $mform->addElement('hidden', 'contentformat');
51          $mform->setDefault('contentformat', $format);
52
53                  $mform->addElement('hidden', 'insert');
54                  $mform->setDefault('insert', 1);
55
56      }
57
58  }
```

2.1.10. mod_form.php

```
1   <?php
2
3   if (!defined('MOODLE_INTERNAL')) {
4       die('Direct access to this script is forbidden.');    ///  It must be included
            from a Moodle page
5   }
6
7   require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
8   require_once($CFG->libdir.'/formslib.php');
9   require_once($CFG->libdir.'/datalib.php');
10  require_once ($CFG->dirroot.'/course/moodleform_mod.php');
11
12  class mod_wikicode_mod_form extends moodleform_mod {
13
14      function definition() {
15          global $COURSE;
16          $mform =& $this->_form;
17
```

```
18          //
                ----------------------------------------------------------------------------

19          /// Adding the "general" fieldset, where all the common settings are showed
20          $mform->addElement('header', 'general', get_string('general', 'form'));
21
22          /// Adding the standard "name" field
23          $mform->addElement('text', 'name', "Wiki name", array('size' => '64'));
24          $mform->setType('name', PARAM_TEXT);
25          $mform->addRule('name', null, 'required', null, 'client');
26          //
                ----------------------------------------------------------------------------

27          /// Adding the rest of wiki settings, spreeading all them into this
                fieldset
28          /// or adding more fieldsets ('header' elements) if needed for better logic
29
30          $mform->addElement('header', 'wikifieldset', "Wiki Settings");
31
32          $attr = array('size' => '20');
33          if (!empty($this->_instance)) {
34              $attr['disabled'] = 'disabled';
35          } else {
36              $attr['value'] = "First page name";
37          }
38
39
40          $mform->addElement('text', 'firstpagetitle', "First page name", $attr);
41
42          if (empty($this->_instance)) {
43              $mform->addRule('firstpagetitle', null, 'required', null, 'client');
44          }
45
46                  $attr = array('size' => '180');
47
48                  $gccvalue = wikicode_get_gccpath();
49                  if ($gccvalue->gccpath != "") {
50                      $attr['value'] = $gccvalue->gccpath;
51                  } else {
52                          $attr['value'] = 'gcc';
53                  }
54
55                  $mform->addElement('text', 'gccpath', 'Unix Compiler Path', $attr);
56
57                  $mingwvalue = wikicode_get_mingwpath();
58                  if ($mingwvalue->mingwpath != "") {
59                      $attr['value'] = $mingwvalue->mingwpath;
60                  } else {
61                          $attr['value'] = 'mingw32-gcc';
62                  }
63
64                  $mform->addElement('text', 'mingwpath', 'Windows Compiler Path',
                        $attr);
65
```

```
66          $wikimodeoptions = array ('collaborative' => "Collaborative wiki", '
                individual' => "Individual wiki");
67          // don't allow to change wiki type once is set
68          $wikitype_attr = array();
69          if (!empty($this->_instance)) {
70              $wikitype_attr['disabled'] = 'disabled';
71          }
72          $mform->addElement('select', 'wikimode', "Wiki mode", $wikimodeoptions,
                $wikitype_attr);
73
74          $formats = wikicode_get_formats();
75          $editoroptions = array();
76          foreach ($formats as $format) {
77              $editoroptions[$format] = get_string($format, 'wikicode');
78          }
79          $mform->addElement('select', 'defaultformat', get_string('defaultformat', '
                wikicode'), $editoroptions);
80          $mform->addElement('checkbox', 'forceformat', get_string('forceformat', '
                wikicode'));
81
82          //
                ---------------------------------------------------------------------------

83          // add standard elements, common to all modules
84          $this->standard_coursemodule_elements();
85          //
                ---------------------------------------------------------------------------

86          // add standard buttons, common to all modules
87          $this->add_action_buttons();
88
89      }
90  }
```

2.1.11. pagelib.php

```
1   <?php
2
3   require_once($CFG->dirroot . '/mod/wikicode/edit_form.php');
4   require_once($CFG->dirroot . '/mod/wikicode/log_form.php');
5   require_once($CFG->dirroot . '/tag/lib.php');
6   require_once($CFG->dirroot . '/lib/formslib.php');
7
8
9   /**
10   * Class page_wikicode contains the common code between all pages
11   *
12   * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
13   */
14  abstract class page_wikicode {
15
```

```
16      /**
17       * @var object Current subwiki
18       */
19      protected $subwiki;
20
21      /**
22       * @var int Current page
23       */
24      protected $page;
25
26      /**
27       * @var string Current page title
28       */
29      protected $title;
30
31      /**
32       * @var int Current group ID
33       */
34      protected $gid;
35
36      /**
37       * @var object module context object
38       */
39      protected $modcontext;
40
41      /**
42       * @var int Current user ID
43       */
44      protected $uid;
45      /**
46       * @var array The tabs set used in wiki module
47       */
48      protected $tabs = array('view' => 'view', 'edit' => 'edit', 'history' => '
            history', 'log' => 'log', 'admin' => 'admin');
49      /**
50       * @var array tabs options
51       */
52      protected $tabs_options = array();
53      /**
54       * @var object wiki renderer
55       */
56      protected $wikioutput;
57
58      /**
59       * page_wikicode constructor
60       *
61       * @param $wiki. Current wiki
62       * @param $subwiki. Current subwiki.
63       * @param $cm. Current course_module.
64       */
65      function __construct($wiki, $subwiki, $cm) {
66
67          global $PAGE, $CFG;
68          $this->subwiki = $subwiki;
```

```
69          $this->modcontext = get_context_instance(CONTEXT_MODULE, $cm->id);
70
71      }
72
73      /**
74       * This method prints the top of the page.
75       */
76      function print_header($cm, $course) {
77          global $OUTPUT, $PAGE, $CFG, $USER, $SESSION;
78
79          if (isset($SESSION->wikipreviousurl) && is_array($SESSION->wikipreviousurl)
                ) {
80              $this->process_session_url();
81          }
82          $this->set_session_url();
83
84          /// Print the page header
85
86          $strwikis = get_string("modulenameplural", "wikicode");
87          $strwiki  = get_string("modulename", "wikicode");
88
89          $navlinks = array();
90                  /// Add page name if not main page
91
92          $navlinks[] = array('name' => format_string($this->title), 'link' => '', '
                type' => 'title');
93
94
95          $navigation = build_navigation($navlinks, $cm);
96          print_header_simple($this->title, "", $navigation,
97                  "", "", $cacheme, update_module_button($cm->id, $course->id,
                        $strwiki),
98                  navmenu($course, $cm));
99                  print_heading($this->title, 'center', 3);
100     }
101
102     /**
103      * Protected method to print current page title.
104      */
105     protected function print_pagetitle() {
106         global $OUTPUT;
107         $html = '';
108         $html .= $OUTPUT->container_start();
109         $html .= $OUTPUT->heading(format_string($this->title), 2, '
                wikicode_headingtitle');
110         $html .= $OUTPUT->container_end();
111         echo $html;
112     }
113
114     /**
115      * Setup page tabs, if options is empty, will set up active tab automatically
116      * @param array $options, tabs options
117      */
118     protected function setup_tabs($options = array()) {
```

```
119             global $CFG, $PAGE;
120             $groupmode = groups_get_activity_groupmode($PAGE->cm);
121 
122             if (empty($CFG->usecomments) || !has_capability('mod/wikicode:viewcomment',
                     $PAGE->context)){
123                 unset($this->tabs['comments']);
124             }
125 
126             if (!has_capability('mod/wikicode:editpage', $PAGE->context)){
127                 unset($this->tabs['edit']);
128             }
129 
130             if ($groupmode and $groupmode == VISIBLEGROUPS) {
131                 $currentgroup = groups_get_activity_group($PAGE->cm);
132                 $manage = has_capability('mod/wikicode:managewiki', $PAGE->cm->context)
                        ;
133                 $edit = has_capability('mod/wikicode:editpage', $PAGE->context);
134                 if (!$manage and !($edit and groups_is_member($currentgroup))) {
135                     unset($this->tabs['edit']);
136                 }
137             } else {
138                 if (!has_capability('mod/wikicode:editpage', $PAGE->context)) {
139                     unset($this->tabs['edit']);
140                 }
141             }
142 
143 
144             if (empty($options)) {
145                 $this->tabs_options = array('activetab' => substr(get_class($this), 10)
                        );
146             } else {
147                 $this->tabs_options = $options;
148             }
149 
150         }
151 
152         /**
153          * This method must be overwritten to print the page content.
154          */
155         function print_content() {
156             throw new coding_exception('Page wiki class does not implement method
                    print_content()');
157         }
158 
159         /**
160          * Method to set the current page
161          *
162          * @param object $page Current page
163          */
164         function set_page($page) {
165             global $PAGE;
166 
167             $this->page = $page;
168             $this->title = $page->title;
```

```
169     }
170
171     /**
172      * Method to set the current page title.
173      * This method must be called when the current page is not created yet.
174      * @param string $title Current page title.
175      */
176     function set_title($title) {
177         global $PAGE;
178
179         $this->page = null;
180         $this->title = $title;
181     }
182
183     /**
184      * Method to set current group id
185      * @param int $gid Current group id
186      */
187     function set_gid($gid) {
188         $this->gid = $gid;
189     }
190
191     /**
192      * Method to set current user id
193      * @param int $uid Current user id
194      */
195     function set_uid($uid) {
196         $this->uid = $uid;
197     }
198
199     /**
200      * Method to set the URL of the page.
201      * This method must be overwritten by every type of page.
202      */
203     protected function set_url() {
204         throw new coding_exception('Page wiki class does not implement method
                set_url()');
205     }
206
207     /**
208      * Protected method to create the common items of the navbar in every page type
             .
209      */
210     protected function create_navbar() {
211     }
212
213     /**
214      * This method print the footer of the page.
215      */
216     function print_footer() {
217         global $OUTPUT;
218         echo $OUTPUT->footer();
219     }
220
```

```
221     protected function process_session_url() {
222         global $USER, $SESSION;
223 
224         //delete locks if edit
225         $url = $SESSION->wikipreviousurl;
226         switch ($url['page']) {
227         case 'edit':
228             wikicode_delete_locks($url['params']['pageid'], $USER->id, $url['params
                ']['section'], false);
229             break;
230         }
231     }
232 
233     protected function set_session_url() {
234         global $SESSION;
235         unset($SESSION->wikipreviousurl);
236     }
237 
238         function print_tab() {
239                 $tabs = array('view', 'edit','history','log');
240 
241         $tabrows = array();
242         $row  = array();
243         $currenttab = '';
244         foreach ($tabs as $tab) {
245             $tabname = $tab;
246             $row[] = new tabobject($tabname, $ewbase.$tab.'.php?pageid='.$this->
                page->id, $tabname);
247             if ($ewiki_action == "$tab" or in_array($page, $specialpages)) {
248                 $currenttab = $tabname;
249             }
250         }
251         $tabrows[] = $row;
252 
253         print_tabs($tabrows, $currenttab);
254         }
255 
256 }
257 
258 /**
259  * View a wiki page
260  *
261  * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
262  */
263 class page_wikicode_view extends page_wikicode {
264     /**
265      * @var int the coursemodule id
266      */
267     private $coursemodule;
268 
269     function print_header($cm, $course) {
270 
271         parent::print_header($cm, $course);
272 
```

```
273         parent::print_tab();
274 
275                 $wiki = wikicode_get_wikicode_from_pageid($this->page->id);
276 
277         $this->wikicode_print_subwiki_selector($wiki, $this->subwiki, $this->page,
                'view');
278     }
279 
280         function wikicode_print_subwiki_selector($wiki, $subwiki, $page, $pagetype
                = 'view') {
281         global $CFG, $USER;
282         switch ($pagetype) {
283         case 'files':
284             $baseurl = $CFG->wwwroot . '/mod/wikicode/files.php';
285             break;
286         case 'view':
287         default:
288             $baseurl = $CFG->wwwroot . '/mod/wikicode/view.php';
289             break;
290         }
291 
292         $cm = get_coursemodule_from_instance('wikicode', $wiki->id);
293         $context = get_context_instance(CONTEXT_MODULE, $cm->id);
294         // @TODO: A plenty of duplicated code below this lines.
295         // Create private functions.
296         switch (groups_get_activity_groupmode($cm)) {
297         case NOGROUPS:
298             if ($wiki->wikimode == 'collaborative') {
299                 // No need to print anything
300                 return;
301             } else if ($wiki->wikimode == 'individual') {
302                 // We have private wikis here
303 
304                 $view = has_capability('mod/wikicode:viewpage', $context);
305                 $manage = has_capability('mod/wikicode:managewiki', $context);
306 
307                 // Only people with these capabilities can view all wikis
308                 if ($view && $manage) {
309                     // @TODO: Print here a combo that contains all users.
310                     $users = get_enrolled_users($context);
311                     $options = array();
312                     foreach ($users as $user) {
313                         $options[$user->id] = fullname($user);
314                     }
315 
316                     print_container_start();
317                     if ($pagetype == 'files') {
318                         $params['pageid'] = $page->id;
319                     }
320                     $baseurl = $baserurl . '?wid=' . $wiki->id . '&title=' . $page
                            ->title;
321                     $name = 'uid';
322                     $selected = $subwiki->userid;
323                     print_container_end();
```

```
324                 }
325                 return;
326             } else {
327                 // error
328                 return;
329             }
330         case SEPARATEGROUPS:
331             if ($wiki->wikimode == 'collaborative') {
332                 // We need to print a select to choose a course group
333 
334                 $params = array('wid'=>$wiki->id, 'title'=>$page->title);
335                 if ($pagetype == 'files') {
336                     $params['pageid'] = $page->id;
337                 }
338                 $baseurl = $baserurl . '?wid=' . $wiki->id . '&title=' . $page->
                    title;
339 
340                 print_container_start(true);
341                                 print_simple_box_start('center','70%','','20');
342                 groups_print_activity_menu($cm, $baseurl, false, true);
343                                 print_simple_box_end();
344                 print_container_end();
345                 return;
346             } else if ($wiki->wikimode == 'individual') {
347                 //  @TODO: Print here a combo that contains all users of that
                    subwiki.
348                 $view = has_capability('mod/wikicode:viewpage', $context);
349                 $manage = has_capability('mod/wikicode:managewiki', $context);
350 
351                 // Only people with these capabilities can view all wikis
352                 if ($view && $manage) {
353                     $users = get_enrolled_users($context);
354                     $options = array();
355                     foreach ($users as $user) {
356                         $groups = groups_get_all_groups($cm->course, $user->id);
357                         if (!empty($groups)) {
358                             foreach ($groups as $group) {
359                                 $options[$group->id][$group->name][$group->id . '-'
                                        . $user->id] = fullname($user);
360                             }
361                         } else {
362                             $name = get_string('notingroup', 'wikicode');
363                             $options[0][$name]['0' . '-' . $user->id] = fullname(
                                    $user);
364                         }
365                     }
366                 } else {
367                     $group = groups_get_group($subwiki->groupid);
368                     $users = groups_get_members($subwiki->groupid);
369                     foreach ($users as $user) {
370                         $options[$group->id][$group->name][$group->id . '-' . $user
                                ->id] = fullname($user);
371                     }
372                 }
```

```
373                 print_container_start();
374                 $params = array('wid' => $wiki->id, 'title' => $page->title);
375                 if ($pagetype == 'files') {
376                     $params['pageid'] = $page->id;
377                 }
378                 $baseurl = $baserurl . '?wid=' . $wiki->id . '&title=' . $page->
                        title;
379                 $name = 'groupanduser';
380                 $selected = $subwiki->groupid . '-' . $subwiki->userid;
381                 print_container_end();
382
383                 return;
384
385             } else {
386                 // error
387                 return;
388             }
389         CASE VISIBLEGROUPS:
390             if ($wiki->wikimode == 'collaborative') {
391                 // We need to print a select to choose a course group
392                 $params = array('wid'=>$wiki->id, 'title'=>urlencode($page->title))
                        ;
393                 if ($pagetype == 'files') {
394                     $params['pageid'] = $page->id;
395                 }
396                 $baseurl = $baserurl . '?wid=' . $wiki->id . '&title=' . $page->
                        title;
397
398                 print_container_start();
399                 groups_print_activity_menu($cm, $baseurl);
400                 print_container_end();
401                 return;
402
403             } else if ($wiki->wikimode == 'individual') {
404                 $users = get_enrolled_users($context);
405                 $options = array();
406                 foreach ($users as $user) {
407                     $groups = groups_get_all_groups($cm->course, $user->id);
408                     if (!empty($groups)) {
409                         foreach ($groups as $group) {
410                             $options[$group->id][$group->name][$group->id . '-' .
                                    $user->id] = fullname($user);
411                         }
412                     } else {
413                         $name = get_string('notingroup', 'wikicode');
414                         $options[0][$name]['0' . '-' . $user->id] = fullname($user)
                                ;
415                     }
416                 }
417
418                 print_container_start();
419                 $params = array('wid' => $wiki->id, 'title' => $page->title);
420                 if ($pagetype == 'files') {
421                     $params['pageid'] = $page->id;
```

```
422                 }
423                 $baseurl = $baserurl . '?wid=' . $wiki->id . '&title=' . $page->
                        title;
424                 $name = 'groupanduser';
425                 $selected = $subwiki->groupid . '-' . $subwiki->userid;
426                 print_container_end();
427 
428                 return;
429 
430             } else {
431                 // error
432                 return;
433             }
434         default:
435             // error
436             return;
437 
438         }
439 
440     }
441 
442     function print_content() {
443         global $PAGE, $CFG;
444 
445         if (wikicode_user_can_view($this->subwiki)) {
446 
447             if (!empty($this->page)) {
448                 wikicode_print_page_content($this->page, $this->modcontext, $this->
                        subwiki->id);
449                 //$wiki = $PAGE->activityrecord;
450             } else {
451                 print_string('nocontent', 'wikicode');
452                 // TODO: fix this part
453                 $swid = 0;
454                 if (!empty($this->subwiki)) {
455                     $swid = $this->subwiki->id;
456                 }
457             }
458         } else {
459             // @TODO: Tranlate it
460             echo "You can not view this page";
461         }
462     }
463 
464     function set_url() {
465         global $PAGE, $CFG;
466         $params = array();
467 
468         if (isset($this->coursemodule)) {
469             $params['id'] = $this->coursemodule;
470         } else if (!empty($this->page) and $this->page != null) {
471             $params['pageid'] = $this->page->id;
472         } else if (!empty($this->gid)) {
473             $params['wid'] = $PAGE->cm->instance;
```

```
474             $params['group'] = $this->gid;
475         } else if (!empty($this->title)) {
476             $params['swid'] = $this->subwiki->id;
477             $params['title'] = $this->title;
478         } else {
479             print_error(get_string('invalidparameters', 'wikicode'));
480         }
481 
482         $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/view.php', $params);
483     }
484 
485     function set_coursemodule($id) {
486         $this->coursemodule = $id;
487     }
488 
489     protected function create_navbar() {
490         global $PAGE, $CFG;
491 
492         $PAGE->navbar->add(format_string($this->title));
493         $PAGE->navbar->add(get_string('view', 'wikicode'));
494     }
495 }
496 
497 /**
498  * Wiki page editing page
499  *
500  * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
501  */
502 class page_wikicode_edit extends page_wikicode {
503 
504     public static $attachmentoptions;
505 
506     protected $sectioncontent;
507     /** @var string the section name needed to be edited */
508     protected $section;
509     protected $overridelock = false;
510     protected $versionnumber = -1;
511     protected $upload = false;
512     protected $attachments = 0;
513     protected $deleteuploads = array();
514     protected $format;
515         protected $compiled = 0;
516 
517     function __construct($wiki, $subwiki, $cm) {
518         global $CFG, $PAGE;
519         parent::__construct($wiki, $subwiki, $cm);
520     }
521 
522     protected function print_pagetitle() {
523         global $OUTPUT;
524 
525         $title = $this->title;
526         if (isset($this->section)) {
527             $title .= ' : ' . $this->section;
```

```
528         }
529                 echo "<script src=\"js/codemirror.js\" type=\"text/javascript\"></
                        script>";
530     }
531
532     function print_header($cm, $course) {
533         global $OUTPUT, $PAGE;
534
535         parent::print_header($cm, $course);
536
537         $this->print_pagetitle();
538                 parent::print_tab();
539     }
540
541     function print_content() {
542         global $PAGE;
543
544         if (wikicode_user_can_edit($this->subwiki)) {
545             $this->print_edit(null, $compile);
546         } else {
547             // @TODO: Translate it
548             echo "You can not edit this page";
549         }
550     }
551
552     protected function set_url() {
553         global $PAGE, $CFG;
554
555         $params = array('pageid' => $this->page->id);
556
557         if (isset($this->section)) {
558             $params['section'] = $this->section;
559         }
560
561         $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/edit.php', $params);
562     }
563
564     protected function set_session_url() {
565         global $SESSION;
566
567         $SESSION->wikipreviousurl = array('page' => 'edit', 'params' => array('
                pageid' => $this->page->id, 'section' => $this->section));
568     }
569
570     protected function process_session_url() {
571     }
572
573     function set_section($sectioncontent, $section) {
574         $this->sectioncontent = $sectioncontent;
575         $this->section = $section;
576     }
577
578     public function set_versionnumber($versionnumber) {
579         $this->versionnumber = $versionnumber;
```

```
580     }
581
582     public function set_overridelock($override) {
583         $this->overridelock = $override;
584     }
585
586     function set_format($format) {
587         $this->format = $format;
588     }
589
590     public function set_upload($upload) {
591         $this->upload = $upload;
592     }
593
594     public function set_attachments($attachments) {
595         $this->attachments = $attachments;
596     }
597
598     public function set_deleteuploads($deleteuploads) {
599         $this->deleteuploads = $deleteuploads;
600     }
601
602         public function set_compiled($compiled) {
603                 $this->compiled = $compiled;
604         }
605
606     protected function create_navbar() {
607         global $PAGE, $CFG;
608
609         parent::create_navbar();
610
611         $PAGE->navbar->add(get_string('edit', 'wikicode'));
612     }
613
614     protected function check_locks() {
615         global $OUTPUT, $USER, $CFG;
616
617         return true;
618     }
619
620     protected function print_edit($content = null) {
621         global $CFG, $OUTPUT, $USER, $PAGE;
622
623         if (!$this->check_locks()) {
624             return;
625         }
626
627         //delete old locks (> 1 hour)
628         wikicode_delete_old_locks();
629
630         $version = wikicode_get_current_version($this->page->id);
631                 $page = wikicode_get_page($this->page->id);
632
633         $format = $version->contentformat;
```

```
634
635         if ($content == null) {
636             if (empty($this->section)) {
637                 $content = $version->content;
638             } else {
639                 $content = $this->sectioncontent;
640             }
641         }
642
643         $versionnumber = $version->version;
644         if ($this->versionnumber >= 0) {
645             if ($version->version != $this->versionnumber) {
646                 //print $OUTPUT->box(get_string('wrongversionlock', 'wikicode'), '
                        errorbox');
647                 $versionnumber = $this->versionnumber;
648             }
649         }
650
651
652
653         $url = $CFG->wwwroot . '/mod/wikicode/edit.php?pageid=' . $this->page->id;
654         if (!empty($this->section)) {
655             $url .= "&section=" . urlencode($this->section);
656         }
657
658         $params = array('attachmentoptions' => page_wikicode_edit::
                $attachmentoptions, 'format' => $version->contentformat, 'version' =>
                $versionnumber, 'pagetitle'=>$this->page->title);
659
660         $data = new StdClass();
661         $data->newcontent = wikicode_remove_tags_owner($content);
662         $data->version = $versionnumber;
663         $data->format = $format;
664
665         switch ($format) {
666         case 'html':
667             $data->newcontentformat = FORMAT_HTML;
668             // Append editor context to editor options, giving preference to
                    existing context.
669             page_wikicode_edit::$attachmentoptions = array_merge(array('context' =>
                     $this->modcontext), page_wikicode_edit::$attachmentoptions);
670             $data = file_prepare_standard_editor($data, 'newcontent',
                    page_wikicode_edit::$attachmentoptions, $this->modcontext, '
                    mod_wikicode', 'attachments', $this->subwiki->id);
671             break;
672         default:
673             break;
674             }
675
676         if ($version->contentformat != 'html') {
677             $params['fileitemid'] = $this->subwiki->id;
678             $params['contextid']  = $this->modcontext->id;
679             $params['component']  = 'mod_wikicode';
680             $params['filearea']   = 'attachments';
```

```
681         }
682
683         if (!empty($CFG->usetags)) {
684             $params['tags'] = tag_get_tags_csv('wikicode_pages', $this->page->id,
                    TAG_RETURN_TEXT);
685         }
686
687         $form = new mod_wikicode_edit_form($url, $params);
688
689         if ($formdata = $form->get_data()) {
690             if (!empty($CFG->usetags)) {
691                 $data->tags = $formdata->tags;
692             }
693         } else {
694             if (!empty($CFG->usetags)) {
695                 $data->tags = tag_get_tags_array('wikicode', $this->page->id);
696             }
697         }
698
699                 if ( $this->compiled == 1 ) {
700                         $data->newcontent = wikicode_remove_tags_owner($page->
                                cachedcompile);
701                         $data->textCompiler = $page->cachedgcc;
702                 }
703
704         $form->set_data($data);
705         $form->display();
706     }
707
708 }
709
710 /**
711  * Wiki page editing page
712  *
713  * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
714  */
715 class page_wikicode_log extends page_wikicode {
716
717         protected $sectioncontent;
718     /** @var string the section name needed to be edited */
719     protected $section;
720     protected $overridelock = false;
721     protected $versionnumber = -1;
722
723     function __construct($wiki, $subwiki, $cm) {
724         global $CFG, $PAGE;
725         parent::__construct($wiki, $subwiki, $cm);
726     }
727
728     protected function print_pagetitle() {
729         global $OUTPUT;
730
731         $title = $this->title;
732         if (isset($this->section)) {
```

```
733             $title .= ' : ' . $this->section;
734         }
735         echo $OUTPUT->container_start('wikicode_clear');
736         echo $OUTPUT->heading(format_string($title), 2, 'wikicode_headingtitle');
737         echo $OUTPUT->container_end();
738     }
739
740     function print_header($cm, $course) {
741         global $OUTPUT, $PAGE;
742
743         parent::print_header($cm, $course);
744
745                 parent::print_tab();
746     }
747
748     function print_content() {
749         global $PAGE;
750
751         if (wikicode_user_can_edit($this->subwiki)) {
752             $this->print_log();
753         } else {
754             // @TODO: Translate it
755             echo "You can not edit this page";
756         }
757     }
758
759     protected function set_url() {
760         global $PAGE, $CFG;
761
762         $params = array('pageid' => $this->page->id);
763
764         if (isset($this->section)) {
765             $params['section'] = $this->section;
766         }
767
768         $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/log.php', $params);
769     }
770
771     protected function set_session_url() {
772         global $SESSION;
773
774         $SESSION->wikipreviousurl = array('page' => 'log', 'params' => array('
            pageid' => $this->page->id, 'section' => $this->section));
775     }
776
777     protected function process_session_url() {
778     }
779
780     protected function create_navbar() {
781         global $PAGE, $CFG;
782
783         parent::create_navbar();
784
785         $PAGE->navbar->add(get_string('log', 'wikicode'));
```

```
786     }
787
788     protected function check_locks() {
789         global $OUTPUT, $USER, $CFG;
790
791         return true;
792     }
793
794     protected function print_log() {
795         global $CFG, $OUTPUT, $USER, $PAGE;
796
797         if (!$this->check_locks()) {
798             return;
799         }
800
801         //delete old locks (> 1 hour)
802         wikicode_delete_old_locks();
803
804         $version = wikicode_get_current_version($this->page->id);
805                 $page = wikicode_get_page($this->page->id);
806
807         $format = $version->contentformat;
808
809         $params = array('attachmentoptions' => page_wikicode_edit::
                $attachmentoptions, 'format' => $version->contentformat, 'version' =>
                $versionnumber, 'pagetitle'=>$this->page->title);
810
811         $data = new StdClass();
812                 $params['page'] = $this->page;
813
814         $form = new mod_wikicode_log_form($url, $params);
815
816         $form->set_data($data);
817         $form->display();
818     }
819
820 }
821
822 /**
823  * Class that models the behavior of wiki's view comments page
824  *
825  * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
826  */
827 class page_wikicode_comments extends page_wikicode {
828
829     function print_header() {
830
831         parent::print_header();
832
833         $this->print_pagetitle();
834
835     }
836
837     function print_content() {
```

```
838         global $CFG, $OUTPUT, $USER, $PAGE;
839         require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
840 
841         $page = $this->page;
842         $subwiki = $this->subwiki;
843         $wiki = $PAGE->activityrecord;
844         list($context, $course, $cm) = get_context_info_array($this->modcontext->id
                ) ;
845 
846         require_capability('mod/wikicode:viewcomment', $this->modcontext, NULL,
                true, 'noviewcommentpermission', 'wikicode');
847 
848         $comments = wikicode_get_comments($this->modcontext->id, $page->id);
849 
850         if (has_capability('mod/wikicode:editcomment', $this->modcontext)) {
851             echo '<div class="midpad"><a href="' . $CFG->wwwroot . '/mod/wikicode/
                    editcomments.php?action=add&amp;pageid=' . $page->id . '">' .
                    get_string('addcomment', 'wikicode') . '</a></div>';
852         }
853 
854         $options = array('swid' => $this->page->subwikiid, 'pageid' => $page->id);
855         $version = wikicode_get_current_version($this->page->id);
856         $format = $version->contentformat;
857 
858         if (empty($comments)) {
859             echo $OUTPUT->heading(get_string('nocomments', 'wikicode'));
860         }
861 
862         foreach ($comments as $comment) {
863 
864             $user = wikicode_get_user_info($comment->userid);
865 
866             $fullname = fullname($user, has_capability('moodle/site:viewfullnames',
                     get_context_instance(CONTEXT_COURSE, $course->id)));
867             $by = new stdclass();
868             $by->name = '<a href="' . $CFG->wwwroot . '/user/view.php?id=' . $user
                    ->id . '&amp;course=' . $course->id . '">' . $fullname . '</a>';
869             $by->date = userdate($comment->timecreated);
870 
871             $t = new html_table();
872             $cell1 = new html_table_cell($OUTPUT->user_picture($user, array('popup'
                     => true)));
873             $cell2 = new html_table_cell(get_string('bynameondate', 'forum', $by));
874             $cell3 = new html_table_cell();
875             $cell3->atributtes ['width'] = "80%";
876             $cell4 = new html_table_cell();
877             $cell5 = new html_table_cell();
878 
879             $row1 = new html_table_row();
880             $row1->cells[] = $cell1;
881             $row1->cells[] = $cell2;
882             $row2 = new html_table_row();
883             $row2->cells[] = $cell3;
884 
```

```
885             if ($format != 'html') {
886                 if ($format == 'creole') {
887                     $parsedcontent = wikicode_parse_content('creole', $comment->
                            content, $options);
888                 } else if ($format == 'nwiki') {
889                     $parsedcontent = wikicode_parse_content('nwiki', $comment->
                            content, $options);
890                 }
891 
892                 $cell4->text = format_text(html_entity_decode($parsedcontent['
                        parsed_text']), FORMAT_HTML);
893             } else {
894                 $cell4->text = format_text($comment->content, FORMAT_HTML);
895             }
896 
897             $row2->cells[] = $cell4;
898 
899             $t->data = array($row1, $row2);
900 
901             $actionicons = false;
902             if ((has_capability('mod/wikicode:managecomment', $this->modcontext)))
                    {
903                 $urledit = new moodle_url('/mod/wikicode/editcomments.php', array('
                        commentid' => $comment->id, 'pageid' => $page->id, 'action' =>
                        'edit'));
904                 $urldelet = new moodle_url('/mod/wikicode/instancecomments.php',
                        array('commentid' => $comment->id, 'pageid' => $page->id, '
                        action' => 'delete'));
905                 $actionicons = true;
906             } else if ((has_capability('mod/wikicode:editcomment', $this->
                    modcontext)) and ($USER->id == $user->id)) {
907                 $urledit = new moodle_url('/mod/wikicode/editcomments.php', array('
                        commentid' => $comment->id, 'pageid' => $page->id, 'action' =>
                        'edit'));
908                 $urldelet = new moodle_url('/mod/wikicode/instancecomments.php',
                        array('commentid' => $comment->id, 'pageid' => $page->id, '
                        action' => 'delete'));
909                 $actionicons = true;
910             }
911 
912             if ($actionicons) {
913                 $cell6 = new html_table_cell($OUTPUT->action_icon($urledit, new
                        pix_icon('t/edit', get_string('edit'))) . $OUTPUT->action_icon(
                        $urldelet, new pix_icon('t/delete', get_string('delete'))));
914                 $row3 = new html_table_row();
915                 $row3->cells[] = $cell5;
916                 $row3->cells[] = $cell6;
917                 $t->data[] = $row3;
918             }
919 
920             echo html_writer::tag('div', html_writer::table($t), array('class'=>'no
                    -overflow'));
921 
922         }
```

```
923         }
924
925         function set_url() {
926             global $PAGE, $CFG;
927             $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/comments.php', array('pageid'
                    => $this->page->id));
928         }
929
930         protected function create_navbar() {
931             global $PAGE, $CFG;
932
933             parent::create_navbar();
934             $PAGE->navbar->add(get_string('comments', 'wikicode'));
935         }
936
937     }
938
939     /**
940      * Class that models the behavior of wiki's edit comment
941      *
942      * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
943      */
944     class page_wikicode_editcomment extends page_wikicode {
945         private $comment;
946         private $action;
947         private $form;
948         private $format;
949
950         function set_url() {
951             global $PAGE, $CFG;
952             $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/comments.php', array('pageid'
                    => $this->page->id));
953         }
954
955         function print_header() {
956             parent::print_header();
957             $this->print_pagetitle();
958         }
959
960         function print_content() {
961             global $PAGE;
962
963             require_capability('mod/wikicode:editcomment', $this->modcontext, NULL,
                    true, 'noeditcommentpermission', 'wikicode');
964
965             if ($this->action == 'add') {
966                 $this->add_comment_form();
967             } else if ($this->action == 'edit') {
968                 $this->edit_comment_form($this->comment);
969             }
970         }
971
972         function set_action($action, $comment) {
973             global $CFG;
```

```
974             require_once($CFG->dirroot . '/mod/wikicode/comments_form.php');
975
976             $this->action = $action;
977             $this->comment = $comment;
978             $version = wikicode_get_current_version($this->page->id);
979             $this->format = $version->contentformat;
980
981             if ($this->format == 'html') {
982                 $destination = $CFG->wwwroot . '/mod/wikicode/instancecomments.php?
                        pageid=' . $this->page->id;
983                 $this->form = new mod_wikicode_comments_form($destination);
984             }
985         }
986
987         protected function create_navbar() {
988             global $PAGE, $CFG;
989
990             $PAGE->navbar->add(get_string('comments', 'wikicode'), $CFG->wwwroot . '/
                    mod/wikicode/comments.php?pageid=' . $this->page->id);
991
992             if ($this->action == 'add') {
993                 $PAGE->navbar->add(get_string('insertcomment', 'wikicode'));
994             } else {
995                 $PAGE->navbar->add(get_string('editcomment', 'wikicode'));
996             }
997         }
998
999         protected function setup_tabs() {
1000            parent::setup_tabs(array('linkedwhenactive' => 'comments', 'activetab' => '
                    comments'));
1001        }
1002
1003        private function add_comment_form() {
1004            global $CFG;
1005            require_once($CFG->dirroot . '/mod/wikicode/editors/wiki_editor.php');
1006
1007            $pageid = $this->page->id;
1008
1009            if ($this->format == 'html') {
1010                $com = new stdClass();
1011                $com->action = 'add';
1012                $com->commentoptions = array('trusttext' => true, 'maxfiles' => 0);
1013                $this->form->set_data($com);
1014                $this->form->display();
1015            } else {
1016                wikicode_print_editor_wiki($this->page->id, null, $this->format, -1,
                        null, false, null, 'addcomments');
1017            }
1018        }
1019
1020        private function edit_comment_form($com) {
1021            global $CFG;
1022            require_once($CFG->dirroot . '/mod/wikicode/comments_form.php');
1023            require_once($CFG->dirroot . '/mod/wikicode/editors/wiki_editor.php');
```

```
1024
1025            if ($this->format == 'html') {
1026                $com->action = 'edit';
1027                $com->entrycomment_editor['text'] = $com->content;
1028                $com->commentoptions = array('trusttext' => true, 'maxfiles' => 0);
1029
1030                $this->form->set_data($com);
1031                $this->form->display();
1032            } else {
1033                wikicode_print_editor_wiki($this->page->id, $com->content, $this->
                        format, -1, null, false, array(), 'editcomments', $com->id);
1034            }
1035
1036        }
1037
1038    }
1039
1040    /**
1041     * Wiki page search page
1042     *
1043     * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
1044     */
1045    class page_wikicode_search extends page_wikicode {
1046        private $search_result;
1047
1048        protected function create_navbar() {
1049            global $PAGE, $CFG;
1050
1051            $PAGE->navbar->add(format_string($this->title));
1052        }
1053
1054        function set_search_string($search, $searchcontent) {
1055            $swid = $this->subwiki->id;
1056            if ($searchcontent) {
1057                $this->search_result = wikicode_search_all($swid, $search);
1058            } else {
1059                $this->search_result = wikicode_search_title($swid, $search);
1060            }
1061
1062        }
1063
1064        function set_url() {
1065            global $PAGE, $CFG;
1066            $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/search.php');
1067        }
1068        function print_content() {
1069            global $PAGE;
1070
1071            require_capability('mod/wikicode:viewpage', $this->modcontext, NULL, true,
                        'noviewpagepermission', 'wikicode');
1072
1073            echo $this->wikioutput->search_result($this->search_result, $this->subwiki)
                        ;
1074        }
```

```
1075  }
1076
1077  /**
1078   *
1079   * Class that models the behavior of wiki's
1080   * create page
1081   *
1082   */
1083  class page_wikicode_create extends page_wikicode {
1084
1085      private $format;
1086      private $swid;
1087      private $wid;
1088      private $action;
1089      private $mform;
1090
1091      function print_header($cm, $course) {
1092          $this->set_url();
1093          parent::print_header($cm, $course);
1094      }
1095
1096      function set_url() {
1097          global $PAGE, $CFG;
1098
1099          $params = array();
1100          if ($this->action == 'new') {
1101              $params['action'] = 'new';
1102              $params['swid'] = $this->swid;
1103              $params['wid'] = $this->wid;
1104              if ($this->title != get_string('newpage', 'wikicode')) {
1105                  $params['title'] = $this->title;
1106              }
1107              //$PAGE->set_url($CFG->wwwroot . '/mod/wikicode/create.php', $params);
1108          } else {
1109              $params['action'] = 'create';
1110              $params['swid'] = $this->swid;
1111              //$PAGE->set_url($CFG->wwwroot . '/mod/wikicode/create.php', $params);
1112          }
1113      }
1114
1115      function set_format($format) {
1116          $this->format = $format;
1117      }
1118
1119      function set_wid($wid) {
1120          $this->wid = $wid;
1121      }
1122
1123      function set_swid($swid) {
1124          $this->swid = $swid;
1125      }
1126
1127      function set_action($action) {
1128          global $PAGE;
```

```
1129            $this->action = $action;
1130
1131            require_once(dirname(__FILE__) . '/create_form.php');
1132            $url = new moodle_url('./create.php', array('action' => 'create', 'wid' =>
                    $this->wid, 'gid' => $this->gid, 'uid' => $this->uid));
1133            $formats = wikicode_get_formats();
1134            $options = array('formats' => $formats, 'defaultformat' => $PAGE->
                    activityrecord->defaultformat, 'forceformat' => $PAGE->activityrecord->
                    forceformat);
1135            if ($this->title != get_string('newpage', 'wikicode')) {
1136                $options['disable_pagetitle'] = true;
1137            }
1138            $this->mform = new mod_wikicode_create_form(str_replace("amp;","",$url->out
                    ()), $options);
1139        }
1140
1141        protected function create_navbar() {
1142            global $PAGE;
1143
1144            $PAGE->navbar->add($this->title);
1145        }
1146
1147        function print_content($pagetitle = '') {
1148
1149            // @TODO: Change this to has_capability and show an alternative interface.
1150            require_capability('mod/wikicode:createpage', $this->modcontext, NULL, true
                    , 'nocreatepermission', 'wikicode');
1151
1152            $data = new stdClass();
1153            if (!empty($pagetitle)) {
1154                $data->pagetitle = $pagetitle;
1155            }
1156            $data->pageformat = "wcode";
1157
1158            $this->mform->set_data($data);
1159            $this->mform->display();
1160        }
1161
1162        function create_page($pagetitle) {
1163            global $USER, $CFG;
1164            $data = $this->mform->get_data();
1165            if (empty($this->subwiki)) {
1166                $swid = wikicode_add_subwiki($this->wid, $this->gid, $this->uid);
1167                $this->subwiki = wikicode_get_subwiki($swid);
1168            }
1169            if ($data) {
1170                $id = wikicode_create_page($this->subwiki->id, $data->pagetitle, $data
                        ->pageformat, $USER->id);
1171            } else {
1172                $id = wikicode_create_page($this->subwiki->id, $pagetitle, $PAGE->
                        activityrecord->defaultformat, $USER->id);
1173            }
1174            redirect($CFG->wwwroot . '/mod/wikicode/edit.php?pageid=' . $id);
1175        }
```

```
1176 }
1177
1178 /**
1179  * Class that models the behavior of wiki's
1180  * compile page
1181  *
1182  */
1183 class page_wikicode_compile extends page_wikicode_edit {
1184
1185     private $newcontent, $download;
1186
1187     function print_header() {
1188     }
1189
1190     function print_content() {
1191         $wiki = wikicode_get_wikicode_from_pageid($this->page->id);
1192         $cm = get_coursemodule_from_instance('wikicode', $wiki->id);
1193
1194         $context = get_context_instance(CONTEXT_MODULE, $cm->id);
1195         require_capability('mod/wikicode:editpage', $context, NULL, true, '
                noeditpermission', 'wikicode');
1196
1197         $this->print_compile();
1198     }
1199
1200     function set_newcontent($newcontent) {
1201         $this->newcontent = $newcontent;
1202     }
1203
1204         function set_download($download) {
1205                 $this->download = $download;
1206         }
1207
1208     protected function set_session_url() {
1209     }
1210
1211     protected function print_compile() {
1212         global $CFG, $USER, $OUTPUT, $PAGE;
1213
1214         $url = $CFG->wwwroot . '/mod/wikicode/edit.php?pageid=' . $this->page->id;
1215         if (!empty($this->section)) {
1216             $url .= "&section=" . urlencode($this->section);
1217         }
1218
1219         $params = array('attachmentoptions' => page_wikicode_edit::
                $attachmentoptions, 'format' => $this->format, 'version' => $this->
                versionnumber);
1220
1221         if ($this->format != 'html') {
1222             $params['fileitemid'] = $this->page->id;
1223             $params['contextid']  = $this->modcontext->id;
1224             $params['component']  = 'mod_wikicode';
1225             $params['filearea']   = 'attachments';
1226         }
```

```
1227
1228            $form = new mod_wikicode_edit_form($url, $params);
1229
1230            $save = false;
1231            $data = false;
1232            if ($data = $form->get_data()) {
1233
1234                $save = wikicode_compile_page($this->page, $data->newcontent, $USER->id
                        , $this->download);
1235
1236                //deleting old locks
1237                wikicode_delete_locks($this->page->id, $USER->id, $this->section);
1238
1239                redirect($CFG->wwwroot . '/mod/wikicode/edit.php?compiled=1&pageid=' .
                        $this->page->id);
1240            } else {
1241                print_error('savingerror', 'wikicode');
1242            }
1243        }
1244    }
1245
1246    /**
1247     *
1248     * Class that models the behavior of wiki's
1249     * view differences
1250     *
1251     */
1252    class page_wikicode_diff extends page_wikicode {
1253
1254        private $compare;
1255        private $comparewith;
1256
1257        function print_header($cm, $course) {
1258            global $OUTPUT;
1259
1260            parent::print_header($cm, $course);
1261
1262            //$this->print_pagetitle();
1263            $vstring = new stdClass();
1264            $vstring->old = $this->compare;
1265            $vstring->new = $this->comparewith;
1266            echo get_string('comparewith', 'wikicode', $vstring);
1267        }
1268
1269        /**
1270         * Print the diff view
1271         */
1272        function print_content() {
1273            global $PAGE;
1274
1275            require_capability('mod/wikicode:viewpage', $this->modcontext, NULL, true,
                    'noviewpagepermission', 'wikicode');
1276
1277            $this->print_diff_content();
```

```
1278        }
1279
1280        function set_url() {
1281            global $PAGE, $CFG;
1282
1283            $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/diff.php', array('pageid' =>
                    $this->page->id, 'comparewith' => $this->comparewith, 'compare' =>
                    $this->compare));
1284        }
1285
1286        function set_comparison($compare, $comparewith) {
1287            $this->compare = $compare;
1288            $this->comparewith = $comparewith;
1289        }
1290
1291        protected function create_navbar() {
1292            global $PAGE, $CFG;
1293
1294            parent::create_navbar();
1295            $PAGE->navbar->add(get_string('history', 'wikicode'), $CFG->wwwroot . '/mod
                    /wikicode/history.php?pageid' . $this->page->id);
1296            $PAGE->navbar->add(get_string('diff', 'wikicode'));
1297        }
1298
1299        protected function setup_tabs() {
1300            parent::setup_tabs(array('linkedwhenactive' => 'history', 'activetab' => '
                    history'));
1301        }
1302
1303        /**
1304         * Given two versions of a page, prints a page displaying the differences
                 between them.
1305         *
1306         * @global object $CFG
1307         * @global object $OUTPUT
1308         * @global object $PAGE
1309         */
1310        private function print_diff_content() {
1311            global $CFG, $OUTPUT, $PAGE;
1312
1313            $pageid = $this->page->id;
1314            $total = wikicode_count_wikicode_page_versions($pageid) - 1;
1315
1316            $oldversion = wikicode_get_wikicode_page_version($pageid, $this->compare);
1317
1318            $newversion = wikicode_get_wikicode_page_version($pageid, $this->
                    comparewith);
1319
1320            if ($oldversion && $newversion) {
1321
1322                $oldtext = format_text(wikicode_remove_tags($oldversion->content),
                        FORMAT_PLAIN, array('overflowdiv'=>true));
1323                        $newtext = format_text(wikicode_remove_tags($newversion->
                            content), FORMAT_PLAIN, array('overflowdiv'=>true));
```

```
1324                list($diff1, $diff2) = ouwiki_diff_html($oldtext, $newtext);
1325                $oldversion->diff = $diff1;
1326                $oldversion->user = wikicode_get_user_info($oldversion->userid);
1327                $newversion->diff = $diff2;
1328                $newversion->user = wikicode_get_user_info($newversion->userid);
1329
1330            } else {
1331                print_error('versionerror', 'wikicode');
1332            }
1333        }
1334    }
1335
1336    /**
1337     *
1338     * Class that models the behavior of wiki's history page
1339     *
1340     */
1341    class page_wikicode_history extends page_wikicode {
1342        /**
1343         * @var int $paging current page
1344         */
1345        private $paging;
1346
1347        /**
1348         * @var int @rowsperpage Items per page
1349         */
1350        private $rowsperpage = 10;
1351
1352        /**
1353         * @var int $allversion if $allversion != 0, all versions will be printed in a
                 signle table
1354         */
1355        private $allversion;
1356
1357        function __construct($wiki, $subwiki, $cm) {
1358            global $PAGE;
1359            parent::__construct($wiki, $subwiki, $cm);
1360            //$PAGE->requires->js_init_call('M.mod_wikicode.history', null, true);
1361        }
1362
1363        function print_header($cm, $course) {
1364            parent::print_header($cm, $course);
1365            //$this->print_pagetitle();
1366            parent::print_tab();
1367        }
1368
1369        function print_pagetitle() {
1370            global $OUTPUT;
1371            $html = '';
1372
1373            $html .= $OUTPUT->container_start();
1374            $html .= $OUTPUT->heading_with_help(format_string($this->title), 'history',
                     'wikicode');
1375            $html .= $OUTPUT->container_end();
```

```
1376            echo $html;
1377        }
1378
1379        function print_content() {
1380            global $PAGE;
1381
1382            require_capability('mod/wikicode:viewpage', $this->modcontext, NULL, true,
                    'noviewpagepermission', 'wikicode');
1383
1384            $this->print_history_content();
1385        }
1386
1387        function set_url() {
1388            global $PAGE, $CFG;
1389            $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/history.php', array('pageid'
                    => $this->page->id));
1390        }
1391
1392        function set_paging($paging) {
1393            $this->paging = $paging;
1394        }
1395
1396        function set_allversion($allversion) {
1397            $this->allversion = $allversion;
1398        }
1399
1400        protected function create_navbar() {
1401            global $PAGE, $CFG;
1402
1403            parent::create_navbar();
1404            $PAGE->navbar->add(get_string('history', 'wikicode'));
1405        }
1406
1407        /**
1408         * Prints the history for a given wiki page
1409         *
1410         * @global object $CFG
1411         * @global object $OUTPUT
1412         * @global object $PAGE
1413         */
1414        private function print_history_content() {
1415            global $CFG, $OUTPUT, $PAGE;
1416
1417            $pageid = $this->page->id;
1418            $offset = $this->paging * $this->rowsperpage;
1419            // vcount is the latest version
1420            $vcount = wikicode_count_wikicode_page_versions($pageid) - 1;
1421            if ($this->allversion) {
1422                $versions = wikicode_get_wikicode_page_versions($pageid, 0, $vcount);
1423            } else {
1424                $versions = wikicode_get_wikicode_page_versions($pageid, $offset,
                        $vcount);
1425            }
1426            // We don't want version 0 to be displayed
```

```
1427            // version 0 is blank page
1428            if (end($versions)->version == 0) {
1429                array_pop($versions);
1430            }
1431
1432            $contents = array();
1433
1434            $version0page = wikicode_get_wikicode_page_version($this->page->id, 0);
1435            $creator = wikicode_get_user_info($version0page->userid);
1436            $a = new StdClass;
1437            $a->date = userdate($this->page->timecreated, get_string('
                    strftimedaydatetime', 'langconfig'));
1438            $a->username = fullname($creator);
1439
1440            //echo $OUTPUT->heading(get_string('createddate', 'wikicode', $a), 4, '
                    wikicode_headingtime');
1441            if ($vcount > 0) {
1442
1443                /// If there is only one version, we don't need radios nor forms
1444                if (count($versions) == 1) {
1445
1446                    $row = array_shift($versions);
1447
1448                    $username = wikicode_get_user_info($row->userid);
1449                    $date = userdate($row->timecreated, get_string('strftimedate', '
                            langconfig'));
1450                    $time = userdate($row->timecreated, get_string('strftimetime', '
                            langconfig'));
1451                    $versionid = wikicode_get_version($row->id);
1452                    $versionlink = $CFG->wwwroot . '/mod/wikicode/viewversion.php?
                            pageid=' . $pageid . '&versionid=' . $versionid->id;
1453                    $userlink = $CFG->wwwroot . '/user/view.php?id=' . $username->id;
1454                    $contents[] = array('', html_writer::link($versionlink, $row->
                            version), $picture . html_writer::link($userlink, fullname(
                            $username)), $time);
1455
1456                    $table = new html_table();
1457                    $table->head = array('', get_string('version'), get_string('user'),
                             get_string('modified'), '');
1458                    $table->data = $contents;
1459                    $table->attributes['class'] = 'mdl-align';
1460                                    $table->attributes['align'] = 'center';
1461
1462                    echo html_writer::table($table);
1463
1464                } else {
1465
1466                    $checked = $vcount - $offset;
1467                    $rowclass = array();
1468
1469                    foreach ($versions as $version) {
1470                        $user = wikicode_get_user_info($version->userid);
1471                        $date = userdate($version->timecreated, get_string('
                                strftimedate'));
```

```
1472                        $rowclass[] = 'wikicode_histnewdate';
1473                        $time = userdate($version->timecreated, get_string('
                                strftimetime', 'langconfig'));
1474                        $versionid = wikicode_get_version($version->id);
1475                        if ($versionid) {
1476                            $url = $CFG->wwwroot . '/mod/wikicode/viewversion.php?
                                    pageid=' . $pageid . '&versionid=' . $versionid->id;
1477                            $viewlink = html_writer::link($url, $version->version);
1478                        } else {
1479                            $viewlink = $version->version;
1480                        }
1481                        $userlink = new moodle_url($CFG->wwwroot . '/user/view.php',
                                array('id' => $version->userid));
1482                        $contents[] = array($viewlink, $picture . html_writer::link(
                                $userlink->out(false), fullname($user)), $time, "");
1483                    }
1484
1485                    $table = new html_table();
1486
1487                    //$icon = $OUTPUT->help_icon('diff', 'wikicode');
1488
1489                    $table->head = array(get_string('version'), get_string('user'),
                            get_string('modified'), '');
1490                    $table->data = $contents;
1491                    $table->attributes['class'] = 'generaltable mdl-align';
1492                                $table->attributes['align'] = 'center';
1493                    $table->rowclasses = $rowclass;
1494
1495                    ///Print the form
1496                                echo html_writer::start_tag('form', array('action'
                                    =>$CFG->wwwroot . '/mod/wikicode/diff.php?
                                    method=get&id=diff'));
1497                    echo html_writer::tag('div', html_writer::empty_tag('input', array(
                            'type'=>'hidden', 'name'=>'pageid', 'value'=>$pageid)));
1498                    echo html_writer::table($table);
1499                    echo html_writer::start_tag('div', array('class'=>'mdl-align'));
1500                    //echo html_writer::empty_tag('input', array('type'=>'submit', '
                            class'=>'wikicode_form-button', 'value'=>get_string('comparesel
                            ', 'wikicode')));
1501                    echo html_writer::end_tag('div');
1502                    echo html_writer::end_tag('form');
1503
1504                                //echo '<form action=' . new moodle_url('/mod/
                                    wikicode/diff.php') . ' id="diff" method="get
                                    ">';
1505
1506                }
1507            } else {
1508                print_string('nohistory', 'wikicode');
1509            }
1510
1511            if (!$this->allversion) {
```

```
1512                //$pagingbar = moodle_paging_bar::make($vcount, $this->paging, $this->
                        rowsperpage, $CFG->wwwroot.'/mod/wikicode/history.php?pageid='.
                        $pageid.'&amp;');
1513                // $pagingbar->pagevar = $pagevar;
1514                //echo $OUTPUT->paging_bar($vcount, $this->paging, $this->rowsperpage,
                        $CFG->wwwroot . '/mod/wikicode/history.php?pageid=' . $pageid . '&
                        amp;');
1515                //print_paging_bar($vcount, $paging, $rowsperpage,$CFG->wwwroot.'/mod/
                        wikicode/history.php?pageid='.$pageid.'&amp;','paging');
1516                } else {
1517                $link = new moodle_url('/mod/wikicode/history.php', array('pageid' =>
                        $pageid));
1518                //$OUTPUT->container(html_writer::link($link->out(false), get_string('
                        viewperpage', 'wikicode', $this->rowsperpage)), 'mdl-align');
1519            }
1520            if ($vcount > $this->rowsperpage && !$this->allversion) {
1521                $link = new moodle_url('/mod/wikicode/history.php', array('pageid' =>
                        $pageid, 'allversion' => 1));
1522                //$OUTPUT->container(html_writer::link($link->out(false), get_string('
                        viewallhistory', 'wikicode')), 'mdl-align');
1523            }
1524        }
1525
1526        /**
1527         * Given an array of values, creates a group of radio buttons to be part of a
                form
1528         *
1529         * @param array  $options  An array of value-label pairs for the radio group (
                values as keys).
1530         * @param string $name     Name of the radiogroup (unique in the form).
1531         * @param string $onclick  Function to be executed when the radios are clicked.
1532         * @param string $checked  The value that is already checked.
1533         * @param bool   $return   If true, return the HTML as a string, otherwise
                print it.
1534         *
1535         * @return mixed If $return is false, returns nothing, otherwise returns a
                string of HTML.
1536         */
1537        private function choose_from_radio($options, $name, $onclick = '', $checked = '
                ', $return = false) {
1538
1539            static $idcounter = 0;
1540
1541            if (!$name) {
1542                $name = 'unnamed';
1543            }
1544
1545            $output = '<span class="radiogroup ' . $name . "\">\n";
1546
1547            if (!empty($options)) {
1548                $currentradio = 0;
1549                foreach ($options as $value => $label) {
1550                    $htmlid = 'auto-rb' . sprintf('%04d', ++$idcounter);
```

```
1551                    $output .= ' <span class="radioelement ' . $name . ' rb' .
                            $currentradio . "\">";
1552                    $output .= '<input name="' . $name . '" id="' . $htmlid . '" type="
                            radio" value="' . $value . '"';
1553                    if ($value == $checked) {
1554                        $output .= ' checked="checked"';
1555                    }
1556                    if ($onclick) {
1557                        $output .= ' onclick="' . $onclick . '"';
1558                    }
1559                    if ($label === '') {
1560                        $output .= ' /> <label for="' . $htmlid . '">' . $value . '</
                                label></span>' . "\n";
1561                    } else {
1562                        $output .= ' /> <label for="' . $htmlid . '">' . $label . '</
                                label></span>' . "\n";
1563                    }
1564                    $currentradio = ($currentradio + 1) % 2;
1565                }
1566            }
1567
1568            $output .= '</span>' . "\n";
1569
1570            if ($return) {
1571                return $output;
1572            } else {
1573                echo $output;
1574            }
1575        }
1576    }
1577
1578    /**
1579     * Class that models the behavior of wiki's map page
1580     *
1581     */
1582    class page_wikicode_map extends page_wikicode {
1583
1584        /**
1585         * @var int wiki view option
1586         */
1587        private $view;
1588
1589        function print_header() {
1590            parent::print_header();
1591            $this->print_pagetitle();
1592        }
1593
1594        function print_content() {
1595            global $CFG, $PAGE;
1596
1597            require_capability('mod/wikicode:viewpage', $this->modcontext, NULL, true,
                    'noviewpagepermission', 'wikicode');
1598
1599            if ($this->view > 0) {
```

```
1600                //echo '<div><a href="' . $CFG->wwwroot . '/mod/wikicode/map.php?pageid
                        =' . $this->page->id . '">' . get_string('backtomapmenu', 'wikicode
                        ') . '</a></div>';
1601            }
1602
1603            switch ($this->view) {
1604            case 1:
1605                echo $this->wikioutput->menu_map($this->page->id, $this->view);
1606                $this->print_contributions_content();
1607                break;
1608            case 2:
1609                echo $this->wikioutput->menu_map($this->page->id, $this->view);
1610                $this->print_navigation_content();
1611                break;
1612            case 3:
1613                echo $this->wikioutput->menu_map($this->page->id, $this->view);
1614                $this->print_orphaned_content();
1615                break;
1616            case 4:
1617                echo $this->wikioutput->menu_map($this->page->id, $this->view);
1618                $this->print_index_content();
1619                break;
1620            case 5:
1621                echo $this->wikioutput->menu_map($this->page->id, $this->view);
1622                $this->print_page_list_content();
1623                break;
1624            case 6:
1625                echo $this->wikioutput->menu_map($this->page->id, $this->view);
1626                $this->print_updated_content();
1627                break;
1628            default:
1629                echo $this->wikioutput->menu_map($this->page->id, $this->view);
1630                $this->print_page_list_content();
1631            }
1632        }
1633
1634        function set_view($option) {
1635            $this->view = $option;
1636        }
1637
1638        function set_url() {
1639            global $PAGE, $CFG;
1640            $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/map.php', array('pageid' =>
                        $this->page->id));
1641        }
1642
1643        protected function create_navbar() {
1644            global $PAGE;
1645
1646            parent::create_navbar();
1647            $PAGE->navbar->add(get_string('map', 'wikicode'));
1648        }
1649
1650        /**
```

```
1651         * Prints the contributions tab content
1652         *
1653         * @uses $OUTPUT, $USER
1654         *
1655         */
1656        private function print_contributions_content() {
1657            global $CFG, $OUTPUT, $USER;
1658            $page = $this->page;
1659
1660            if ($page->timerendered + wikicode_REFRESH_CACHE_TIME < time()) {
1661                $fresh = wikicode_refresh_cachedcontent($page);
1662                $page = $fresh['page'];
1663            }
1664
1665            $swid = $this->subwiki->id;
1666
1667            $table = new html_table();
1668            $table->head = array(get_string('contributions', 'wikicode') . $OUTPUT->
                help_icon('contributions', 'wikicode'));
1669            $table->attributes['class'] = 'wikicode_editor generalbox';
1670            $table->data = array();
1671            $table->rowclasses = array();
1672
1673            $lastversions = array();
1674            $pages = array();
1675            $users = array();
1676
1677            if ($contribs = wikicode_get_contributions($swid, $USER->id)) {
1678                foreach ($contribs as $contrib) {
1679                    if (!array_key_exists($contrib->pageid, $pages)) {
1680                        $page = wikicode_get_page($contrib->pageid);
1681                        $pages[$contrib->pageid] = $page;
1682                    } else {
1683                        continue;
1684                    }
1685
1686                    if (!array_key_exists($page->id, $lastversions)) {
1687                        $version = wikicode_get_last_version($page->id);
1688                        $lastversions[$page->id] = $version;
1689                    } else {
1690                        $version = $lastversions[$page->id];
1691                    }
1692
1693                    if (!array_key_exists($version->userid, $users)) {
1694                        $user = wikicode_get_user_info($version->userid);
1695                        $users[$version->userid] = $user;
1696                    } else {
1697                        $user = $users[$version->userid];
1698                    }
1699
1700                    $link = wikicode_parser_link(format_string($page->title), array('
                        swid' => $swid));
1701                    $class = ($link['new']) ? 'class="wiki_newentry"' : '';
1702
```

```
1703                $linkpage = '<a href="' . $link['url'] . '"' . $class . '>' . $link
                        ['content'] . '</a>';
1704                $icon = $OUTPUT->user_picture($user, array('popup' => true));
1705
1706                $table->data[] = array("$icon $linkpage");
1707            }
1708        } else {
1709            $table->data[] = array(get_string('nocontribs', 'wikicode'));
1710        }
1711        echo html_writer::table($table);
1712    }
1713
1714    /**
1715     * Prints the navigation tab content
1716     *
1717     * @uses $OUTPUT
1718     *
1719     */
1720    private function print_navigation_content() {
1721        global $OUTPUT;
1722        $page = $this->page;
1723
1724        if ($page->timerendered + wikicode_REFRESH_CACHE_TIME < time()) {
1725            $fresh = wikicode_refresh_cachedcontent($page);
1726            $page = $fresh['page'];
1727        }
1728
1729        $tolinks = wikicode_get_linked_to_pages($page->id);
1730        $fromlinks = wikicode_get_linked_from_pages($page->id);
1731
1732        $table = new html_table();
1733        $table->attributes['class'] = 'wikicode_navigation_from';
1734        $table->head = array(get_string('navigationfrom', 'wikicode') . $OUTPUT->
                help_icon('navigationfrom', 'wikicode') . ':');
1735        $table->data = array();
1736        $table->rowclasses = array();
1737        foreach ($fromlinks as $link) {
1738            $lpage = wikicode_get_page($link->frompageid);
1739            $link = new moodle_url('/mod/wikicode/view.php', array('pageid' =>
                    $lpage->id));
1740            $table->data[] = array(html_writer::link($link->out(false),
                    format_string($lpage->title)));
1741            $table->rowclasses[] = 'mdl-align';
1742        }
1743
1744        $table_left = html_writer::table($table);
1745
1746        $table = new html_table();
1747        $table->attributes['class'] = 'wikicode_navigation_to';
1748        $table->head = array(get_string('navigationto', 'wikicode') . $OUTPUT->
                help_icon('navigationto', 'wikicode') . ':');
1749        $table->data = array();
1750        $table->rowclasses = array();
1751        foreach ($tolinks as $link) {
```

```
1752                if ($link->tomissingpage) {
1753                    $viewlink = new moodle_url('/mod/wikicode/create.php', array('swid'
                        => $page->subwikiid, 'title' => $link->tomissingpage, 'action'
                        => 'new'));
1754                    $table->data[] = array(html_writer::link($viewlink->out(false),
                        format_string($link->tomissingpage), array('class' => '
                        wikicode_newentry')));
1755                } else {
1756                    $lpage = wikicode_get_page($link->topageid);
1757                    $viewlink = new moodle_url('/mod/wikicode/view.php', array('pageid'
                        => $lpage->id));
1758                    $table->data[] = array(html_writer::link($viewlink->out(false),
                        format_string($lpage->title)));
1759                }
1760                $table->rowclasses[] = 'mdl-align';
1761            }
1762            $table_right = html_writer::table($table);
1763            echo $OUTPUT->container($table_left . $table_right, '
                wikicode_navigation_container');
1764        }
1765
1766        /**
1767         * Prints the index page tab content
1768         *
1769         *
1770         */
1771        private function print_index_content() {
1772            global $OUTPUT;
1773            $page = $this->page;
1774
1775            if ($page->timerendered + wikicode_REFRESH_CACHE_TIME < time()) {
1776                $fresh = wikicode_refresh_cachedcontent($page);
1777                $page = $fresh['page'];
1778            }
1779
1780            $node = new navigation_node($page->title);
1781
1782            $keys = array();
1783            $tree = array();
1784            $tree = wikicode_build_tree($page, $node, $keys);
1785
1786            $table = new html_table();
1787            $table->head = array(get_string('pageindex', 'wikicode') . $OUTPUT->
                help_icon('pageindex', 'wikicode'));
1788            $table->attributes['class'] = 'wikicode_editor generalbox';
1789            $table->data[] = array($this->render_navigation_node($tree));
1790
1791            echo html_writer::table($table);
1792        }
1793
1794        /**
1795         * Prints the page list tab content
1796         *
1797         *
```

```
1798         */
1799        private function print_page_list_content() {
1800            global $OUTPUT;
1801            $page = $this->page;
1802
1803            if ($page->timerendered + wikicode_REFRESH_CACHE_TIME < time()) {
1804                $fresh = wikicode_refresh_cachedcontent($page);
1805                $page = $fresh['page'];
1806            }
1807
1808            $pages = wikicode_get_page_list($this->subwiki->id);
1809
1810            $stdaux = new stdClass();
1811            $strspecial = get_string('special', 'wikicode');
1812
1813            foreach ($pages as $page) {
1814                $letter = textlib::strtoupper(textlib::substr($page->title, 0, 1));
1815                if (preg_match('/[A-Z]/', $letter)) {
1816                    $stdaux->{
1817                        $letter}
1818                    [] = wikicode_parser_link($page);
1819                } else {
1820                    $stdaux->{
1821                        $strspecial}
1822                    [] = wikicode_parser_link($page);
1823                }
1824            }
1825
1826            $table = new html_table();
1827            $table->head = array(get_string('pagelist', 'wikicode') . $OUTPUT->
                help_icon('pagelist', 'wikicode'));
1828            $table->attributes['class'] = 'wikicode_editor generalbox';
1829            $table->align = array('center');
1830            foreach ($stdaux as $key => $elem) {
1831                $table->data[] = array($key);
1832                foreach ($elem as $e) {
1833                    $table->data[] = array(html_writer::link($e['url'], $e['content']))
                        ;
1834                }
1835            }
1836            echo html_writer::table($table);
1837        }
1838
1839        /**
1840         * Prints the orphaned tab content
1841         *
1842         *
1843         */
1844        private function print_orphaned_content() {
1845            global $OUTPUT;
1846
1847            $page = $this->page;
1848
1849            if ($page->timerendered + wikicode_REFRESH_CACHE_TIME < time()) {
```

```
1850                $fresh = wikicode_refresh_cachedcontent($page);
1851                $page = $fresh['page'];
1852            }
1853
1854            $swid = $this->subwiki->id;
1855
1856            $table = new html_table();
1857            $table->head = array(get_string('orphaned', 'wikicode') . $OUTPUT->
                    help_icon('orphaned', 'wikicode'));
1858            $table->attributes['class'] = 'wikicode_editor generalbox';
1859            $table->data = array();
1860            $table->rowclasses = array();
1861
1862            if ($orphanedpages = wikicode_get_orphaned_pages($swid)) {
1863                foreach ($orphanedpages as $page) {
1864                    $link = wikicode_parser_link($page->title, array('swid' => $swid));
1865                    $class = ($link['new']) ? 'class="wiki_newentry"' : '';
1866                    $table->data[] = array('<a href="' . $link['url'] . '"' . $class .
                            '>' . format_string($link['content']) . '</a>');
1867                }
1868            } else {
1869                $table->data[] = array(get_string('noorphanedpages', 'wikicode'));
1870            }
1871
1872            echo html_writer::table($table);
1873        }
1874
1875        /**
1876         * Prints the updated tab content
1877         *
1878         * @uses $COURSE, $OUTPUT
1879         *
1880         */
1881        private function print_updated_content() {
1882            global $COURSE, $OUTPUT;
1883            $page = $this->page;
1884
1885            if ($page->timerendered + wikicode_REFRESH_CACHE_TIME < time()) {
1886                $fresh = wikicode_refresh_cachedcontent($page);
1887                $page = $fresh['page'];
1888            }
1889
1890            $swid = $this->subwiki->id;
1891
1892            $table = new html_table();
1893            $table->head = array(get_string('updatedpages', 'wikicode') . $OUTPUT->
                    help_icon('updatedpages', 'wikicode'));
1894            $table->attributes['class'] = 'wikicode_editor generalbox';
1895            $table->data = array();
1896            $table->rowclasses = array();
1897
1898            if ($pages = wikicode_get_updated_pages_by_subwiki($swid)) {
1899                $strdataux = '';
1900                foreach ($pages as $page) {
```

```
1901                    $user = wikicode_get_user_info($page->userid);
1902                    $strdata = strftime('%d %b %Y', $page->timemodified);
1903                    if ($strdata != $strdataux) {
1904                        $table->data[] = array($OUTPUT->heading($strdata, 4));
1905                        $strdataux = $strdata;
1906                    }
1907                    $link = wikicode_parser_link($page->title, array('swid' => $swid));
1908                    $class = ($link['new']) ? 'class="wiki_newentry"' : '';
1909
1910                    $linkpage = '<a href="' . $link['url'] . '"' . $class . '>' .
                            format_string($link['content']) . '</a>';
1911                    $icon = $OUTPUT->user_picture($user, array($COURSE->id));
1912                    $table->data[] = array("$icon $linkpage");
1913                }
1914            } else {
1915                $table->data[] = array(get_string('noupdatedpages', 'wikicode'));
1916            }
1917
1918            echo html_writer::table($table);
1919        }
1920
1921        protected function render_navigation_node($items, $attrs = array(),
                $expansionlimit = null, $depth = 1) {
1922
1923            // exit if empty, we don't want an empty ul element
1924            if (count($items) == 0) {
1925                return '';
1926            }
1927
1928            // array of nested li elements
1929            $lis = array();
1930            foreach ($items as $item) {
1931                if (!$item->display) {
1932                    continue;
1933                }
1934                $content = $item->get_content();
1935                $title = $item->get_title();
1936                if ($item->icon instanceof renderable) {
1937                    $icon = $this->wikioutput->render($item->icon);
1938                    $content = $icon . ' ' . $content; // use CSS for spacing of
                            icons
1939                    }
1940                if ($item->helpbutton !== null) {
1941                    $content = trim($item->helpbutton) . html_writer::tag('span',
                            $content, array('class' => 'clearhelpbutton'));
1942                }
1943
1944                if ($content === '') {
1945                    continue;
1946                }
1947
1948                if ($item->action instanceof action_link) {
1949                    //TODO: to be replaced with something else
1950                    $link = $item->action;
```

```
1951                    if ($item->hidden) {
1952                        $link->add_class('dimmed');
1953                    }
1954                    $content = $this->output->render($link);
1955                } else if ($item->action instanceof moodle_url) {
1956                    $attributes = array();
1957                    if ($title !== '') {
1958                        $attributes['title'] = $title;
1959                    }
1960                    if ($item->hidden) {
1961                        $attributes['class'] = 'dimmed_text';
1962                    }
1963                    $content = html_writer::link($item->action, $content, $attributes);
1964
1965                } else if (is_string($item->action) || empty($item->action)) {
1966                    $attributes = array();
1967                    if ($title !== '') {
1968                        $attributes['title'] = $title;
1969                    }
1970                    if ($item->hidden) {
1971                        $attributes['class'] = 'dimmed_text';
1972                    }
1973                    $content = html_writer::tag('span', $content, $attributes);
1974                }
1975
1976                // this applies to the li item which contains all child lists too
1977                $liclasses = array($item->get_css_type(), 'depth_' . $depth);
1978                if ($item->has_children() && (!$item->forceopen || $item->collapse)) {
1979                    $liclasses[] = 'collapsed';
1980                }
1981                if ($item->isactive === true) {
1982                    $liclasses[] = 'current_branch';
1983                }
1984                $liattr = array('class' => join(' ', $liclasses));
1985                // class attribute on the div item which only contains the item content
1986                $divclasses = array('tree_item');
1987                if ((empty($expansionlimit) || $item->type != $expansionlimit) && (
                        $item->children->count() > 0 || ($item->nodetype == navigation_node
                        ::NODETYPE_BRANCH && $item->children->count() == 0 && isloggedin())
                        )) {
1988                    $divclasses[] = 'branch';
1989                } else {
1990                    $divclasses[] = 'leaf';
1991                }
1992                if (!empty($item->classes) && count($item->classes) > 0) {
1993                    $divclasses[] = join(' ', $item->classes);
1994                }
1995                $divattr = array('class' => join(' ', $divclasses));
1996                if (!empty($item->id)) {
1997                    $divattr['id'] = $item->id;
1998                }
1999                $content = html_writer::tag('p', $content, $divattr) . $this->
                        render_navigation_node($item->children, array(), $expansionlimit,
                        $depth + 1);
```

```
2000                if (!empty($item->preceedwithhr) && $item->preceedwithhr === true) {
2001                    $content = html_writer::empty_tag('hr') . $content;
2002                }
2003                $content = html_writer::tag('li', $content, $liattr);
2004                $lis[] = $content;
2005            }
2006
2007            if (count($lis)) {
2008                return html_writer::tag('ul', implode("\n", $lis), $attrs);
2009            } else {
2010                return '';
2011            }
2012        }
2013
2014    }
2015
2016    /**
2017     * Class that models the behavior of wiki's restore version page
2018     *
2019     */
2020    class page_wikicode_restoreversion extends page_wikicode {
2021        private $version;
2022
2023        function print_header($cm, $course) {
2024            parent::print_header($cm, $course);
2025        }
2026
2027        function print_content() {
2028            global $CFG, $PAGE;
2029
2030            require_capability('mod/wikicode:managewiki', $this->modcontext, NULL, true
                    , 'nomanagewikipermission', 'wikicode');
2031
2032            $this->print_restoreversion();
2033        }
2034
2035        function set_url() {
2036            global $PAGE, $CFG;
2037            $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/viewversion.php', array('
                    pageid' => $this->page->id, 'versionid' => $this->version->id));
2038        }
2039
2040        function set_versionid($versionid) {
2041            $this->version = wikicode_get_version($versionid);
2042        }
2043
2044        protected function create_navbar() {
2045            global $PAGE, $CFG;
2046
2047            parent::create_navbar();
2048            $PAGE->navbar->add(get_string('restoreversion', 'wikicode'));
2049        }
2050
2051        protected function setup_tabs() {
```

```
2052            parent::setup_tabs(array('linkedwhenactive' => 'history', 'activetab' => '
                    history'));
2053        }
2054
2055        /**
2056         * Prints the restore version content
2057         *
2058         * @uses $CFG
2059         *
2060         * @param page $page The page whose version will be restored
2061         * @param int  $versionid The version to be restored
2062         * @param bool $confirm If false, shows a yes/no confirmation page.
2063         *     If true, restores the old version and redirects the user to the 'view'
                tab.
2064         */
2065        private function print_restoreversion() {
2066            global $CFG;
2067
2068            $version = wikicode_get_version($this->version->id);
2069
2070            $restoreurl = $CFG->wwwroot . '/mod/wikicode/restoreversion.php?confirm=1&
                    pageid=' . $this->page->id . '&versionid=' . $version->id . '&sesskey='
                     . sesskey();
2071            $return = $CFG->wwwroot . '/mod/wikicode/viewversion.php?pageid=' . $this->
                    page->id . '&versionid=' . $version->id;
2072
2073            echo get_string('restoreconfirm', 'wikicode', $version->version);
2074            print_container_start(false, 'wikicode_restoreform');
2075            echo '<form class="wiki_restore_yes" action="' . $restoreurl . '" method="
                    post" id="restoreversion">';
2076            echo '<div><input type="submit" name="confirm" value="' . get_string('yes')
                     . '" /></div>';
2077            echo '</form>';
2078            echo '<form class="wiki_restore_no" action="' . $return . '" method="post">
                    ';
2079            echo '<div><input type="submit" name="norestore" value="' . get_string('no'
                    ) . '" /></div>';
2080            echo '</form>';
2081            print_container_end();
2082        }
2083    }
2084    /**
2085     * Class that models the behavior of wiki's delete comment confirmation page
2086     *
2087     */
2088    class page_wikicode_deletecomment extends page_wikicode {
2089        private $commentid;
2090
2091        function print_header() {
2092            parent::print_header();
2093            $this->print_pagetitle();
2094        }
2095
2096        function print_content() {
```

```
2097            $this->printconfirmdelete();
2098        }
2099
2100        function set_url() {
2101            global $PAGE;
2102            $PAGE->set_url('/mod/wikicode/instancecomments.php', array('pageid' =>
                    $this->page->id, 'commentid' => $this->commentid));
2103        }
2104
2105        public function set_action($action, $commentid, $content) {
2106            $this->action = $action;
2107            $this->commentid = $commentid;
2108            $this->content = $content;
2109        }
2110
2111        protected function create_navbar() {
2112            global $PAGE;
2113
2114            parent::create_navbar();
2115            $PAGE->navbar->add(get_string('deletecommentcheck', 'wikicode'));
2116        }
2117
2118        protected function setup_tabs() {
2119            parent::setup_tabs(array('linkedwhenactive' => 'comments', 'activetab' => '
                    comments'));
2120        }
2121
2122        /**
2123         * Prints the comment deletion confirmation form
2124         *
2125         * @param page $page The page whose version will be restored
2126         * @param int  $versionid The version to be restored
2127         * @param bool $confirm If false, shows a yes/no confirmation page.
2128         *     If true, restores the old version and redirects the user to the 'view'
                tab.
2129         */
2130        private function printconfirmdelete() {
2131            global $OUTPUT;
2132
2133            $strdeletecheck = get_string('deletecommentcheck', 'wikicode');
2134            $strdeletecheckfull = get_string('deletecommentcheckfull', 'wikicode');
2135
2136            //ask confirmation
2137            $optionsyes = array('confirm'=>1, 'pageid'=>$this->page->id, 'action'=>'
                    delete', 'commentid'=>$this->commentid, 'sesskey'=>sesskey());
2138            $deleteurl = new moodle_url('/mod/wikicode/instancecomments.php',
                    $optionsyes);
2139            $return = new moodle_url('/mod/wikicode/comments.php', array('pageid'=>
                    $this->page->id));
2140
2141            echo $OUTPUT->heading($strdeletecheckfull);
2142            print_container_start(false, 'wikicode_deletecommentform');
2143            echo '<form class="wiki_deletecomment_yes" action="' . $deleteurl . '"
                    method="post" id="deletecomment">';
```

```
2144            echo '<div><input type="submit" name="confirmdeletecomment" value="' .
                    get_string('yes') . '" /></div>';
2145            echo '</form>';
2146            echo '<form class="wiki_deletecomment_no" action="' . $return . '" method="
                    post">';
2147            echo '<div><input type="submit" name="norestore" value="' . get_string('no'
                    ) . '" /></div>';
2148            echo '</form>';
2149            print_container_end();
2150        }
2151    }
2152
2153    /**
2154     * Class that models the behavior of wiki's
2155     * save page
2156     *
2157     */
2158    class page_wikicode_save extends page_wikicode_edit {
2159
2160        private $newcontent;
2161
2162        function print_header() {
2163        }
2164
2165        function print_content() {
2166                    $wiki = wikicode_get_wikicode_from_pageid($this->page->id);
2167            $cm = get_coursemodule_from_instance('wikicode', $wiki->id);
2168
2169            $context = get_context_instance(CONTEXT_MODULE, $cm->id);
2170            require_capability('mod/wikicode:editpage', $context, NULL, true, '
                    noeditpermission', 'wikicode');
2171
2172            $this->print_save();
2173        }
2174
2175        function set_newcontent($newcontent) {
2176            $this->newcontent = $newcontent;
2177        }
2178
2179        protected function set_session_url() {
2180        }
2181
2182        protected function print_save() {
2183            global $CFG, $USER, $OUTPUT, $PAGE;
2184
2185            $url = $CFG->wwwroot . '/mod/wikicode/edit.php?pageid=' . $this->page->id;
2186            if (!empty($this->section)) {
2187                $url .= "&section=" . urlencode($this->section);
2188            }
2189
2190            $params = array('attachmentoptions' => page_wikicode_edit::
                    $attachmentoptions, 'format' => $this->format, 'version' => $this->
                    versionnumber);
2191
```

```
2192            if ($this->format != 'html') {
2193                $params['fileitemid'] = $this->page->id;
2194                $params['contextid']  = $this->modcontext->id;
2195                $params['component']  = 'mod_wikicode';
2196                $params['filearea']   = 'attachments';
2197            }
2198
2199            $form = new mod_wikicode_edit_form($url, $params);
2200
2201            $save = false;
2202            $data = false;
2203            if ($data = $form->get_data()) {
2204                if ($this->format == 'html') {
2205                    $data = file_postupdate_standard_editor($data, 'newcontent',
                            page_wikicode_edit::$attachmentoptions, $this->modcontext, '
                            mod_wikicode', 'attachments', $this->subwiki->id);
2206                }
2207
2208                if (isset($this->section)) {
2209                    $save = wikicode_save_section($this->page, $this->section, $data->
                            newcontent, $USER->id);
2210                } else {
2211                    $save = wikicode_save_page($this->page, $data->newcontent, $USER->
                            id);
2212                }
2213            }
2214
2215            if ($save && $data) {
2216                if (!empty($CFG->usetags)) {
2217                    tag_set('wikicode_pages', $this->page->id, $data->tags);
2218                }
2219
2220                $message = '<p>' . get_string('saving', 'wikicode') . '</p>';
2221
2222                if (!empty($save['sections'])) {
2223                    foreach ($save['sections'] as $s) {
2224                        $message .= '<p>' . get_string('repeatedsection', 'wikicode',
                                $s) . '</p>';
2225                    }
2226                }
2227
2228                if ($this->versionnumber + 1 != $save['version']) {
2229                    $message .= '<p>' . get_string('wrongversionsave', 'wikicode') . '
                            </p>';
2230                }
2231
2232                if (isset($errors) && !empty($errors)) {
2233                    foreach ($errors as $e) {
2234                        $message .= "<p>" . get_string('filenotuploadederror', '
                                wikicode', $e->get_filename()) . "</p>";
2235                    }
2236                }
2237
2238                //deleting old locks
```

```
2239                wikicode_delete_locks($this->page->id, $USER->id, $this->section);
2240
2241                redirect($CFG->wwwroot . '/mod/wikicode/edit.php?pageid=' . $this->page
                        ->id);
2242            } else {
2243                print_error('savingerror', 'wikicode');
2244            }
2245        }
2246    }
2247
2248    /**
2249     * Class that models the behavior of wiki's view an old version of a page
2250     *
2251     */
2252    class page_wikicode_viewversion extends page_wikicode {
2253
2254        private $version;
2255
2256        function print_header($cm, $course) {
2257            parent::print_header($cm, $course);
2258        }
2259
2260        function print_content() {
2261            global $PAGE;
2262
2263            require_capability('mod/wikicode:viewpage', $this->modcontext, NULL, true,
                    'noviewpagepermission', 'wikicode');
2264
2265            $this->print_version_view();
2266        }
2267
2268        function set_url() {
2269            global $PAGE, $CFG;
2270            $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/viewversion.php', array('
                    pageid' => $this->page->id, 'versionid' => $this->version->id));
2271        }
2272
2273        function set_versionid($versionid) {
2274            $this->version = wikicode_get_version($versionid);
2275        }
2276
2277        protected function create_navbar() {
2278            global $PAGE, $CFG;
2279
2280            parent::create_navbar();
2281            $PAGE->navbar->add(get_string('history', 'wikicode'), $CFG->wwwroot . '/mod
                    /wikicode/history.php?pageid' . $this->page->id);
2282            $PAGE->navbar->add(get_string('versionnum', 'wikicode', $this->version->
                    version));
2283        }
2284
2285        protected function setup_tabs() {
2286            parent::setup_tabs(array('linkedwhenactive' => 'history', 'activetab' => '
                    history', 'inactivetabs' => array('edit')));
```

```
2287        }
2288
2289        /**
2290         * Given an old page version, output the version content
2291         *
2292         * @global object $CFG
2293         * @global object $OUTPUT
2294         * @global object $PAGE
2295         */
2296        private function print_version_view() {
2297            global $CFG, $OUTPUT, $PAGE;
2298            $pageversion = wikicode_get_version($this->version->id);
2299
2300            if ($pageversion) {
2301                $restorelink = $CFG->wwwroot . '/mod/wikicode/restoreversion.php?' . '
                        pageid=' . $this->page->id . '&versionid=' . $this->version->id;
2302                echo '<p>' . get_string('viewversion', 'wikicode', $pageversion->
                        version) . '<br />' . html_writer::link($restorelink, '(' .
                        get_string('restorethis', 'wikicode') . ')', array('class' => '
                        wikicode_restore')) . ' ' . '</p>';
2303                $userinfo = wikicode_get_user_info($pageversion->userid);
2304                $heading = '<p><strong>' . get_string('modified', 'wikicode') . ':</
                        strong> ' . userdate($pageversion->timecreated, get_string('
                        strftimedatetime', 'langconfig'));
2305                $viewlink = $CFG->wwwroot . '/user/view.php?' . 'id=' . $userinfo->id;
2306                $heading .= '   <strong>' . get_string('user') . ':</
                        strong> ' . html_writer::link($viewlink, fullname($userinfo));
2307                print_container($heading, false, 'mdl-align wikicode_modifieduser
                        wikicode_headingtime');
2308                $options = array('swid' => $this->subwiki->id, 'pretty_print' => true,
                        'pageid' => $this->page->id);
2309
2310                $pageversion->content = wikicode_remove_tags($pageversion->content);
2311                    $content = format_text($pageversion->content, FORMAT_PLAIN, array('
                            overflowdiv'=>true));
2312
2313                            print_simple_box_start('center','70%','','20');
2314                print_box($content);
2315                            print_simple_box_end();
2316
2317            } else {
2318                print_error('versionerror', 'wikicode');
2319            }
2320        }
2321    }
2322
2323    class page_wikicode_confirmrestore extends page_wikicode_save {
2324
2325        private $version;
2326
2327        function set_url() {
2328            global $PAGE, $CFG;
2329            $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/viewversion.php', array('
                    pageid' => $this->page->id, 'versionid' => $this->version->id));
```

```
2330        }
2331
2332        function print_content() {
2333            global $CFG, $PAGE;
2334
2335            require_capability('mod/wikicode:managewiki', $this->modcontext, NULL, true
                    , 'nomanagewikipermission', 'wikicode');
2336
2337            $version = wikicode_get_version($this->version->id);
2338            if (wikicode_restore_page($this->page, $version->content, $version->userid)
                    ) {
2339                redirect($CFG->wwwroot . '/mod/wikicode/view.php?pageid=' . $this->page
                        ->id, get_string('restoring', 'wikicode', $version->version), 3);
2340            } else {
2341                print_error('restoreerror', 'wikicode', $version->version);
2342            }
2343        }
2344
2345        function set_versionid($versionid) {
2346            $this->version = wikicode_get_version($versionid);
2347        }
2348    }
2349
2350    class page_wikicode_prettyview extends page_wikicode {
2351
2352        function print_header() {
2353            global $CFG, $PAGE, $OUTPUT;
2354            $PAGE->set_pagelayout('embedded');
2355            echo $OUTPUT->header();
2356
2357            echo '<h1 id="wiki_printable_title">' . format_string($this->title) . '</h1
                    >';
2358        }
2359
2360        function print_content() {
2361            global $PAGE;
2362
2363            require_capability('mod/wikicode:viewpage', $this->modcontext, NULL, true,
                    'noviewpagepermission', 'wikicode');
2364
2365            $this->print_pretty_view();
2366        }
2367
2368        function set_url() {
2369            global $PAGE, $CFG;
2370
2371            $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/prettyview.php', array('
                    pageid' => $this->page->id));
2372        }
2373
2374        private function print_pretty_view() {
2375            $version = wikicode_get_current_version($this->page->id);
2376
```

```
2377            $content = wikicode_parse_content($version->contentformat, $version->
                    content, array('printable' => true, 'swid' => $this->subwiki->id, '
                    pageid' => $this->page->id, 'pretty_print' => true));
2378
2379            echo '<div id="wiki_printable_content">';
2380            echo format_text($content['parsed_text'], FORMAT_HTML);
2381            echo '</div>';
2382        }
2383    }
2384
2385    class page_wikicode_handlecomments extends page_wikicode {
2386        private $action;
2387        private $content;
2388        private $commentid;
2389        private $format;
2390
2391        function print_header() {
2392            $this->set_url();
2393        }
2394
2395        public function print_content() {
2396            global $CFG, $PAGE, $USER;
2397
2398            if ($this->action == 'add') {
2399                if (has_capability('mod/wikicode:editcomment', $this->modcontext)) {
2400                    $this->add_comment($this->content, $this->commentid);
2401                }
2402            } else if ($this->action == 'edit') {
2403                $comment = wikicode_get_comment($this->commentid);
2404                $edit = has_capability('mod/wikicode:editcomment', $this->modcontext);
2405                $owner = ($comment->userid == $USER->id);
2406                if ($owner && $edit) {
2407                    $this->add_comment($this->content, $this->commentid);
2408                }
2409            } else if ($this->action == 'delete') {
2410                $comment = wikicode_get_comment($this->commentid);
2411                $manage = has_capability('mod/wikicode:managecomment', $this->
                        modcontext);
2412                $owner = ($comment->userid == $USER->id);
2413                if ($owner || $manage) {
2414                    $this->delete_comment($this->commentid);
2415                    redirect($CFG->wwwroot . '/mod/wikicode/comments.php?pageid=' .
                            $this->page->id, get_string('deletecomment', 'wikicode'), 2);
2416                }
2417            }
2418
2419        }
2420
2421        public function set_url() {
2422            global $PAGE, $CFG;
2423            $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/comments.php', array('pageid'
                     => $this->page->id));
2424        }
2425
```

```
2426        public function set_action($action, $commentid, $content) {
2427            $this->action = $action;
2428            $this->commentid = $commentid;
2429            $this->content = $content;
2430
2431            $version = wikicode_get_current_version($this->page->id);
2432            $format = $version->contentformat;
2433
2434            $this->format = $format;
2435        }
2436
2437        private function add_comment($content, $idcomment) {
2438            global $CFG, $PAGE;
2439            require_once($CFG->dirroot . "/mod/wikicode/locallib.php");
2440
2441            $pageid = $this->page->id;
2442
2443            wikicode_add_comment($this->modcontext, $pageid, $content, $this->format);
2444
2445            if (!$idcomment) {
2446                redirect($CFG->wwwroot . '/mod/wikicode/comments.php?pageid=' . $pageid
                        , get_string('createcomment', 'wikicode'), 2);
2447            } else {
2448                $this->delete_comment($idcomment);
2449                redirect($CFG->wwwroot . '/mod/wikicode/comments.php?pageid=' . $pageid
                        , get_string('editingcomment', 'wikicode'), 2);
2450            }
2451        }
2452
2453        private function delete_comment($commentid) {
2454            global $CFG, $PAGE;
2455
2456            $pageid = $this->page->id;
2457
2458            wikicode_delete_comment($commentid, $this->modcontext, $pageid);
2459        }
2460
2461    }
2462
2463    class page_wikicode_lock extends page_wikicode_edit {
2464
2465        public function print_header() {
2466            $this->set_url();
2467        }
2468
2469        protected function set_url() {
2470            global $PAGE, $CFG;
2471
2472            $params = array('pageid' => $this->page->id);
2473
2474            if ($this->section) {
2475                $params['section'] = $this->section;
2476            }
2477
```

```
2478            $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/lock.php', $params);
2479        }
2480
2481        protected function set_session_url() {
2482        }
2483
2484        public function print_content() {
2485            global $USER, $PAGE;
2486
2487            require_capability('mod/wikicode:editpage', $this->modcontext, NULL, true,
                    'noeditpermission', 'wikicode');
2488
2489            wikicode_set_lock($this->page->id, $USER->id, $this->section);
2490        }
2491
2492        public function print_footer() {
2493        }
2494    }
2495
2496    class page_wikicode_overridelocks extends page_wikicode_edit {
2497        function print_header() {
2498            $this->set_url();
2499        }
2500
2501        function print_content() {
2502            global $CFG, $PAGE;
2503
2504            require_capability('mod/wikicode:overridelock', $this->modcontext, NULL,
                    true, 'nooverridelockpermission', 'wikicode');
2505
2506            wikicode_delete_locks($this->page->id, null, $this->section, true, true);
2507
2508            $args = "pageid=" . $this->page->id;
2509
2510            if (!empty($this->section)) {
2511                $args .= "&section=" . urlencode($this->section);
2512            }
2513
2514            redirect($CFG->wwwroot . '/mod/wikicode/edit.php?' . $args, get_string('
                    overridinglocks', 'wikicode'), 2);
2515        }
2516
2517        function set_url() {
2518            global $PAGE, $CFG;
2519
2520            $params = array('pageid' => $this->page->id);
2521
2522            if (!empty($this->section)) {
2523                $params['section'] = $this->section;
2524            }
2525
2526            $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/overridelocks.php', $params);
2527        }
2528
```

```
2529        protected function set_session_url() {
2530        }
2531
2532        private function print_overridelocks() {
2533            global $CFG;
2534
2535            wikicode_delete_locks($this->page->id, null, $this->section, true, true);
2536
2537            $args = "pageid=" . $this->page->id;
2538
2539            if (!empty($this->section)) {
2540                $args .= "&section=" . urlencode($this->section);
2541            }
2542
2543            redirect($CFG->wwwroot . '/mod/wikicode/edit.php?' . $args, get_string('
                overridinglocks', 'wikicode'), 2);
2544        }
2545
2546    }
2547
2548    /**
2549     * This class will let user to delete wiki pages and page versions
2550     *
2551     */
2552    class page_wikicode_admin extends page_wikicode {
2553
2554        public $view, $action;
2555        public $listorphan = false;
2556
2557        /**
2558         * Constructor
2559         *
2560         * @global object $PAGE
2561         * @param mixed $wiki instance of wiki
2562         * @param mixed $subwiki instance of subwiki
2563         * @param stdClass $cm course module
2564         */
2565        function __construct($wiki, $subwiki, $cm) {
2566            global $PAGE;
2567            parent::__construct($wiki, $subwiki, $cm);
2568            $PAGE->requires->js_init_call('M.mod_wikicode.deleteversion', null, true);
2569        }
2570
2571        /**
2572         * Prints header for wiki page
2573         */
2574        function print_header() {
2575            parent::print_header();
2576            $this->print_pagetitle();
2577        }
2578
2579        /**
2580         * This function will display administration view to users with managewiki
              capability
```

```
2581         */
2582        function print_content() {
2583            //make sure anyone trying to access this page has managewiki capabilities
2584            require_capability('mod/wikicode:managewiki', $this->modcontext, NULL, true
                    , 'noviewpagepermission', 'wikicode');
2585
2586            //update wiki cache if timedout
2587            $page = $this->page;
2588            if ($page->timerendered + wikicode_REFRESH_CACHE_TIME < time()) {
2589                $fresh = wikicode_refresh_cachedcontent($page);
2590                $page = $fresh['page'];
2591            }
2592
2593            //dispaly admin menu
2594            echo $this->wikioutput->menu_admin($this->page->id, $this->view);
2595
2596            //Display appropriate admin view
2597            switch ($this->view) {
2598                case 1: //delete page view
2599                    $this->print_delete_content($this->listorphan);
2600                    break;
2601                case 2: //delete version view
2602                    $this->print_delete_version();
2603                    break;
2604                default: //default is delete view
2605                    $this->print_delete_content($this->listorphan);
2606                    break;
2607            }
2608        }
2609
2610        /**
2611         * Sets admin view option
2612         *
2613         * @param int $view page view id
2614         * @param bool $listorphan is only valid for view 1.
2615         */
2616        public function set_view($view, $listorphan = true) {
2617            $this->view = $view;
2618            $this->listorphan = $listorphan;
2619        }
2620
2621        /**
2622         * Sets page url
2623         *
2624         * @global object $PAGE
2625         * @global object $CFG
2626         */
2627        function set_url() {
2628            global $PAGE, $CFG;
2629            $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/admin.php', array('pageid' =>
                     $this->page->id));
2630        }
2631
2632        /**
```

```
2633        * sets navigation bar for the page
2634        *
2635        * @global object $PAGE
2636        */
2637       protected function create_navbar() {
2638           global $PAGE;
2639
2640           parent::create_navbar();
2641           $PAGE->navbar->add(get_string('admin', 'wikicode'));
2642       }
2643
2644       /**
2645        * Show wiki page delete options
2646        *
2647        * @param bool $showorphan
2648        */
2649       protected function print_delete_content($showorphan = true) {
2650           $contents = array();
2651           $table = new html_table();
2652           $table->head = array('','Page name');
2653           $table->attributes['class'] = 'generaltable mdl-align';
2654           $swid = $this->subwiki->id;
2655           if ($showorphan) {
2656               if ($orphanedpages = wikicode_get_orphaned_pages($swid)) {
2657                   $this->add_page_delete_options($orphanedpages, $swid, $table);
2658               } else {
2659                   $table->data[] = array('', get_string('noorphanedpages', 'wikicode'
                        ));
2660               }
2661           } else {
2662               if ($pages = wikicode_get_page_list($swid)) {
2663                   $this->add_page_delete_options($pages, $swid, $table);
2664               } else {
2665                   $table->data[] = array('', get_string('nopages', 'wikicode'));
2666               }
2667           }
2668
2669           ///Print the form
2670           echo html_writer::start_tag('form', array(
2671                                                   'action' => new moodle_url('/mod/
                                                        wikicode/admin.php'),
2672                                                   'method' => 'post'));
2673           echo html_writer::tag('div', html_writer::empty_tag('input', array(
2674                                                                           'type'  =>
                                                                                '
                                                                               hidden
                                                                                ',
2675                                                                           'name'  =>
                                                                                '
                                                                               pageid
                                                                                ',
2676                                                                           'value' =>
                                                                                $this
                                                                               ->page
```

```
                                                                    ->id))
                                                                    );
2677
2678            echo html_writer::empty_tag('input', array('type' => 'hidden', 'name' => '
                    option', 'value' => $this->view));
2679            echo html_writer::table($table);
2680            echo html_writer::start_tag('div', array('class' => 'mdl-align'));
2681            if (!$showorphan) {
2682                echo html_writer::empty_tag('input', array(
2683                                                    'type'    => 'submit',
2684                                                    'class'   => 'wikicode_form-
                                                        button',
2685                                                    'value'   => get_string('
                                                        listorphan', 'wikicode'),
2686                                                    'sesskey' => sesskey()));
2687            } else {
2688                echo html_writer::empty_tag('input', array('type'=>'hidden', 'name'=>'
                        listall', 'value'=>'1'));
2689                echo html_writer::empty_tag('input', array(
2690                                                    'type'    => 'submit',
2691                                                    'class'   => 'wikicode_form-
                                                        button',
2692                                                    'value'   => get_string('
                                                        listall', 'wikicode'),
2693                                                    'sesskey' => sesskey()));
2694            }
2695            echo html_writer::end_tag('div');
2696            echo html_writer::end_tag('form');
2697        }
2698
2699        /**
2700         * helper function for print_delete_content. This will add data to the table.
2701         *
2702         * @global object $OUTPUT
2703         * @param array $pages objects of wiki pages in subwiki
2704         * @param int $swid id of subwiki
2705         * @param object $table reference to the table in which data needs to be added
2706         */
2707        protected function add_page_delete_options($pages, $swid, &$table) {
2708            global $OUTPUT;
2709            foreach ($pages as $page) {
2710                $link = wikicode_parser_link($page->title, array('swid' => $swid));
2711                $class = ($link['new']) ? 'class="wiki_newentry"' : '';
2712                $pagelink = '<a href="' . $link['url'] . '"' . $class . '>' .
                            format_string($link['content']) . '</a>';
2713                $urledit = new moodle_url('/mod/wikicode/edit.php', array('pageid' =>
                            $page->id, 'sesskey' => sesskey()));
2714                $urldelete = new moodle_url('/mod/wikicode/admin.php', array(
2715                                                                    'pageid'  =>
                                                                        $this->page
                                                                        ->id,
2716                                                                    'delete'  =>
                                                                        $page->id,
```

```
2717                                                                    'option'  =>
                                                                            $this->view,
2718                                                                    'listall' => !
                                                                            $this->
                                                                            listorphan?'
                                                                            1': '',
2719                                                                    'sesskey' =>
                                                                            sesskey()));
2720
2721            $editlinks = $OUTPUT->action_icon($urledit, new pix_icon('t/edit',
                    get_string('edit')));
2722            $editlinks .= $OUTPUT->action_icon($urldelete, new pix_icon('t/delete',
                     get_string('delete')));
2723            $table->data[] = array($editlinks, $pagelink);
2724        }
2725    }
2726
2727    /**
2728     * Prints lists of versions which can be deleted
2729     *
2730     * @global object $OUTPUT
2731     */
2732    private function print_delete_version() {
2733        global $OUTPUT;
2734        $pageid = $this->page->id;
2735
2736        // versioncount is the latest version
2737        $versioncount = wikicode_count_wikicode_page_versions($pageid) - 1;
2738        $versions = wikicode_get_wikicode_page_versions($pageid, 0, $versioncount);
2739
2740        // We don't want version 0 to be displayed
2741        // version 0 is blank page
2742        if (end($versions)->version == 0) {
2743            array_pop($versions);
2744        }
2745
2746        $contents = array();
2747        $version0page = wikicode_get_wikicode_page_version($this->page->id, 0);
2748        $creator = wikicode_get_user_info($version0page->userid);
2749        $a = new stdClass();
2750        $a->date = userdate($this->page->timecreated, get_string('
                strftimedaydatetime', 'langconfig'));
2751        $a->username = fullname($creator);
2752        echo $OUTPUT->heading(get_string('createddate', 'wikicode', $a), 4, '
                wikicode_headingtime');
2753        if ($versioncount > 0) {
2754            /// If there is only one version, we don't need radios nor forms
2755            if (count($versions) == 1) {
2756                $row = array_shift($versions);
2757                $username = wikicode_get_user_info($row->userid);
2758                $picture = $OUTPUT->user_picture($username);
2759                $date = userdate($row->timecreated, get_string('strftimedate', '
                        langconfig'));
```

```
2760                    $time = userdate($row->timecreated, get_string('strftimetime', '
                            langconfig'));
2761                    $versionid = wikicode_get_version($row->id);
2762                    $versionlink = new moodle_url('/mod/wikicode/viewversion.php',
                            array('pageid' => $pageid, 'versionid' => $versionid->id));
2763                    $userlink = new moodle_url('/user/view.php', array('id' =>
                            $username->id));
2764                    $picturelink = $picture . html_writer::link($userlink->out(false),
                            fullname($username));
2765                    $historydate = $OUTPUT->container($date, 'wikicode_histdate');
2766                    $contents[] = array('', html_writer::link($versionlink->out(false),
                             $row->version), $picturelink, $time, $historydate);
2767
2768                    //Show current version
2769                    $table = new html_table();
2770                    $table->head = array('', get_string('version'), get_string('user'),
                             get_string('modified'), '');
2771                    $table->data = $contents;
2772                    $table->attributes['class'] = 'mdl-align';
2773
2774                    echo html_writer::table($table);
2775                } else {
2776                    $lastdate = '';
2777                    $rowclass = array();
2778
2779                    foreach ($versions as $version) {
2780                        $user = wikicode_get_user_info($version->userid);
2781                        $picture = $OUTPUT->user_picture($user, array('popup' => true))
                                ;
2782                        $date = userdate($version->timecreated, get_string('
                            strftimedate'));
2783                        if ($date == $lastdate) {
2784                            $date = '';
2785                            $rowclass[] = '';
2786                        } else {
2787                            $lastdate = $date;
2788                            $rowclass[] = 'wikicode_histnewdate';
2789                        }
2790
2791                        $time = userdate($version->timecreated, get_string('
                            strftimetime', 'langconfig'));
2792                        $versionid = wikicode_get_version($version->id);
2793                        if ($versionid) {
2794                            $url = new moodle_url('/mod/wikicode/viewversion.php',
                                    array('pageid' => $pageid, 'versionid' => $versionid->
                                    id));
2795                            $viewlink = html_writer::link($url->out(false), $version->
                                    version);
2796                        } else {
2797                            $viewlink = $version->version;
2798                        }
2799
2800                        $userlink = new moodle_url('/user/view.php', array('id' =>
                            $version->userid));
```

```
2801                        $picturelink = $picture . html_writer::link($userlink->out(
                                false), fullname($user));
2802                        $historydate = $OUTPUT->container($date, 'wikicode_histdate');
2803                        $radiofromelement = $this->choose_from_radio(array($version->
                                version  => null), 'fromversion', 'M.mod_wikicode.
                                deleteversion()', $versioncount, true);
2804                        $radiotoelement = $this->choose_from_radio(array($version->
                                version  => null), 'toversion', 'M.mod_wikicode.
                                deleteversion()', $versioncount, true);
2805                        $contents[] = array( $radiofromelement . $radiotoelement,
                                $viewlink, $picturelink, $time, $historydate);
2806                    }
2807
2808                    $table = new html_table();
2809                    $table->head = array(get_string('deleteversions', 'wikicode'),
                            get_string('version'), get_string('user'), get_string('modified
                            '), '');
2810                    $table->data = $contents;
2811                    $table->attributes['class'] = 'generaltable mdl-align';
2812                    $table->rowclasses = $rowclass;
2813
2814                    ///Print the form
2815                    echo html_writer::start_tag('form', array('action'=>new moodle_url(
                            '/mod/wikicode/admin.php'), 'method' => 'post'));
2816                    echo html_writer::tag('div', html_writer::empty_tag('input', array(
                            'type' => 'hidden', 'name' => 'pageid', 'value' => $pageid)));
2817                    echo html_writer::empty_tag('input', array('type' => 'hidden', '
                            name' => 'option', 'value' => $this->view));
2818                    echo html_writer::empty_tag('input', array('type' => 'hidden', '
                            name' => 'sesskey', 'value' =>  sesskey()));
2819                    echo html_writer::table($table);
2820                    echo html_writer::start_tag('div', array('class' => 'mdl-align'));
2821                    echo html_writer::empty_tag('input', array('type' => 'submit', '
                            class' => 'wikicode_form-button', 'value' => get_string('
                            deleteversions', 'wikicode')));
2822                    echo html_writer::end_tag('div');
2823                    echo html_writer::end_tag('form');
2824                }
2825            } else {
2826                print_string('nohistory', 'wikicode');
2827            }
2828        }
2829
2830        /**
2831         * Given an array of values, creates a group of radio buttons to be part of a
                form
2832         * helper function for print_delete_version
2833         *
2834         * @param array  $options  An array of value-label pairs for the radio group (
                values as keys).
2835         * @param string $name     Name of the radiogroup (unique in the form).
2836         * @param string $onclick  Function to be executed when the radios are clicked.
2837         * @param string $checked  The value that is already checked.
```

```
2838        * @param bool   $return   If true, return the HTML as a string, otherwise
                print it.
2839        *
2840        * @return mixed If $return is false, returns nothing, otherwise returns a
                string of HTML.
2841        */
2842       private function choose_from_radio($options, $name, $onclick = '', $checked = '
               ', $return = false) {
2843
2844           static $idcounter = 0;
2845
2846           if (!$name) {
2847               $name = 'unnamed';
2848           }
2849
2850           $output = '<span class="radiogroup ' . $name . "\">\n";
2851
2852           if (!empty($options)) {
2853               $currentradio = 0;
2854               foreach ($options as $value => $label) {
2855                   $htmlid = 'auto-rb' . sprintf('%04d', ++$idcounter);
2856                   $output .= ' <span class="radioelement ' . $name . ' rb' .
                           $currentradio . "\">";
2857                   $output .= '<input name="' . $name . '" id="' . $htmlid . '" type="
                           radio" value="' . $value . '"';
2858                   if ($value == $checked) {
2859                       $output .= ' checked="checked"';
2860                   }
2861                   if ($onclick) {
2862                       $output .= ' onclick="' . $onclick . '"';
2863                   }
2864                   if ($label === '') {
2865                       $output .= ' /> <label for="' . $htmlid . '">' . $value . '</
                               label></span>' . "\n";
2866                   } else {
2867                       $output .= ' /> <label for="' . $htmlid . '">' . $label . '</
                               label></span>' . "\n";
2868                   }
2869                   $currentradio = ($currentradio + 1) % 2;
2870               }
2871           }
2872
2873           $output .= '</span>' . "\n";
2874
2875           if ($return) {
2876               return $output;
2877           } else {
2878               echo $output;
2879           }
2880       }
2881   }
2882
2883    /* @copyright 2009 David Mudrak <david.mudrak@gmail.com>
2884    * @license http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
```

```
2885     * @since Moodle 2.0
2886     * @package core
2887     * @category output
2888     */
2889    class html_table {
2890
2891        /**
2892         * @var string Value to use for the id attribute of the table
2893         */
2894        public $id = null;
2895
2896        /**
2897         * @var array Attributes of HTML attributes for the <table> element
2898         */
2899        public $attributes = array();
2900
2901        /**
2902         * @var array An array of headings. The n-th array item is used as a heading of
                the n-th column.
2903         * For more control over the rendering of the headers, an array of
                html_table_cell objects
2904         * can be passed instead of an array of strings.
2905         *
2906         * Example of usage:
2907         * $t->head = array('Student', 'Grade');
2908         */
2909        public $head;
2910
2911        /**
2912         * @var array An array that can be used to make a heading span multiple columns
                .
2913         * In this example, {@link html_table:$data} is supposed to have three columns.
                For the first two columns,
2914         * the same heading is used. Therefore, {@link html_table::$head} should
                consist of two items.
2915         *
2916         * Example of usage:
2917         * $t->headspan = array(2,1);
2918         */
2919        public $headspan;
2920
2921        /**
2922         * @var array An array of column alignments.
2923         * The value is used as CSS 'text-align' property. Therefore, possible
2924         * values are 'left', 'right', 'center' and 'justify'. Specify 'right' or 'left
                ' from the perspective
2925         * of a left-to-right (LTR) language. For RTL, the values are flipped
                automatically.
2926         *
2927         * Examples of usage:
2928         * $t->align = array(null, 'right');
2929         * or
2930         * $t->align[1] = 'right';
2931         */
```

```
2932        public $align;
2933
2934        /**
2935         * @var array The value is used as CSS 'size' property.
2936         *
2937         * Examples of usage:
2938         * $t->size = array('50%', '50%');
2939         * or
2940         * $t->size[1] = '120px';
2941         */
2942        public $size;
2943
2944        /**
2945         * @var array An array of wrapping information.
2946         * The only possible value is 'nowrap' that sets the
2947         * CSS property 'white-space' to the value 'nowrap' in the given column.
2948         *
2949         * Example of usage:
2950         * $t->wrap = array(null, 'nowrap');
2951         */
2952        public $wrap;
2953
2954        /**
2955         * @var array Array of arrays or html_table_row objects containing the data.
                Alternatively, if you have
2956         * $head specified, the string 'hr' (for horizontal ruler) can be used
2957         * instead of an array of cells data resulting in a divider rendered.
2958         *
2959         * Example of usage with array of arrays:
2960         * $row1 = array('Harry Potter', '76 %');
2961         * $row2 = array('Hermione Granger', '100 %');
2962         * $t->data = array($row1, $row2);
2963         *
2964         * Example with array of html_table_row objects: (used for more fine-grained
                control)
2965         * $cell1 = new html_table_cell();
2966         * $cell1->text = 'Harry Potter';
2967         * $cell1->colspan = 2;
2968         * $row1 = new html_table_row();
2969         * $row1->cells[] = $cell1;
2970         * $cell2 = new html_table_cell();
2971         * $cell2->text = 'Hermione Granger';
2972         * $cell3 = new html_table_cell();
2973         * $cell3->text = '100 %';
2974         * $row2 = new html_table_row();
2975         * $row2->cells = array($cell2, $cell3);
2976         * $t->data = array($row1, $row2);
2977         */
2978        public $data;
2979
2980        /**
2981         * @deprecated since Moodle 2.0. Styling should be in the CSS.
2982         * @var string Width of the table, percentage of the page preferred.
2983         */
```

```
2984        public $width = null;
2985
2986        /**
2987         * @deprecated since Moodle 2.0. Styling should be in the CSS.
2988         * @var string Alignment for the whole table. Can be 'right', 'left' or 'center
                ' (default).
2989         */
2990        public $tablealign = null;
2991
2992        /**
2993         * @deprecated since Moodle 2.0. Styling should be in the CSS.
2994         * @var int Padding on each cell, in pixels
2995         */
2996        public $cellpadding = null;
2997
2998        /**
2999         * @var int Spacing between cells, in pixels
3000         * @deprecated since Moodle 2.0. Styling should be in the CSS.
3001         */
3002        public $cellspacing = null;
3003
3004        /**
3005         * @var array Array of classes to add to particular rows, space-separated
                string.
3006         * Classes 'r0' or 'r1' are added automatically for every odd or even row,
3007         * respectively. Class 'lastrow' is added automatically for the last row
3008         * in the table.
3009         *
3010         * Example of usage:
3011         * $t->rowclasses[9] = 'tenth'
3012         */
3013        public $rowclasses;
3014
3015        /**
3016         * @var array An array of classes to add to every cell in a particular column,
3017         * space-separated string. Class 'cell' is added automatically by the renderer.
3018         * Classes 'c0' or 'c1' are added automatically for every odd or even column,
3019         * respectively. Class 'lastcol' is added automatically for all last cells
3020         * in a row.
3021         *
3022         * Example of usage:
3023         * $t->colclasses = array(null, 'grade');
3024         */
3025        public $colclasses;
3026
3027        /**
3028         * @var string Description of the contents for screen readers.
3029         */
3030        public $summary;
3031
3032        /**
3033         * Constructor
3034         */
3035        public function __construct() {
```

```
3036            $this->attributes['class'] = '';
3037        }
3038    }
3039
3040    /**
3041     * Simple html output class
3042     *
3043     * @copyright 2009 Tim Hunt, 2010 Petr Skoda
3044     * @license http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
3045     * @since Moodle 2.0
3046     * @package core
3047     * @category output
3048     */
3049    class html_writer {
3050
3051        /**
3052         * Outputs a tag with attributes and contents
3053         *
3054         * @param string $tagname The name of tag ('a', 'img', 'span' etc.)
3055         * @param string $contents What goes between the opening and closing tags
3056         * @param array $attributes The tag attributes (array('src' => $url, 'class' =>
                'class1') etc.)
3057         * @return string HTML fragment
3058         */
3059        public static function tag($tagname, $contents, array $attributes = null) {
3060            return self::start_tag($tagname, $attributes) . $contents . self::end_tag(
                    $tagname);
3061        }
3062
3063        /**
3064         * Outputs an opening tag with attributes
3065         *
3066         * @param string $tagname The name of tag ('a', 'img', 'span' etc.)
3067         * @param array $attributes The tag attributes (array('src' => $url, 'class' =>
                'class1') etc.)
3068         * @return string HTML fragment
3069         */
3070        public static function start_tag($tagname, array $attributes = null) {
3071            return '<' . $tagname . self::attributes($attributes) . '>';
3072        }
3073
3074        /**
3075         * Outputs a closing tag
3076         *
3077         * @param string $tagname The name of tag ('a', 'img', 'span' etc.)
3078         * @return string HTML fragment
3079         */
3080        public static function end_tag($tagname) {
3081            return '</' . $tagname . '>';
3082        }
3083
3084        /**
3085         * Outputs an empty tag with attributes
3086         *
```

```
3087         * @param string $tagname The name of tag ('input', 'img', 'br' etc.)
3088         * @param array $attributes The tag attributes (array('src' => $url, 'class' =>
                 'class1') etc.)
3089         * @return string HTML fragment
3090         */
3091        public static function empty_tag($tagname, array $attributes = null) {
3092            return '<' . $tagname . self::attributes($attributes) . ' />';
3093        }
3094
3095        /**
3096         * Outputs a tag, but only if the contents are not empty
3097         *
3098         * @param string $tagname The name of tag ('a', 'img', 'span' etc.)
3099         * @param string $contents What goes between the opening and closing tags
3100         * @param array $attributes The tag attributes (array('src' => $url, 'class' =>
                 'class1') etc.)
3101         * @return string HTML fragment
3102         */
3103        public static function nonempty_tag($tagname, $contents, array $attributes =
                null) {
3104            if ($contents === '' || is_null($contents)) {
3105                return '';
3106            }
3107            return self::tag($tagname, $contents, $attributes);
3108        }
3109
3110        /**
3111         * Outputs a HTML attribute and value
3112         *
3113         * @param string $name The name of the attribute ('src', 'href', 'class' etc.)
3114         * @param string $value The value of the attribute. The value will be escaped
                 with {@link s()}
3115         * @return string HTML fragment
3116         */
3117        public static function attribute($name, $value) {
3118            if (is_array($value)) {
3119                debugging("Passed an array for the HTML attribute $name",
                        DEBUG_DEVELOPER);
3120            }
3121            if ($value instanceof moodle_url) {
3122                return ' ' . $name . '="' . $value->out() . '"';
3123            }
3124
3125            // special case, we do not want these in output
3126            if ($value === null) {
3127                return '';
3128            }
3129
3130            // no sloppy trimming here!
3131            return ' ' . $name . '="' . s($value) . '"';
3132        }
3133
3134        /**
3135         * Outputs a list of HTML attributes and values
```

```
3136         *
3137         * @param array $attributes The tag attributes (array('src' => $url, 'class' =>
                 'class1') etc.)
3138         *       The values will be escaped with {@link s()}
3139         * @return string HTML fragment
3140         */
3141        public static function attributes(array $attributes = null) {
3142            $attributes = (array)$attributes;
3143            $output = '';
3144            foreach ($attributes as $name => $value) {
3145                $output .= self::attribute($name, $value);
3146            }
3147            return $output;
3148        }
3149
3150        /**
3151         * Generates random html element id.
3152         *
3153         * @staticvar int $counter
3154         * @staticvar type $uniq
3155         * @param string $base A string fragment that will be included in the random ID
                .
3156         * @return string A unique ID
3157         */
3158        public static function random_id($base='random') {
3159            static $counter = 0;
3160            static $uniq;
3161
3162            if (!isset($uniq)) {
3163                $uniq = uniqid();
3164            }
3165
3166            $counter++;
3167            return $base.$uniq.$counter;
3168        }
3169
3170        /**
3171         * Generates a simple html link
3172         *
3173         * @param string|moodle_url $url The URL
3174         * @param string $text The text
3175         * @param array $attributes HTML attributes
3176         * @return string HTML fragment
3177         */
3178        public static function link($url, $text, array $attributes = null) {
3179            $attributes = (array)$attributes;
3180            $attributes['href']  = $url;
3181            return self::tag('a', $text, $attributes);
3182        }
3183
3184        /**
3185         * Generates a simple checkbox with optional label
3186         *
3187         * @param string $name The name of the checkbox
```

```
3188         * @param string $value The value of the checkbox
3189         * @param bool $checked Whether the checkbox is checked
3190         * @param string $label The label for the checkbox
3191         * @param array $attributes Any attributes to apply to the checkbox
3192         * @return string html fragment
3193         */
3194        public static function checkbox($name, $value, $checked = true, $label = '',
                array $attributes = null) {
3195            $attributes = (array)$attributes;
3196            $output = '';
3197
3198            if ($label !== '' and !is_null($label)) {
3199                if (empty($attributes['id'])) {
3200                    $attributes['id'] = self::random_id('checkbox_');
3201                }
3202            }
3203            $attributes['type']    = 'checkbox';
3204            $attributes['value']   = $value;
3205            $attributes['name']    = $name;
3206            $attributes['checked'] = $checked ? 'checked' : null;
3207
3208            $output .= self::empty_tag('input', $attributes);
3209
3210            if ($label !== '' and !is_null($label)) {
3211                $output .= self::tag('label', $label, array('for'=>$attributes['id']));
3212            }
3213
3214            return $output;
3215        }
3216
3217        /**
3218         * Generates a simple select yes/no form field
3219         *
3220         * @param string $name name of select element
3221         * @param bool $selected
3222         * @param array $attributes - html select element attributes
3223         * @return string HTML fragment
3224         */
3225        public static function select_yes_no($name, $selected=true, array $attributes =
                null) {
3226            $options = array('1'=>get_string('yes'), '0'=>get_string('no'));
3227            return self::select($options, $name, $selected, null, $attributes);
3228        }
3229
3230        /**
3231         * Generates a simple select form field
3232         *
3233         * @param array $options associative array value=>label ex.:
3234         *                array(1=>'One', 2=>Two)
3235         *              it is also possible to specify optgroup as complex label array
                ex.:
3236         *                array(array('Odd'=>array(1=>'One', 3=>'Three')), array('Even
                '=>array(2=>'Two')))
```

```
3237        *                array(1=>'One', '--1uniquekey'=>array('More'=>array(2=>'Two',
                3=>'Three')))
3238        * @param string $name name of select element
3239        * @param string|array $selected value or array of values depending on multiple
                attribute
3240        * @param array|bool $nothing add nothing selected option, or false of not
                added
3241        * @param array $attributes html select element attributes
3242        * @return string HTML fragment
3243        */
3244       public static function select(array $options, $name, $selected = '', $nothing =
            array('' => 'choosedots'), array $attributes = null) {
3245           $attributes = (array)$attributes;
3246           if (is_array($nothing)) {
3247               foreach ($nothing as $k=>$v) {
3248                   if ($v === 'choose' or $v === 'choosedots') {
3249                       $nothing[$k] = get_string('choosedots');
3250                   }
3251               }
3252               $options = $nothing + $options; // keep keys, do not override
3253
3254           } else if (is_string($nothing) and $nothing !== '') {
3255               // BC
3256               $options = array(''=>$nothing) + $options;
3257           }
3258
3259           // we may accept more values if multiple attribute specified
3260           $selected = (array)$selected;
3261           foreach ($selected as $k=>$v) {
3262               $selected[$k] = (string)$v;
3263           }
3264
3265           if (!isset($attributes['id'])) {
3266               $id = 'menu'.$name;
3267               // name may contaion [], which would make an invalid id. e.g. numeric
                    question type editing form, assignment quickgrading
3268               $id = str_replace('[', '', $id);
3269               $id = str_replace(']', '', $id);
3270               $attributes['id'] = $id;
3271           }
3272
3273           if (!isset($attributes['class'])) {
3274               $class = 'menu'.$name;
3275               // name may contaion [], which would make an invalid class. e.g.
                    numeric question type editing form, assignment quickgrading
3276               $class = str_replace('[', '', $class);
3277               $class = str_replace(']', '', $class);
3278               $attributes['class'] = $class;
3279           }
3280           $attributes['class'] = 'select ' . $attributes['class']; // Add 'select'
                selector always
3281
3282           $attributes['name'] = $name;
3283
```

```
3284            if (!empty($attributes['disabled'])) {
3285                $attributes['disabled'] = 'disabled';
3286            } else {
3287                unset($attributes['disabled']);
3288            }
3289
3290            $output = '';
3291            foreach ($options as $value=>$label) {
3292                if (is_array($label)) {
3293                    // ignore key, it just has to be unique
3294                    $output .= self::select_optgroup(key($label), current($label),
                                $selected);
3295                } else {
3296                    $output .= self::select_option($label, $value, $selected);
3297                }
3298            }
3299            return self::tag('select', $output, $attributes);
3300        }
3301
3302        /**
3303         * Returns HTML to display a select box option.
3304         *
3305         * @param string $label The label to display as the option.
3306         * @param string|int $value The value the option represents
3307         * @param array $selected An array of selected options
3308         * @return string HTML fragment
3309         */
3310        private static function select_option($label, $value, array $selected) {
3311            $attributes = array();
3312            $value = (string)$value;
3313            if (in_array($value, $selected, true)) {
3314                $attributes['selected'] = 'selected';
3315            }
3316            $attributes['value'] = $value;
3317            return self::tag('option', $label, $attributes);
3318        }
3319
3320        /**
3321         * Returns HTML to display a select box option group.
3322         *
3323         * @param string $groupname The label to use for the group
3324         * @param array $options The options in the group
3325         * @param array $selected An array of selected values.
3326         * @return string HTML fragment.
3327         */
3328        private static function select_optgroup($groupname, $options, array $selected)
                {
3329            if (empty($options)) {
3330                return '';
3331            }
3332            $attributes = array('label'=>$groupname);
3333            $output = '';
3334            foreach ($options as $value=>$label) {
3335                $output .= self::select_option($label, $value, $selected);
```

```
3336            }
3337            return self::tag('optgroup', $output, $attributes);
3338        }
3339
3340        /**
3341         * This is a shortcut for making an hour selector menu.
3342         *
3343         * @param string $type The type of selector (years, months, days, hours,
                minutes)
3344         * @param string $name fieldname
3345         * @param int $currenttime A default timestamp in GMT
3346         * @param int $step minute spacing
3347         * @param array $attributes - html select element attributes
3348         * @return HTML fragment
3349         */
3350        public static function select_time($type, $name, $currenttime = 0, $step = 5,
                array $attributes = null) {
3351            if (!$currenttime) {
3352                $currenttime = time();
3353            }
3354            $currentdate = usergetdate($currenttime);
3355            $userdatetype = $type;
3356            $timeunits = array();
3357
3358            switch ($type) {
3359                case 'years':
3360                    for ($i=1970; $i<=2020; $i++) {
3361                        $timeunits[$i] = $i;
3362                    }
3363                    $userdatetype = 'year';
3364                    break;
3365                case 'months':
3366                    for ($i=1; $i<=12; $i++) {
3367                        $timeunits[$i] = userdate(gmmktime(12,0,0,$i,15,2000), "%B");
3368                    }
3369                    $userdatetype = 'month';
3370                    $currentdate['month'] = (int)$currentdate['mon'];
3371                    break;
3372                case 'days':
3373                    for ($i=1; $i<=31; $i++) {
3374                        $timeunits[$i] = $i;
3375                    }
3376                    $userdatetype = 'mday';
3377                    break;
3378                case 'hours':
3379                    for ($i=0; $i<=23; $i++) {
3380                        $timeunits[$i] = sprintf("%02d",$i);
3381                    }
3382                    break;
3383                case 'minutes':
3384                    if ($step != 1) {
3385                        $currentdate['minutes'] = ceil($currentdate['minutes']/$step)*
                            $step;
3386                    }
```

```
3387
3388                    for ($i=0; $i<=59; $i+=$step) {
3389                        $timeunits[$i] = sprintf("%02d",$i);
3390                    }
3391                    break;
3392                default:
3393                    throw new coding_exception("Time type $type is not supported by
                        html_writer::select_time().");
3394            }
3395
3396            if (empty($attributes['id'])) {
3397                $attributes['id'] = self::random_id('ts_');
3398            }
3399            $timerselector = self::select($timeunits, $name, $currentdate[$userdatetype
                ], null, array('id'=>$attributes['id']));
3400            $label = self::tag('label', get_string(substr($type, 0, -1), 'form'), array
                ('for'=>$attributes['id'], 'class'=>'accesshide'));
3401
3402            return $label.$timerselector;
3403        }
3404
3405        /**
3406         * Shortcut for quick making of lists
3407         *
3408         * Note: 'list' is a reserved keyword ;-)
3409         *
3410         * @param array $items
3411         * @param array $attributes
3412         * @param string $tag ul or ol
3413         * @return string
3414         */
3415        public static function alist(array $items, array $attributes = null, $tag = 'ul
                ') {
3416            $output = '';
3417
3418            foreach ($items as $item) {
3419                $output .= html_writer::start_tag('li') . "\n";
3420                $output .= $item . "\n";
3421                $output .= html_writer::end_tag('li') . "\n";
3422            }
3423
3424            return html_writer::tag($tag, $output, $attributes);
3425        }
3426
3427        /**
3428         * Returns hidden input fields created from url parameters.
3429         *
3430         * @param moodle_url $url
3431         * @param array $exclude list of excluded parameters
3432         * @return string HTML fragment
3433         */
3434        public static function input_hidden_params(moodle_url $url, array $exclude =
                null) {
3435            $exclude = (array)$exclude;
```

```
3436            $params = $url->params();
3437            foreach ($exclude as $key) {
3438                unset($params[$key]);
3439            }
3440
3441            $output = '';
3442            foreach ($params as $key => $value) {
3443                $attributes = array('type'=>'hidden', 'name'=>$key, 'value'=>$value);
3444                $output .= self::empty_tag('input', $attributes)."\n";
3445            }
3446            return $output;
3447        }
3448
3449        /**
3450         * Generate a script tag containing the the specified code.
3451         *
3452         * @param string $jscode the JavaScript code
3453         * @param moodle_url|string $url optional url of the external script, $code
                ignored if specified
3454         * @return string HTML, the code wrapped in <script> tags.
3455         */
3456        public static function script($jscode, $url=null) {
3457            if ($jscode) {
3458                $attributes = array('type'=>'text/javascript');
3459                return self::tag('script', "\n//<![CDATA[\n$jscode\n//]]>\n",
                        $attributes) . "\n";
3460
3461            } else if ($url) {
3462                $attributes = array('type'=>'text/javascript', 'src'=>$url);
3463                return self::tag('script', '', $attributes) . "\n";
3464
3465            } else {
3466                return '';
3467            }
3468        }
3469
3470        /**
3471         * Renders HTML table
3472         *
3473         * This method may modify the passed instance by adding some default properties
                if they are not set yet.
3474         * If this is not what you want, you should make a full clone of your data
                before passing them to this
3475         * method. In most cases this is not an issue at all so we do not clone by
                default for performance
3476         * and memory consumption reasons.
3477         *
3478         * @param html_table $table data to be rendered
3479         * @return string HTML code
3480         */
3481        public static function table(html_table $table) {
3482            // prepare table data and populate missing properties with reasonable
                    defaults
3483
```

```
3484            if (!empty($table->align)) {
3485                foreach ($table->align as $key => $aa) {
3486                    if ($aa) {
3487                        $table->align[$key] = 'text-align:'. fix_align_rtl($aa) .';';
                            // Fix for RTL languages
3488                    } else {
3489                        $table->align[$key] = null;
3490                    }
3491                }
3492            }
3493            if (!empty($table->size)) {
3494                foreach ($table->size as $key => $ss) {
3495                    if ($ss) {
3496                        $table->size[$key] = 'width:'. $ss .';';
3497                    } else {
3498                        $table->size[$key] = null;
3499                    }
3500                }
3501            }
3502            if (!empty($table->wrap)) {
3503                foreach ($table->wrap as $key => $ww) {
3504                    if ($ww) {
3505                        $table->wrap[$key] = 'white-space:nowrap;';
3506                    } else {
3507                        $table->wrap[$key] = '';
3508                    }
3509                }
3510            }
3511            if (!empty($table->head)) {
3512                foreach ($table->head as $key => $val) {
3513                    if (!isset($table->align[$key])) {
3514                        $table->align[$key] = null;
3515                    }
3516                    if (!isset($table->size[$key])) {
3517                        $table->size[$key] = null;
3518                    }
3519                    if (!isset($table->wrap[$key])) {
3520                        $table->wrap[$key] = null;
3521                    }
3522
3523                }
3524            }
3525            if (empty($table->attributes['class'])) {
3526                $table->attributes['class'] = 'generaltable';
3527            }
3528            if (!empty($table->tablealign)) {
3529                $table->attributes['class'] .= ' boxalign' . $table->tablealign;
3530            }
3531
3532            // explicitly assigned properties override those defined via $table->
                    attributes
3533            $table->attributes['class'] = trim($table->attributes['class']);
3534            $attributes = array_merge($table->attributes, array(
3535                    'id'            => $table->id,
```

```
3536                    'width'         => $table->width,
3537                    'summary'       => $table->summary,
3538                    'cellpadding'   => $table->cellpadding,
3539                    'cellspacing'   => $table->cellspacing,
3540                ));
3541            $output = html_writer::start_tag('table', $attributes) . "\n";
3542
3543            $countcols = 0;
3544
3545            if (!empty($table->head)) {
3546                $countcols = count($table->head);
3547
3548                $output .= html_writer::start_tag('thead', array()) . "\n";
3549                $output .= html_writer::start_tag('tr', array()) . "\n";
3550                $keys = array_keys($table->head);
3551                $lastkey = end($keys);
3552
3553                foreach ($table->head as $key => $heading) {
3554                    // Convert plain string headings into html_table_cell objects
3555                    if (!($heading instanceof html_table_cell)) {
3556                        $headingtext = $heading;
3557                        $heading = new html_table_cell();
3558                        $heading->text = $headingtext;
3559                        $heading->header = true;
3560                    }
3561
3562                    if ($heading->header !== false) {
3563                        $heading->header = true;
3564                    }
3565
3566                    if ($heading->header && empty($heading->scope)) {
3567                        $heading->scope = 'col';
3568                    }
3569
3570                    $heading->attributes['class'] .= ' header c' . $key;
3571                    if (isset($table->headspan[$key]) && $table->headspan[$key] > 1) {
3572                        $heading->colspan = $table->headspan[$key];
3573                        $countcols += $table->headspan[$key] - 1;
3574                    }
3575
3576                    if ($key == $lastkey) {
3577                        $heading->attributes['class'] .= ' lastcol';
3578                    }
3579                    if (isset($table->colclasses[$key])) {
3580                        $heading->attributes['class'] .= ' ' . $table->colclasses[$key
                                ];
3581                    }
3582                    $heading->attributes['class'] = trim($heading->attributes['class'])
                            ;
3583                    $attributes = array_merge($heading->attributes, array(
3584                            'style'     => $table->align[$key] . $table->size[$key] .
                                    $heading->style,
3585                            'scope'     => $heading->scope,
3586                            'colspan'   => $heading->colspan,
```

```
3587                        ));
3588
3589                    $tagtype = 'td';
3590                    if ($heading->header === true) {
3591                        $tagtype = 'th';
3592                    }
3593                    $output .= html_writer::tag($tagtype, $heading->text, $attributes)
                            . "\n";
3594                }
3595                $output .= html_writer::end_tag('tr') . "\n";
3596                $output .= html_writer::end_tag('thead') . "\n";
3597
3598                if (empty($table->data)) {
3599                    // For valid XHTML strict every table must contain either a valid
                            tr
3600                    // or a valid tbody... both of which must contain a valid td
3601                    $output .= html_writer::start_tag('tbody', array('class' => 'empty'
                            ));
3602                    $output .= html_writer::tag('tr', html_writer::tag('td', '', array(
                            'colspan'=>count($table->head))));
3603                    $output .= html_writer::end_tag('tbody');
3604                }
3605            }
3606
3607            if (!empty($table->data)) {
3608                $oddeven    = 1;
3609                $keys       = array_keys($table->data);
3610                $lastrowkey = end($keys);
3611                $output .= html_writer::start_tag('tbody', array());
3612
3613                foreach ($table->data as $key => $row) {
3614                    if (($row === 'hr') && ($countcols)) {
3615                        $output .= html_writer::tag('td', html_writer::tag('div', '',
                                array('class' => 'tabledivider')), array('colspan' =>
                                $countcols));
3616                    } else {
3617                        // Convert array rows to html_table_rows and cell strings to
                                html_table_cell objects
3618                        if (!($row instanceof html_table_row)) {
3619                            $newrow = new html_table_row();
3620
3621                            foreach ($row as $cell) {
3622                                if (!($cell instanceof html_table_cell)) {
3623                                    $cell = new html_table_cell($cell);
3624                                }
3625                                $newrow->cells[] = $cell;
3626                            }
3627                            $row = $newrow;
3628                        }
3629
3630                        $oddeven = $oddeven ? 0 : 1;
3631                        if (isset($table->rowclasses[$key])) {
3632                            $row->attributes['class'] .= ' ' . $table->rowclasses[$key
                                    ];
```

```
3633                        }
3634
3635                        $row->attributes['class'] .= ' r' . $oddeven;
3636                        if ($key == $lastrowkey) {
3637                            $row->attributes['class'] .= ' lastrow';
3638                        }
3639
3640                        $output .= html_writer::start_tag('tr', array('class' => trim(
                                $row->attributes['class']), 'style' => $row->style, 'id' =>
                                 $row->id)) . "\n";
3641                        $keys2 = array_keys($row->cells);
3642                        $lastkey = end($keys2);
3643
3644                        $gotlastkey = false; //flag for sanity checking
3645                        foreach ($row->cells as $key => $cell) {
3646                            if ($gotlastkey) {
3647                                //This should never happen. Why do we have a cell after
                                        the last cell?
3648                                mtrace("A cell with key ($key) was found after the last
                                        key ($lastkey)");
3649                            }
3650
3651                            if (!($cell instanceof html_table_cell)) {
3652                                $mycell = new html_table_cell();
3653                                $mycell->text = $cell;
3654                                $cell = $mycell;
3655                            }
3656
3657                            if (($cell->header === true) && empty($cell->scope)) {
3658                                $cell->scope = 'row';
3659                            }
3660
3661                            if (isset($table->colclasses[$key])) {
3662                                $cell->attributes['class'] .= ' ' . $table->colclasses[
                                        $key];
3663                            }
3664
3665                            $cell->attributes['class'] .= ' cell c' . $key;
3666                            if ($key == $lastkey) {
3667                                $cell->attributes['class'] .= ' lastcol';
3668                                $gotlastkey = true;
3669                            }
3670                            $tdstyle = '';
3671                            $tdstyle .= isset($table->align[$key]) ? $table->align[$key
                                    ] : '';
3672                            $tdstyle .= isset($table->size[$key]) ? $table->size[$key]
                                    : '';
3673                            $tdstyle .= isset($table->wrap[$key]) ? $table->wrap[$key]
                                    : '';
3674                            $cell->attributes['class'] = trim($cell->attributes['class'
                                    ]);
3675                            $tdattributes = array_merge($cell->attributes, array(
3676                                    'style' => $tdstyle . $cell->style,
3677                                    'colspan' => $cell->colspan,
```

```
3678                                    'rowspan' => $cell->rowspan,
3679                                    'id' => $cell->id,
3680                                    'abbr' => $cell->abbr,
3681                                    'scope' => $cell->scope,
3682                                ));
3683                            $tagtype = 'td';
3684                            if ($cell->header === true) {
3685                                $tagtype = 'th';
3686                            }
3687                            $output .= html_writer::tag($tagtype, $cell->text,
                                    $tdattributes) . "\n";
3688                        }
3689                    }
3690                    $output .= html_writer::end_tag('tr') . "\n";
3691                }
3692                $output .= html_writer::end_tag('tbody') . "\n";
3693            }
3694            $output .= html_writer::end_tag('table') . "\n";
3695
3696            return $output;
3697        }
3698
3699        /**
3700         * Renders form element label
3701         *
3702         * By default, the label is suffixed with a label separator defined in the
3703         * current language pack (colon by default in the English lang pack).
3704         * Adding the colon can be explicitly disabled if needed. Label separators
3705         * are put outside the label tag itself so they are not read by
3706         * screenreaders (accessibility).
3707         *
3708         * Parameter $for explicitly associates the label with a form control. When
3709         * set, the value of this attribute must be the same as the value of
3710         * the id attribute of the form control in the same document. When null,
3711         * the label being defined is associated with the control inside the label
3712         * element.
3713         *
3714         * @param string $text content of the label tag
3715         * @param string|null $for id of the element this label is associated with,
              null for no association
3716         * @param bool $colonize add label separator (colon) to the label text, if it
              is not there yet
3717         * @param array $attributes to be inserted in the tab, for example array('
              accesskey' => 'a')
3718         * @return string HTML of the label element
3719         */
3720        public static function label($text, $for, $colonize = true, array $attributes=
              array()) {
3721            if (!is_null($for)) {
3722                $attributes = array_merge($attributes, array('for' => $for));
3723            }
3724            $text = trim($text);
3725            $label = self::tag('label', $text, $attributes);
3726
```

```
3727            // TODO MDL-12192 $colonize disabled for now yet
3728            // if (!empty($text) and $colonize) {
3729            //     // the $text may end with the colon already, though it is bad string
                        definition style
3730            //     $colon = get_string('labelsep', 'langconfig');
3731            //     if (!empty($colon)) {
3732            //         $trimmed = trim($colon);
3733            //         if ((substr($text, -strlen($trimmed)) == $trimmed) or (substr(
                        $text, -1) == ':')) {
3734            //             //debugging('The label text should not end with colon or
                        other label separator,
3735            //             //           please fix the string definition.',
                        DEBUG_DEVELOPER);
3736            //         } else {
3737            //             $label .= $colon;
3738            //         }
3739            //     }
3740            // }
3741
3742             return $label;
3743         }
3744     }
3745
3746     /**
3747      * Component representing a table cell.
3748      *
3749      * @copyright 2009 Nicolas Connault
3750      * @license http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
3751      * @since Moodle 2.0
3752      * @package core
3753      * @category output
3754      */
3755     class html_table_cell {
3756
3757         /**
3758          * @var string Value to use for the id attribute of the cell.
3759          */
3760         public $id = null;
3761
3762         /**
3763          * @var string The contents of the cell.
3764          */
3765         public $text;
3766
3767         /**
3768          * @var string Abbreviated version of the contents of the cell.
3769          */
3770         public $abbr = null;
3771
3772         /**
3773          * @var int Number of columns this cell should span.
3774          */
3775         public $colspan = null;
3776
```

```
3777        /**
3778         * @var int Number of rows this cell should span.
3779         */
3780        public $rowspan = null;
3781
3782        /**
3783         * @var string Defines a way to associate header cells and data cells in a
                table.
3784         */
3785        public $scope = null;
3786
3787        /**
3788         * @var bool Whether or not this cell is a header cell.
3789         */
3790        public $header = null;
3791
3792        /**
3793         * @var string Value to use for the style attribute of the table cell
3794         */
3795        public $style = null;
3796
3797        /**
3798         * @var array Attributes of additional HTML attributes for the <td> element
3799         */
3800        public $attributes = array();
3801
3802        /**
3803         * Constructs a table cell
3804         *
3805         * @param string $text
3806         */
3807        public function __construct($text = null) {
3808            $this->text = $text;
3809            $this->attributes['class'] = '';
3810        }
3811    }
3812
3813
3814    /**
3815     * Component representing a table row.
3816     *
3817     * @copyright 2009 Nicolas Connault
3818     * @license http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
3819     * @since Moodle 2.0
3820     * @package core
3821     * @category output
3822     */
3823    class html_table_row {
3824
3825        /**
3826         * @var string Value to use for the id attribute of the row.
3827         */
3828        public $id = null;
3829
```

```
3830        /**
3831         * @var array Array of html_table_cell objects
3832         */
3833        public $cells = array();
3834
3835        /**
3836         * @var string Value to use for the style attribute of the table row
3837         */
3838        public $style = null;
3839
3840        /**
3841         * @var array Attributes of additional HTML attributes for the <tr> element
3842         */
3843        public $attributes = array();
3844
3845        /**
3846         * Constructor
3847         * @param array $cells
3848         */
3849        public function __construct(array $cells=null) {
3850            $this->attributes['class'] = '';
3851            $cells = (array)$cells;
3852            foreach ($cells as $cell) {
3853                if ($cell instanceof html_table_cell) {
3854                    $this->cells[] = $cell;
3855                } else {
3856                    $this->cells[] = new html_table_cell($cell);
3857                }
3858            }
3859        }
3860    }
```

2.1.12. version.php

```
1   <?php
2
3   defined('MOODLE_INTERNAL') || die();
4
5   $module->version   = 2012022100;       // The current module version (Date:
        YYYYMMDDXX)
6   $module->requires  = 2006101592;       // Requires this Moodle version
7   $module->component = 'mod_wikicode';       // Full name of the plugin (used for
        diagnostics)
8   $module->cron      = 0;
```

2.1.13. view.php

```
<?php

require_once('../../config.php');
require_once($CFG->dirroot . '/mod/wikicode/lib.php');
require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
require_once($CFG->dirroot . '/mod/wikicode/pagelib.php');

$id = optional_param('id', 0, PARAM_INT); // Course Module ID

$pageid = optional_param('pageid', 0, PARAM_INT); // Page ID

$wid = optional_param('wid', 0, PARAM_INT); // Wiki ID
$title = optional_param('title', '', PARAM_TEXT); // Page Title
$currentgroup = optional_param('group', 0, PARAM_INT); // Group ID
$userid = optional_param('uid', 0, PARAM_INT); // User ID
$groupanduser = optional_param('groupanduser', 0, PARAM_TEXT);

$edit = optional_param('edit', -1, PARAM_BOOL);

$action = optional_param('action', '', PARAM_ALPHA);
$swid = optional_param('swid', 0, PARAM_INT); // Subwiki ID

/*
 * Case 0:
 *
 * User that comes from a course. First wiki page must be shown
 *
 * URL params: id -> course module id
 *
 */
if ($id) {
    // Cheacking course module instance
    if (!$cm = get_coursemodule_from_id('wikicode', $id)) {
        print_error('invalidcoursemodule');
    }

    // Checking course instance
    $course = get_record('course', 'id', $cm->course);

    // Checking wiki instance
    if (!$wiki = wikicode_get_wiki($cm->instance)) {
        print_error('incorrectwikiid', 'wikicode');
    }

    //$PAGE->set_cm($cm);

    // Getting the subwiki corresponding to that wiki, group and user.
    //
    // Also setting the page if it exists or getting the first page title form
    // that wiki

    // Getting current group id
    $currentgroup = groups_get_activity_group($cm);
    $currentgroup = !empty($currentgroup) ? $currentgroup : 0;
```

```
55      // Getting current user id
56      if ($wiki->wikimode == 'individual') {
57          $userid = $USER->id;
58      } else {
59          $userid = 0;
60      }
61
62      // Getting subwiki. If it does not exists, redirecting to create page
63      if (!$subwiki = wikicode_get_subwiki_by_group($wiki->id, $currentgroup, $userid
            )) {
64          $params = array('wid' => $wiki->id, 'gid' => $currentgroup, 'uid' =>
                $userid, 'title' => $wiki->firstpagetitle);
65          $url = new moodle_url('./create.php', $params);
66          redirect($url->out());
67      }
68
69      // Getting first page. If it does not exists, redirecting to create page
70      if (!$page = wikicode_get_first_page($subwiki->id, $wiki)) {
71          $params = array('swid'=>$subwiki->id, 'title'=>$wiki->firstpagetitle);
72          $url = new moodle_url('./create.php', $params);
73          redirect($url->out());
74      }
75
76      /*
77       * Case 1:
78       *
79       * A user wants to see a page.
80       *
81       * URL Params: pageid -> page id
82       *
83       */
84  } elseif ($pageid) {
85
86      // Checking page instance
87      if (!$page = wikicode_get_page($pageid)) {
88          print_error('incorrectpageid', 'wikicode');
89      }
90
91      // Checking subwiki
92      if (!$subwiki = wikicode_get_subwiki($page->subwikiid)) {
93          print_error('incorrectsubwikiid', 'wikicode');
94      }
95
96      // Checking wiki instance of that subwiki
97      if (!$wiki = wikicode_get_wiki($subwiki->wikiid)) {
98          print_error('incorrectwikiid', 'wikicode');
99      }
100
101     // Checking course module instance
102     if (!$cm = get_coursemodule_from_instance("wikicode", $subwiki->wikiid)) {
103         print_error('invalidcoursemodule');
104     }
105
106     // Checking course instance
```

```
107         $course = get_record('course', 'id', $cm->course);
108
109         /*
110          * Case 2:
111          *
112          * Trying to read a page from another group or user
113          *
114          * Page can exists or not.
115          *  * If it exists, page must be shown
116          *  * If it does not exists, system must ask for its creation
117          *
118          * URL params: wid -> subwiki id (required)
119          *             title -> a page title (required)
120          *             group -> group id (optional)
121          *             uid -> user id (optional)
122          *             groupanduser -> (optional)
123          */
124     } elseif ($wid && $title) {
125
126         // Setting wiki instance
127         if (!$wiki = wikicode_get_wiki($wid)) {
128             print_error('incorrectwikiid', 'wikicode');
129         }
130
131         // Checking course module
132         if (!$cm = get_coursemodule_from_instance("wikicode", $wiki->id)) {
133             print_error('invalidcoursemodule');
134         }
135
136         // Checking course instance
137         if (!$course = get_record("course", "id", $cm->course)) {
138             print_error('coursemisconf');
139         }
140
141         $groupmode = groups_get_activity_groupmode($cm);
142         if (empty($currentgroup)) {
143             $currentgroup = groups_get_activity_group($cm);
144             $currentgroup = !empty($currentgroup) ? $currentgroup : 0;
145         }
146
147         if ($wiki->wikimode == 'individual' && ($groupmode == SEPARATEGROUPS ||
                $groupmode == VISIBLEGROUPS)) {
148             list($gid, $uid) = explode('-', $groupanduser);
149         } else if ($wiki->wikimode == 'individual') {
150             $gid = 0;
151             $uid = $userid;
152         } else if ($groupmode == NOGROUPS) {
153             $gid = 0;
154             $uid = 0;
155         } else {
156             $gid = $currentgroup;
157             $uid = 0;
158         }
159
```

```
160     // Getting subwiki instance. If it does not exists, redirect to create page
161     if (!$subwiki = wikicode_get_subwiki_by_group($wiki->id, $gid, $uid)) {
162         $context = get_context_instance(CONTEXT_MODULE, $cm->id);
163 
164         $modeanduser = $wiki->wikimode == 'individual' && $uid != $USER->id;
165         $modeandgroupmember = $wiki->wikimode == 'collaborative' && !
                groups_is_member($gid);
166 
167         $manage = has_capability('mod/wikicode:managewiki', $context);
168         $edit = has_capability('mod/wikicode:editpage', $context);
169         $manageandedit = $manage && $edit;
170 
171         if ($groupmode == VISIBLEGROUPS and ($modeanduser || $modeandgroupmember)
                and !$manageandedit) {
172             print_error('nocontent','wikicode');
173         }
174 
175         $params = array('wid' => $wiki->id, 'gid' => $gid, 'uid' => $uid, 'title'
                => $title);
176         $url = new moodle_url('/mod/wikicode/create.php', $params);
177         redirect($url);
178     }
179 
180     // Checking is there is a page with this title. If it does not exists, redirect
          to first page
181     if (!$page = wikicode_get_page_by_title($subwiki->id, $title)) {
182         $params = array('wid' => $wiki->id, 'gid' => $gid, 'uid' => $uid, 'title'
                => $wiki->firstpagetitle);
183         $url = new moodle_url('/mod/wikicode/view.php', $params);
184         redirect($url);
185     }
186 
187     //    /*
188     //     * Case 3:
189     //     *
190     //     * A user switches group when is 'reading' a non-existent page.
191     //     *
192     //     * URL Params: wid -> wiki id
193     //     *             title -> page title
194     //     *             currentgroup -> group id
195     //     *
196     //     */
197     //} elseif ($wid && $title && $currentgroup) {
198     //
199     //    // Checking wiki instance
200     //    if (!$wiki = wikicode_get_wiki($wid)) {
201     //        print_error('incorrectwikiid', 'wiki');
202     //    }
203     //
204     //    // Checking subwiki instance
205     //    // @TODO: Fix call to wikicode_get_subwiki_by_group
206     //    if (!$currentgroup = groups_get_activity_group($cm)){
207     //        $currentgroup = 0;
208     //    }
```

```
209         //    if (!$subwiki = wikicode_get_subwiki_by_group($wid, $currentgroup)) {
210         //        print_error('incorrectsubwikiid', 'wiki');
211         //    }
212         //
213         //    // Checking page instance
214         //    if ($page = wikicode_get_page_by_title($subwiki->id, $title)) {
215         //        unset($title);
216         //    }
217         //
218         //    // Checking course instance
219         //    $course = $DB->get_record('course', array('id'=>$cm->course), '*',
                MUST_EXIST);
220         //
221         //    // Checking course module instance
222         //    if (!$cm = get_coursemodule_from_instance("wiki", $wiki->id, $course->id)
                ) {
223         //        print_error('invalidcoursemodule');
224         //    }
225         //
226         //    $subwiki = null;
227         //    $page = null;
228         //
229         //    /*
230         //     * Case 4:
231         //     *
232         //     * Error. No more options
233         //     */
234     } else {
235         print_error('incorrectparameters');
236     }
237     require_login($course, true, $cm);
238
239
240     $context = get_context_instance(CONTEXT_MODULE, $cm->id);
241     require_capability('mod/wikicode:viewpage', $context);
242
243     add_to_log($course->id, 'wikicode', 'view', 'view.php?id=' . $cm->id, $wiki->id);
244
245     // Update 'viewed' state if required by completion system
246     /*require_once($CFG->libdir . '/completionlib.php');
247     $completion = new completion_info($course);
248     $completion->set_module_viewed($cm);
249
250     if (($edit != - 1) and $PAGE->user_allowed_editing()) {
251         $USER->editing = $edit;
252     }*/
253
254     $wikipage = new page_wikicode_view($wiki, $subwiki, $cm);
255
256     /*The following piece of code is used in order
257      * to perform set_url correctly. It is necessary in order
258      * to make page_wikicode_view class know that this page
259      * has been called via its id.
260      */
```

```
261 if ($id) {
262     $wikipage->set_coursemodule($id);
263 }
264
265 $wikipage->set_gid($currentgroup);
266 $wikipage->set_page($page);
267
268 $wikipage->print_header($cm, $course);
269
270 $wikipage->print_content();
271
272 print_footer($course);
```

2.1.14. viewversion.php

```
1  <?php
2
3  require_once('../../config.php');
4
5  require_once($CFG->dirroot . '/mod/wikicode/lib.php');
6  require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
7  require_once($CFG->dirroot . '/mod/wikicode/pagelib.php');
8
9  $pageid = required_param('pageid', PARAM_TEXT);
10 $versionid = required_param('versionid', PARAM_INT);
11
12 if (!$page = wikicode_get_page($pageid)) {
13     print_error('incorrectpageid', 'wikicode');
14 }
15
16 if (!$subwiki = wikicode_get_subwiki($page->subwikiid)) {
17     print_error('incorrectsubwikiid', 'wikicode');
18 }
19
20 if (!$wiki = wikicode_get_wiki($subwiki->wikiid)) {
21     print_error('incorrectwikiid', 'wikicode');
22 }
23
24 if (!$cm = get_coursemodule_from_instance('wikicode', $wiki->id)) {
25     print_error('invalidcoursemodule');
26 }
27
28 $course = get_record('course', 'id', $cm->course);
29
30 require_login($course->id, true, $cm);
31 add_to_log($course->id, "wikicode", "history", "history.php?id=$cm->id", "$wiki->id
       ");
32
33 /// Print the page header
34 $wikipage = new page_wikicode_viewversion($wiki, $subwiki, $cm);
35
```

```
36 $wikipage->set_page($page);
37 $wikipage->set_versionid($versionid);
38
39 $wikipage->print_header($cm, $course);
40 $wikipage->print_content();
41
42 print_footer($course);
```

2.2. Back-End

2.2.1. lib.php

```
1  <?php
2
3  /**
4   * Given an object containing all the necessary data,
5   * (defined by the form in mod.html) this function
6   * will create a new instance and return the id number
7   * of the new instance.
8   *
9   * @param object $instance An object from the form in mod.html
10  * @return int The id of the newly inserted wiki record
11  **/
12 function wikicode_add_instance($wiki) {
13
14     $wiki->timemodified = time();
15     # May have to add extra stuff in here #
16     if (empty($wiki->forceformat)) {
17         $wiki->forceformat = 0;
18     }
19         return insert_record("wikicode", $wiki);
20 }
21
22 /**
23  * Given an object containing all the necessary data,
24  * (defined by the form in mod.html) this function
25  * will update an existing instance with new data.
26  *
27  * @param object $instance An object from the form in mod.html
28  * @return boolean Success/Fail
29  **/
30 function wikicode_update_instance($wiki) {
31
32     $wiki->timemodified = time();
33     $wiki->id = $wiki->instance;
34     if (empty($wiki->forceformat)) {
35         $wiki->forceformat = 0;
36     }
37
```

```
38      # May have to add extra stuff in here #
39
40      return update_record('wikicode', $wiki);
41  }
42
43  /**
44   * Given an ID of an instance of this module,
45   * this function will permanently delete the instance
46   * and any data that depends on it.
47   *
48   * @param int $id Id of the module instance
49   * @return boolean Success/Failure
50   **/
51  function wikicode_delete_instance($id) {
52
53      if (!$wiki = get_record('wikicode', 'id', $id)) {
54          return false;
55      }
56
57      $result = true;
58
59      # Get subwiki information #
60      $subwikis = get_records('wikicode_subwikis', array('wikiid' => $wiki->id));
61
62      foreach ($subwikis as $subwiki) {
63          # Get existing links, and delete them #
64          if (!delete_records('wikicode_links', 'subwikiid',$subwiki->id)) {
65              $result = false;
66          }
67
68          # Get existing pages #
69          if ($pages = get_records('wikicode_pages', 'subwikiid', $subwiki->id)) {
70              foreach ($pages as $page) {
71                  # Get locks, and delete them #
72                  if (!delete_records('wikicode_locks', 'pageid', $page->id)) {
73                      $result = false;
74                  }
75
76                  # Get versions, and delete them #
77                  if (!delete_records('wikicode_versions', 'pageid', $page->id)) {
78                      $result = false;
79                  }
80              }
81
82              # Delete pages #
83              if (!delete_records('wikicode_pages', 'subwikiid', $subwiki->id)) {
84                  $result = false;
85              }
86          }
87
88          # Get existing synonyms, and delete them #
89          if (!delete_records('wikicode_synonyms', 'subwikiid', $subwiki->id)) {
90              $result = false;
91          }
```

```
92
93          # Delete any subwikis #
94          if (!delete_records('wikicode_subwikis', 'id', $subwiki->id)) {
95              $result = false;
96          }
97      }
98
99      # Delete any dependent records here #
100     if (!delete_records('wikicode', 'id', $wiki->id)) {
101         $result = false;
102     }
103
104     return $result;
105 }
106
107 function wikicode_reset_userdata($data) {
108     global $CFG,$DB;
109     require_once($CFG->dirroot . '/mod/wikicode/pagelib.php');
110     require_once($CFG->dirroot . '/tag/lib.php');
111
112     $componentstr = get_string('modulenameplural', 'wikicode');
113     $status = array();
114
115     //get the wiki(s) in this course.
116     if (!$wikis = get_records('wikicode', 'course', $data->courseid)) {
117         return false;
118     }
119     $errors = false;
120     foreach ($wikis as $wiki) {
121
122         // remove all comments
123         if (!empty($data->reset_wikicode_comments)) {
124             if (!$cm = get_coursemodule_from_instance('wikicode', $wiki->id)) {
125                 continue;
126             }
127             $context = get_context_instance(CONTEXT_MODULE, $cm->id);
128             delete_records_select('comments', "contextid = ".$context->id." AND
                    commentarea='wikicode_page'");
129             $status[] = array('component'=>$componentstr, 'item'=>get_string('
                    deleteallcomments'), 'error'=>false);
130         }
131
132         if (!empty($data->reset_wikicode_tags)) {
133             # Get subwiki information #
134             $subwikis = get_records('wikicode_subwikis', 'wikiid', $wiki->id);
135
136             foreach ($subwikis as $subwiki) {
137                 if ($pages = get_records('wikicode_pages', 'subwikiid', $subwiki->
                        id)) {
138                     foreach ($pages as $page) {
139                         $tags = tag_get_tags_array('wikicode_pages', $page->id);
140                         foreach ($tags as $tagid => $tagname) {
141                             // Delete the related tag_instances related to the wiki
                                page.
```

```
142                             $errors = tag_delete_instance('wikicode_pages', $page->
                                    id, $tagid);
143                             $status[] = array('component' => $componentstr, 'item'
                                    => get_string('tagsdeleted', 'wikicode'), 'error'
                                    => $errors);
144                         }
145                     }
146                 }
147             }
148         }
149     }
150     return $status;
151 }
152
153
154 function wikicode_reset_course_form_definition(&$mform) {
155     $mform->addElement('header', 'wikiheader', get_string('modulenameplural', '
            wikicode'));
156     $mform->addElement('advcheckbox', 'reset_wikicode_tags', get_string('
            removeallwikitags', 'wikicode'));
157     $mform->addElement('advcheckbox', 'reset_wikicode_comments', get_string('
            deleteallcomments'));
158 }
159
160 /**
161  * Return a small object with summary information about what a
162  * user has done with a given particular instance of this module
163  * Used for user activity reports.
164  * $return->time = the time they did it
165  * $return->info = a short text description
166  *
167  * @return null
168  * @todo Finish documenting this function
169  **/
170 function wikicode_user_outline($course, $user, $mod, $wiki) {
171     $return = NULL;
172     return $return;
173 }
174
175 /**
176  * Print a detailed representation of what a user has done with
177  * a given particular instance of this module, for user activity reports.
178  *
179  * @return boolean
180  * @todo Finish documenting this function
181  **/
182 function wikicode_user_complete($course, $user, $mod, $wiki) {
183     return true;
184 }
185
186 /**
187  * Indicates API features that the wiki supports.
188  *
189  * @uses FEATURE_GROUPS
```

```
190     * @uses FEATURE_GROUPINGS
191     * @uses FEATURE_GROUPMEMBERSONLY
192     * @uses FEATURE_MOD_INTRO
193     * @uses FEATURE_COMPLETION_TRACKS_VIEWS
194     * @uses FEATURE_COMPLETION_HAS_RULES
195     * @uses FEATURE_GRADE_HAS_GRADE
196     * @uses FEATURE_GRADE_OUTCOMES
197     * @param string $feature
198     * @return mixed True if yes (some features may use other values)
199     */
200    function wikicode_supports($feature) {
201        switch ($feature) {
202        case FEATURE_GROUPS:
203            return true;
204        case FEATURE_GROUPINGS:
205            return true;
206        case FEATURE_GROUPMEMBERSONLY:
207            return true;
208        case FEATURE_MOD_INTRO:
209            return true;
210        case FEATURE_COMPLETION_TRACKS_VIEWS:
211            return true;
212        case FEATURE_GRADE_HAS_GRADE:
213            return false;
214        case FEATURE_GRADE_OUTCOMES:
215            return false;
216        case FEATURE_RATE:
217            return false;
218        case FEATURE_BACKUP_MOODLE2:
219            return true;
220        case FEATURE_SHOW_DESCRIPTION:
221            return true;
222
223        default:
224            return null;
225        }
226    }
227
228    /**
229     * Given a course and a time, this module should find recent activity
230     * that has occurred in wiki activities and print it out.
231     * Return true if there was output, or false is there was none.
232     *
233     * @global $CFG
234     * @global $DB
235     * @uses CONTEXT_MODULE
236     * @uses VISIBLEGROUPS
237     * @param object $course
238     * @param bool $viewfullnames capability
239     * @param int $timestart
240     * @return boolean
241     **/
242    function wikicode_print_recent_activity($course, $viewfullnames, $timestart) {
243        global $CFG, $OUTPUT;
```

```
244
245     $sql = "SELECT p.*, w.id as wikiid, sw.groupid
246             FROM {$CFG->prefix}wikicode_pages p
247                 JOIN {$CFG->prefix}wikicode_subwikis sw ON sw.id = p.subwikiid
248                 JOIN {$CFG->prefix}wikicode w ON w.id = sw.wikiid
249             WHERE p.timemodified > ".$timestart." AND w.course = ".$course->id."
250             ORDER BY p.timemodified ASC";
251     if (!$pages = get_records_sql($sql)) {
252         return false;
253     }
254     $modinfo =& get_fast_modinfo($course);
255
256     $wikis = array();
257
258     $modinfo = get_fast_modinfo($course);
259
260     foreach ($pages as $page) {
261         if (!isset($modinfo->instances['wikicode'][$page->wikiid])) {
262             // not visible
263             continue;
264         }
265         $cm = $modinfo->instances['wikicode'][$page->wikiid];
266         if (!$cm->uservisible) {
267             continue;
268         }
269         $context = get_context_instance(CONTEXT_MODULE, $cm->id);
270
271         if (!has_capability('mod/wikicode:viewpage', $context)) {
272             continue;
273         }
274
275         $groupmode = groups_get_activity_groupmode($cm, $course);
276
277         if ($groupmode) {
278             if ($groupmode == SEPARATEGROUPS and !has_capability('mod/wikicode:
                    managewiki', $context)) {
279                 // separate mode
280                 if (isguestuser()) {
281                     // shortcut
282                     continue;
283                 }
284
285                 if (is_null($modinfo->groups)) {
286                     $modinfo->groups = groups_get_user_groups($course->id); // load
                            all my groups and cache it in modinfo
287                     }
288
289                 if (!in_array($page->groupid, $modinfo->groups[0])) {
290                     continue;
291                 }
292             }
293         }
294         $wikis[] = $page;
295     }
```

```
296         unset($pages);
297 
298         if (!$wikis) {
299             return false;
300         }
301         echo $OUTPUT->heading(get_string("updatedwikipages", 'wikicode') . ':', 3);
302         foreach ($wikis as $wiki) {
303             $cm = $modinfo->instances['wikicode'][$wiki->wikiid];
304             $link = $CFG->wwwroot . '/mod/wikicode/view.php?pageid=' . $wiki->id;
305             print_recent_activity_note($wiki->timemodified, $wiki, $cm->name, $link,
                    false, $viewfullnames);
306         }
307 
308         return true; //  True if anything was printed, otherwise false
309     }
310     /**
311      * Function to be run periodically according to the moodle cron
312      * This function searches for things that need to be done, such
313      * as sending out mail, toggling flags etc ...
314      *
315      * @uses $CFG
316      * @return boolean
317      * @todo Finish documenting this function
318      **/
319     function wikicode_cron() {
320         global $CFG;
321 
322         return true;
323     }
324 
325     /**
326      * Must return an array of grades for a given instance of this module,
327      * indexed by user.  It also returns a maximum allowed grade.
328      *
329      * Example:
330      *    $return->grades = array of grades;
331      *    $return->maxgrade = maximum allowed grade;
332      *
333      *    return $return;
334      *
335      * @param int $wikiid ID of an instance of this module
336      * @return mixed Null or object with an array of grades and with the maximum grade
337      **/
338     function wikicode_grades($wikiid) {
339         return null;
340     }
341 
342     /**
343      * Must return an array of user records (all data) who are participants
344      * for a given instance of wiki. Must include every user involved
345      * in the instance, independient of his role (student, teacher, admin...)
346      * See other modules as example.
347      *
348      * @todo: deprecated - to be deleted in 2.2
```

```
349      *
350      * @param int $wikiid ID of an instance of this module
351      * @return mixed boolean/array of students
352      **/
353     function wikicode_get_participants($wikiid) {
354         return false;
355     }
356
357     /**
358      * This function returns if a scale is being used by one wiki
359      * it it has support for grading and scales. Commented code should be
360      * modified if necessary. See forum, glossary or journal modules
361      * as reference.
362      *
363      * @param int $wikiid ID of an instance of this module
364      * @return mixed
365      * @todo Finish documenting this function
366      **/
367     function wikicode_scale_used($wikiid, $scaleid) {
368         $return = false;
369
370         //$rec = get_record("wiki","id","$wikiid","scale","-$scaleid");
371         //
372         //if (!empty($rec)  && !empty($scaleid)) {
373         //    $return = true;
374         //}
375
376         return $return;
377     }
378
379     /**
380      * Checks if scale is being used by any instance of wiki.
381      * This function was added in 1.9
382      *
383      * This is used to find out if scale used anywhere
384      * @param $scaleid int
385      * @return boolean True if the scale is used by any wiki
386      */
387     function wikicode_scale_used_anywhere($scaleid) {
388
389         //if ($scaleid and $DB->record_exists('wikicode', array('grade' => -$scaleid)))
                 {
390         //    return true;
391         //} else {
392         //    return false;
393         //}
394
395         return false;
396     }
397
398     /**
399      * Pluginfile hook
400      *
401      */
```

```
402 function wikicode_pluginfile($course, $cm, $context, $filearea, $args,
        $forcedownload) {
403     global $CFG;
404
405     if ($context->contextlevel != CONTEXT_MODULE) {
406         return false;
407     }
408
409     require_login($course, true, $cm);
410
411     require_once($CFG->dirroot . "/mod/wikicode/locallib.php");
412
413     if ($filearea == 'attachments') {
414         $swid = (int) array_shift($args);
415
416         if (!$subwiki = wikicode_get_subwiki($swid)) {
417             return false;
418         }
419
420         require_capability('mod/wikicode:viewpage', $context);
421
422         $relativepath = implode('/', $args);
423
424         $fullpath = "/$context->id/mod_wikicode/attachments/$swid/$relativepath";
425
426         $fs = get_file_storage();
427         if (!$file = $fs->get_file_by_hash(sha1($fullpath)) or $file->is_directory
                ()) {
428             return false;
429         }
430
431         $lifetime = isset($CFG->filelifetime) ? $CFG->filelifetime : 86400;
432
433         send_stored_file($file, $lifetime, 0);
434     }
435 }
436
437 function wikicode_search_form($cm, $search = '') {
438     global $CFG, $OUTPUT;
439
440     $output = '<div class="wikisearch">';
441     $output .= '<form method="post" action="' . $CFG->wwwroot . '/mod/wikicode/
            search.php" style="display:inline">';
442     $output .= '<fieldset class="invisiblefieldset">';
443     $output .= '<input name="searchstring" type="text" size="18" value="' . s(
            $search, true) . '" alt="search" />';
444     $output .= '<input name="courseid" type="hidden" value="' . $cm->course . '" />
            ';
445     $output .= '<input name="cmid" type="hidden" value="' . $cm->id . '" />';
446     $output .= '<input name="searchwikicontent" type="hidden" value="1" />';
447     $output .= ' <input value="' . get_string('searchwikis', 'wikicode') . '" type
            ="submit" />';
448     $output .= '</fieldset>';
449     $output .= '</form>';
```

```
450         $output .= '</div>';
451
452         return $output;
453     }
454     function wikicode_extend_navigation(navigation_node $navref, $course, $module, $cm)
            {
455         global $CFG, $PAGE, $USER;
456
457         require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
458
459         $context = get_context_instance(CONTEXT_MODULE, $cm->id);
460         $url = $PAGE->url;
461         $userid = 0;
462         if ($module->wikimode == 'individual') {
463             $userid = $USER->id;
464         }
465
466         if (!$wiki = wikicode_get_wiki($cm->instance)) {
467             return false;
468         }
469
470         if (!$gid = groups_get_activity_group($cm)) {
471             $gid = 0;
472         }
473         if (!$subwiki = wikicode_get_subwiki_by_group($cm->instance, $gid, $userid)) {
474             return null;
475         } else {
476             $swid = $subwiki->id;
477         }
478
479         $pageid = $url->param('pageid');
480         $cmid = $url->param('id');
481         if (empty($pageid) && !empty($cmid)) {
482             // wiki main page
483             $page = wikicode_get_page_by_title($swid, $wiki->firstpagetitle);
484             $pageid = $page->id;
485         }
486
487         if (has_capability('mod/wikicode:createpage', $context)) {
488            //$link = new moodle_url('/mod/wikicode/create.php', array('action' => 'new
                   ', 'swid' => $swid));
489            //$node = $navref->add(get_string('newpage', 'wikicode'), $link,
                   navigation_node::TYPE_SETTING);
490         }
491
492         if (is_numeric($pageid)) {
493
494             if (has_capability('mod/wikicode:viewpage', $context)) {
495                 $link = new moodle_url('/mod/wikicode/view.php', array('pageid' =>
                        $pageid));
496                 $node = $navref->add(get_string('view', 'wikicode'), $link,
                        navigation_node::TYPE_SETTING);
497             }
498
```

```
499             if (has_capability('mod/wikicode:editpage', $context)) {
500                 $link = new moodle_url('/mod/wikicode/edit.php', array('pageid' =>
                        $pageid));
501                 $node = $navref->add(get_string('edit', 'wikicode'), $link,
                        navigation_node::TYPE_SETTING);
502             }
503
504             if (has_capability('mod/wikicode:viewcomment', $context)) {
505                 //$link = new moodle_url('/mod/wikicode/comments.php', array('pageid'
                        => $pageid));
506                 //$node = $navref->add(get_string('comments', 'wikicode'), $link,
                        navigation_node::TYPE_SETTING);
507             }
508
509             if (has_capability('mod/wikicode:viewpage', $context)) {
510                 $link = new moodle_url('/mod/wikicode/history.php', array('pageid' =>
                        $pageid));
511                 $node = $navref->add(get_string('history', 'wikicode'), $link,
                        navigation_node::TYPE_SETTING);
512             }
513
514             if (has_capability('mod/wikicode:viewpage', $context)) {
515                 //$link = new moodle_url('/mod/wikicode/map.php', array('pageid' =>
                        $pageid));
516                 //$node = $navref->add(get_string('map', 'wikicode'), $link,
                        navigation_node::TYPE_SETTING);
517             }
518
519             if (has_capability('mod/wikicode:viewpage', $context)) {
520                 //$link = new moodle_url('/mod/wikicode/files.php', array('pageid' =>
                        $pageid));
521                 //$node = $navref->add(get_string('files', 'wikicode'), $link,
                        navigation_node::TYPE_SETTING);
522             }
523
524                 if (has_capability('mod/wikicode:viewpage', $context)) {
525                 $link = new moodle_url('/mod/wikicode/log.php', array('pageid' =>
                        $pageid));
526                 $node = $navref->add(get_string('log', 'wikicode'), $link,
                        navigation_node::TYPE_SETTING);
527             }
528
529             if (has_capability('mod/wikicode:managewiki', $context)) {
530                 $link = new moodle_url('/mod/wikicode/admin.php', array('pageid' =>
                        $pageid));
531                 $node = $navref->add(get_string('admin', 'wikicode'), $link,
                        navigation_node::TYPE_SETTING);
532             }
533         }
534     }
535     /**
536      * Returns all other caps used in wiki module
537      *
538      * @return array
```

```
539     */
540    function wikicode_get_extra_capabilities() {
541        return array('moodle/comment:view', 'moodle/comment:post', 'moodle/comment:
               delete');
542    }
543
544    /**
545     * Running addtional permission check on plugin, for example, plugins
546     * may have switch to turn on/off comments option, this callback will
547     * affect UI display, not like pluginname_comment_validate only throw
548     * exceptions.
549     * Capability check has been done in comment->check_permissions(), we
550     * don't need to do it again here.
551     *
552     * @param stdClass $comment_param {
553     *              context  => context the context object
554     *              courseid => int course id
555     *              cm       => stdClass course module object
556     *              commentarea => string comment area
557     *              itemid      => int itemid
558     * }
559     * @return array
560     */
561    function wikicode_comment_permissions($comment_param) {
562        return array('post'=>true, 'view'=>true);
563    }
564
565    /**
566     * Validate comment parameter before perform other comments actions
567     *
568     * @param stdClass $comment_param {
569     *              context  => context the context object
570     *              courseid => int course id
571     *              cm       => stdClass course module object
572     *              commentarea => string comment area
573     *              itemid      => int itemid
574     * }
575     * @return boolean
576     */
577    function wikicode_comment_validate($comment_param) {
578        global $CFG;
579        require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
580        // validate comment area
581        if ($comment_param->commentarea != 'wikicode_page') {
582            throw new comment_exception('invalidcommentarea');
583        }
584        // validate itemid
585        if (!$record = get_record('wikicode_pages', 'id', $comment_param->itemid)) {
586            throw new comment_exception('invalidcommentitemid');
587        }
588        if (!$subwiki = wikicode_get_subwiki($record->subwikiid)) {
589            throw new comment_exception('invalidsubwikiid');
590        }
591        if (!$wiki = wikicode_get_wikicode_from_pageid($comment_param->itemid)) {
```

```
592             throw new comment_exception('invalidid', 'data');
593         }
594         if (!$course = get_record('course', 'id', $wiki->course)) {
595             throw new comment_exception('coursemisconf');
596         }
597         if (!$cm = get_coursemodule_from_instance('wikicode', $wiki->id, $course->id))
                {
598             throw new comment_exception('invalidcoursemodule');
599         }
600         $context = get_context_instance(CONTEXT_MODULE, $cm->id);
601         // group access
602         if ($subwiki->groupid) {
603             $groupmode = groups_get_activity_groupmode($cm, $course);
604             if ($groupmode == SEPARATEGROUPS and !has_capability('moodle/site:
                    accessallgroups', $context)) {
605                 if (!groups_is_member($subwiki->groupid)) {
606                     throw new comment_exception('notmemberofgroup');
607                 }
608             }
609         }
610         // validate context id
611         if ($context->id != $comment_param->context->id) {
612             throw new comment_exception('invalidcontext');
613         }
614         // validation for comment deletion
615         if (!empty($comment_param->commentid)) {
616             if ($comment = get_record('comments', 'id', $comment_param->commentid)) {
617                 if ($comment->commentarea != 'wikicode_page') {
618                     throw new comment_exception('invalidcommentarea');
619                 }
620                 if ($comment->contextid != $context->id) {
621                     throw new comment_exception('invalidcontext');
622                 }
623                 if ($comment->itemid != $comment_param->itemid) {
624                     throw new comment_exception('invalidcommentitemid');
625                 }
626             } else {
627                 throw new comment_exception('invalidcommentid');
628             }
629         }
630         return true;
631     }
632
633     /**
634      * Return a list of page types
635      * @param string $pagetype current page type
636      * @param stdClass $parentcontext Block's parent context
637      * @param stdClass $currentcontext Current context of block
638      */
639     function wikicode_page_type_list($pagetype, $parentcontext, $currentcontext) {
640         $module_pagetype = array(
641             'mod-wiki-*'=>get_string('page-mod-wiki-x', 'wikicode'),
642             'mod-wiki-view'=>get_string('page-mod-wiki-view', 'wikicode'),
643             'mod-wiki-comments'=>get_string('page-mod-wiki-comments', 'wikicode'),
```

```
644             'mod-wiki-history'=>get_string('page-mod-wiki-history', 'wikicode'),
645             'mod-wiki-map'=>get_string('page-mod-wiki-map', 'wikicode')
646         );
647         return $module_pagetype;
648     }
```

2.2.2. localib.php

```
1   <?php
2   require_once($CFG->dirroot . '/mod/wikicode/lib.php');
3   require_once($CFG->libdir . '/filelib.php');
4   require_once($CFG->libdir.'/adminlib.php');
5
6   define('WIKI_REFRESH_CACHE_TIME', 30); // @TODO: To be deleted.
7   define('FORMAT_CREOLE', '37');
8   define('FORMAT_NWIKI', '38');
9   define('FORMAT_WCODE', '39');
10  define('NO_VALID_RATE', '-999');
11  define('IMPROVEMENT', '+');
12  define('EQUAL', '=');
13  define('WORST', '-');
14
15  define('LOCK_TIMEOUT', 30);
16
17  /**
18   * Get a wiki instance
19   * @param int $wikiid the instance id of wiki
20   */
21  function wikicode_get_wiki($wikiid) {
22
23      return get_record('wikicode', 'id', $wikiid);
24  }
25
26  /**
27   * Get sub wiki instances with same wiki id
28   * @param int $wikiid
29   */
30  function wikicode_get_subwikis($wikiid) {
31          return get_records('wikicode_subwikis', 'wikiid', $wikiid);
32  }
33
34  /**
35   * Get a sub wiki instance by wiki id and group id
36   * @param int $wikiid
37   * @param int $groupid
38   * @return object
39   */
40  function wikicode_get_subwiki_by_group($wikiid, $groupid, $userid = 0) {
41          return get_record('wikicode_subwikis', 'wikiid', $wikiid, 'groupid',
                $groupid, 'userid', $userid);
42  }
```

```
43
44  /**
45   * Get a sub wiki instace by instance id
46   * @param int $subwikiid
47   * @return object
48   */
49  function wikicode_get_subwiki($subwikiid) {
50          return get_record('wikicode_subwikis', 'id', $subwikiid);
51
52  }
53
54  /**
55   * Add a new sub wiki instance
56   * @param int $wikiid
57   * @param int $groupid
58   * @return int $insertid
59   */
60  function wikicode_add_subwiki($wikiid, $groupid, $userid = 0) {
61
62      $record = new StdClass();
63      $record->wikiid = $wikiid;
64      $record->groupid = $groupid;
65      $record->userid = $userid;
66
67      $insertid = insert_record('wikicode_subwikis', $record);
68      return $insertid;
69  }
70
71  /**
72   * Get a wiki instance by pageid
73   * @param int $pageid
74   * @return object
75   */
76  function wikicode_get_wikicode_from_pageid($pageid) {
77
78          global $CFG;
79
80      $sql = "SELECT w.*
81              FROM {$CFG->prefix}wikicode w, {$CFG->prefix}wikicode_subwikis s, {$CFG
                    ->prefix}wikicode_pages p
82              WHERE p.id = ".$pageid." AND
83              p.subwikiid = s.id AND
84              s.wikiid = w.id";
85
86      return get_record_sql($sql);
87  }
88
89  /**
90   * Get gcc path most appearances
91   * @return object
92   */
93  function wikicode_get_gccpath() {
94
95          global $CFG;
```

```
96
97      $sql = "SELECT w.gccpath
98              FROM {$CFG->prefix}wikicode w
99              GROUP BY w.gccpath ORDER BY w.gccpath DESC
100             LIMIT 1";
101
102     return get_record_sql($sql);
103 }
104
105 /**
106  * Get mingw path most appearances
107  * @return object
108  */
109 function wikicode_get_mingwpath() {
110
111         global $CFG;
112
113     $sql = "SELECT w.mingwpath
114             FROM {$CFG->prefix}wikicode w
115             GROUP BY w.mingwpath ORDER BY w.mingwpath DESC
116             LIMIT 1";
117
118     return get_record_sql($sql);
119 }
120
121 /**
122  * Get a wiki page by pageid
123  * @param int $pageid
124  * @return object
125  */
126 function wikicode_get_page($pageid) {
127         return get_record('wikicode_pages', 'id', $pageid);
128 }
129
130 /**
131  * Get latest version of wiki page
132  * @param int $pageid
133  * @return object
134  */
135 function wikicode_get_current_version($pageid) {
136
137         global $CFG;
138
139     // @TODO: Fix this query
140     $sql = "SELECT *
141             FROM {$CFG->prefix}wikicode_versions
142             WHERE pageid = ".$pageid."
143             ORDER BY version DESC";
144
145     return array_pop(get_records_sql($sql, 0, 1));
146
147 }
148
149 /**
```

```
150     * Alias of wikicode_get_current_version
151     * @TODO, does the exactly same thing as wikicode_get_current_version, should be
            removed
152     * @param int $pageid
153     * @return object
154     */
155    function wikicode_get_last_version($pageid) {
156        return wikicode_get_current_version($pageid);
157    }
158
159    /**
160     * Get page section
161     * @param int $pageid
162     * @param string $section
163     */
164    function wikicode_get_section_page($page, $section) {
165
166        $version = wikicode_get_current_version($page->id);
167        return wikicode_parser_proxy::get_section($version->content, $version->
               contentformat, $section);
168    }
169
170    /**
171     * Get a wiki page by page title
172     * @param int $swid, sub wiki id
173     * @param string $title
174     * @return object
175     */
176    function wikicode_get_page_by_title($swid, $title) {
177            return get_record('wikicode_pages', 'subwikiid', $swid, 'title', $title);
178    }
179
180    /**
181     * Get a version record by record id
182     * @param int $versionid, the version id
183     * @return object
184     */
185    function wikicode_get_version($versionid) {
186            return get_record('wikicode_versions', 'id', $versionid);
187    }
188
189    /**
190     * Get first page of wiki instace
191     * @param int $subwikiid
192     * @param int $module, wiki instance object
193     */
194    function wikicode_get_first_page($subwikid, $module = null) {
195        global $USER, $CFG;
196
197        $sql = "SELECT p.*
198                FROM {$CFG->prefix}wikicode w, {$CFG->prefix}wikicode_subwikis s, {$CFG
                       ->prefix}wikicode_pages p
199                WHERE s.id = ".$subwikid." AND
200                s.wikiid = w.id AND
```

```
201             w.firstpagetitle = p.title AND
202             p.subwikiid = s.id";
203     return get_record_sql($sql);
204 }
205
206 function wikicode_save_section($wikipage, $sectiontitle, $sectioncontent, $userid)
        {
207
208     $wiki = wikicode_get_wikicode_from_pageid($wikipage->id);
209     $cm = get_coursemodule_from_instance('wikicode', $wiki->id);
210     $context = get_context_instance(CONTEXT_MODULE, $cm->id);
211
212     if (has_capability('mod/wikicode:editpage', $context)) {
213         $version = wikicode_get_current_version($wikipage->id);
214         $content = wikicode_parser_proxy::get_section($version->content, $version->
                contentformat, $sectiontitle, true);
215
216         $newcontent = $content[0] . $sectioncontent . $content[2];
217
218         return wikicode_save_page($wikipage, $newcontent, $userid);
219     } else {
220         return false;
221     }
222 }
223
224 /**
225  * Save page content
226  * @param object $wikipage
227  * @param string $newcontent
228  * @param int $userid
229  */
230 function wikicode_save_page($wikipage, $newcontent, $userid) {
231
232     $wiki = wikicode_get_wikicode_from_pageid($wikipage->id);
233     $cm = get_coursemodule_from_instance('wikicode', $wiki->id);
234     $context = get_context_instance(CONTEXT_MODULE, $cm->id);
235
236     if (has_capability('mod/wikicode:editpage', $context)) {
237         $version = wikicode_get_current_version($wikipage->id);
238
239
240         $version->content = $newcontent;
241         $version->userid = $userid;
242         $version->version++;
243         $version->timecreated = time();
244         $versionid = insert_record('wikicode_versions', $version);
245
246         $wikipage->timemodified = $version->timecreated;
247         $wikipage->userid = $userid;
248         $return = wikicode_refresh_cachedcontent($wikipage, $newcontent);
249
250         return $return;
251     } else {
252         return false;
```

```
253     }
254 }
255
256 /**
257  * Compile page content
258  * @param object $wikipage
259  * @param string $newcontent
260  * @param int $userid
261  */
262 function wikicode_compile_page($wikipage, $newcontent, $userid, $download) {
263
264     $wiki = wikicode_get_wikicode_from_pageid($wikipage->id);
265     $cm = get_coursemodule_from_instance('wikicode', $wiki->id);
266     $context = get_context_instance(CONTEXT_MODULE, $cm->id);
267
268     if (has_capability('mod/wikicode:editpage', $context)) {
269
270                 //Creamos el fichero
271                 $fileC = fopen("code.c","w");
272                 $codigo = wikicode_remove_tags($newcontent);
273                 $codigo = str_replace("\\", "", $codigo);
274                 fputs($fileC,$codigo);
275                 fclose($fileC);
276
277                 //Detectamos el OS
278                 $userAgent = $_SERVER[HTTP_USER_AGENT];
279                 $userAgent = strtolower ($userAgent);
280
281                 //Llamamos al compilador correspondiente
282                 if(strpos($userAgent, "windows") !== false) {
283                 $output=system($wiki->mingwpath . " code.c -o moodle.exe -w 2>
                        output.txt");
284                         $ejecutable='moodle.exe';
285                 }
286                 else {
287                         $output=system($wiki->gccpath . " code.c -o moodle.out -w
                            2> output.txt");
288                         $ejecutable='moodle.out';
289                 }
290
291                 $file_output = fopen ("output.txt", "r");
292                 $output = fread($file_output, filesize("output.txt"));
293                 $output = utf8_encode($output);
294
295                 if ( $output == '0' || $output == '') {
296                         $output = 'Compilacion correcta';
297                 } else {
298                         $wikipage->errorcompile++;
299                 }
300
301         //Guardamos los datos
302         $wikipage->cachedcompile = $newcontent;
303                 $wikipage->cachedgcc = $output;
304                 update_record('wikicode_pages', $wikipage);
```

```
305
306                 //Descargamos el ejecutable
307                 if ($output == 'Compilacion correcta' and $download == TRUE) {
308                         echo "<script language='JavaScript'>window.open('".
                                $ejecutable."')</script>";
309                 }
310
311         return array('page' => $wikipage);
312     } else {
313         return false;
314     }
315 }
316
317 function wikicode_remove_tags($newcontent) {
318
319         $codigo = str_replace("</!>", "", $newcontent);
320
321         $desde = strpos($codigo,"<!");
322
323         while (is_numeric($desde)) {
324                 $hasta = strpos($codigo, ">", $desde);
325                 $codigo = substr_replace($codigo, '', $desde, $hasta - $desde + 1);
326
327                 $desde = strpos($codigo,"<!");
328         }
329
330         return $codigo;
331 }
332
333 function wikicode_remove_tags_owner($codigo) {
334
335         global $USER;
336
337         $desde = strpos($codigo,"<!".$USER->username.">");
338
339         while (is_numeric($desde)) {
340                 $codigoaux = substr($codigo, $desde);
341                 $desdeaux = strpos($codigoaux, "</!>");
342
343                 $codigo = substr_replace($codigo, '', $desde + $desdeaux, 4);
344                 $codigo = substr_replace($codigo, '', $desde, strlen("<!".$USER->
                        username.">"));
345
346                 $desde = strpos($codigo,"<!".$USER->username.">");
347         }
348
349         return $codigo;
350 }
351
352 function wikicode_refresh_cachedcontent($page, $newcontent = null) {
353
354     $version = wikicode_get_current_version($page->id);
355     if (empty($version)) {
356         return null;
```

```
357         }
358         if (!isset($newcontent)) {
359             $newcontent = $version->content;
360         }
361 
362         $options = array('swid' => $page->subwikiid, 'pageid' => $page->id);
363 
364             if ($veresion->contentformat = 'nwiki') {
365                     $page->cachedcontent = $version->content;
366             }
367             else {
368                     $page->cachedcontent = $parseroutput['toc'] . $parseroutput['
                            parsed_text'];
369             }
370 
371         $page->timerendered = time();
372         update_record('wikicode_pages', $page);
373 
374         wikicode_refresh_page_links($page, $parseroutput['link_count']);
375 
376         return array('page' => $page, 'sections' => $parseroutput['repeated_sections'],
                'version' => $version->version);
377     }
378     /**
379      * Restore a page
380      */
381     function wikicode_restore_page($wikipage, $newcontent, $userid) {
382         $return = wikicode_save_page($wikipage, $newcontent, $userid);
383         return $return['page'];
384     }
385 
386     function wikicode_refresh_page_links($page, $links) {
387 
388         delete_records('wikicode_links', 'frompageid', $page->id);
389         foreach ($links as $linkname => $linkinfo) {
390 
391             $newlink = new stdClass();
392             $newlink->subwikiid = $page->subwikiid;
393             $newlink->frompageid = $page->id;
394 
395             if ($linkinfo['new']) {
396                 $newlink->tomissingpage = $linkname;
397             } else {
398                 $newlink->topageid = $linkinfo['pageid'];
399             }
400 
401             try {
402                 insert_record('wikicode_links', $newlink);
403             } catch (dml_exception $e) {
404                 debugging($e->getMessage());
405             }
406 
407         }
408     }
```

```
409
410 /**
411  * Create a new wiki page, if the page exists, return existing pageid
412  * @param int $swid
413  * @param string $title
414  * @param string $format
415  * @param int $userid
416  */
417 function wikicode_create_page($swid, $title, $format, $userid) {
418     global $PAGE;
419     $subwiki = wikicode_get_subwiki($swid);
420     $cm = get_coursemodule_from_instance('wikicode', $subwiki->wikiid);
421     $context = get_context_instance(CONTEXT_MODULE, $cm->id);
422     require_capability('mod/wikicode:editpage', $context);
423     // if page exists
424     if ($page = wikicode_get_page_by_title($swid, $title)) {
425         return $page->id;
426     }
427
428     // Creating a new empty version
429     $version = new stdClass();
430     $version->content = '';
431     $version->contentformat = $format;
432     $version->version = 0;
433     $version->timecreated = time();
434     $version->userid = $userid;
435
436     $versionid = null;
437     $versionid = insert_record('wikicode_versions', $version);
438
439     // Createing a new empty page
440     $page = new stdClass();
441     $page->subwikiid = $swid;
442     $page->title = $title;
443     $page->cachedcontent = '';
444     $page->timecreated = $version->timecreated;
445     $page->timemodified = $version->timecreated;
446     $page->timerendered = $version->timecreated;
447     $page->userid = $userid;
448     $page->pageviews = 0;
449     $page->readonly = 0;
450         $page->cachedcompile = '';
451         $page->cachedgcc = '';
452         $page->errorcompile = 0;
453         $page->timestartedit = 0;
454         $page->timeendedit = 0;
455
456     $pageid = insert_record('wikicode_pages', $page);
457
458     // Setting the pageid
459     $version->id = $versionid;
460     $version->pageid = $pageid;
461     update_record('wikicode_versions', $version);
462
```

```
463     wikicode_make_cache_expire($page->title);
464     return $pageid;
465 }
466 
467 function wikicode_make_cache_expire($pagename) {
468 
469         global $CFG;
470 
471     $sql = "UPDATE {$CFG->prefix}wikicode_pages
472             SET timerendered = 0
473             WHERE id IN ( SELECT l.frompageid
474                 FROM {$CFG->prefix}wikicode_links l
475                 WHERE l.tomissingpage = ".$pagename."
476             )";
477     execute_sql ($sql);
478 }
479 
480 /**
481  * Get a specific version of page
482  * @param int $pageid
483  * @param int $version
484  */
485 function wikicode_get_wikicode_page_version($pageid, $version) {
486         return get_record('wikicode_versions', 'pageid', $pageid, 'version',
            $version);
487 }
488 
489 /**
490  * Get version list
491  * @param int $pageid
492  * @param int $limitfrom
493  * @param int $limitnum
494  */
495 function wikicode_get_wikicode_page_versions($pageid, $limitfrom, $limitnum) {
496         return get_records('wikicode_versions', 'pageid', $pageid, 'version DESC',
            '*', $limitfrom, $limitnum);
497 }
498 
499 /**
500  * Count the number of page version
501  * @param int $pageid
502  */
503 function wikicode_count_wikicode_page_versions($pageid) {
504         return count_records('wikicode_versions', 'pageid', $pageid);
505 }
506 
507 /**
508  * Get linked from page
509  * @param int $pageid
510  */
511 function wikicode_get_linked_to_pages($pageid) {
512         return get_records('wikicode_links', 'frompageid', $pageid);
513 }
514 
```

```
515 /**
516  * Get linked from page
517  * @param int $pageid
518  */
519 function wikicode_get_linked_from_pages($pageid) {
520         return get_records('wikicode_links', 'topageid', $pageid);
521 }
522
523 /**
524  * Get pages which user have been edited
525  * @param int $swid
526  * @param int $userid
527  */
528 function wikicode_get_contributions($swid, $userid) {
529
530         global $CFG;
531
532     $sql = "SELECT v.*
533             FROM {$CFG->prefix}wikicode_versions v, {$CFG->prefix}wikicode_pages p
534             WHERE p.subwikiid = ".$swid." AND
535             v.pageid = p.id AND
536             v.userid = ".$userid."";
537
538     return get_records_sql($sql);
539 }
540
541 /**
542  * Get missing or empty pages in wiki
543  * @param int $swid sub wiki id
544  */
545 function wikicode_get_missing_or_empty_pages($swid) {
546
547         global $CFG;
548
549     $sql = "SELECT DISTINCT p.title, p.id, p.subwikiid
550             FROM {$CFG->prefix}wikicode w, {$CFG->prefix}wikicode_subwikis s, {$CFG
                    ->prefix}wikicode_pages p
551             WHERE s.wikiid = w.id and
552             s.id = ".$swid." and
553             w.firstpagetitle != p.title and
554             p.subwikiid = ".$swid." and
555             1 =  (SELECT count(*)
556                 FROM {$CFG->prefix}wikicode_versions v
557                 WHERE v.pageid = p.id)
558             UNION
559             SELECT DISTINCT l.tomissingpage as title, 0 as id, l.subwikiid
560             FROM {$CFG->prefix}wikicode_links l
561             WHERE l.subwikiid = ".$swid." and
562             l.topageid = 0";
563
564     return get_records_sql($sql);
565 }
566
567 /**
```

```
568     * Get pages list in wiki
569     * @param int $swid sub wiki id
570     */
571    function wikicode_get_page_list($swid) {
572            $records = get_records('wikicode_pages', 'subwikiid', $swid, 'title ASC');
573        return $records;
574    }
575
576    /**
577     * Return a list of orphaned wikis for one specific subwiki
578     * @global object
579     * @param int $swid sub wiki id
580     */
581    function wikicode_get_orphaned_pages($swid) {
582
583            global $CFG;
584
585        $sql = "SELECT p.id, p.title
586                FROM {$CFG->prefix}wikicode_pages p, {$CFG->prefix}wikicode w , {$CFG->
                       prefix}wikicode_subwikis s
587                WHERE p.subwikiid = ".$swid."
588                AND s.id = ".$swid."
589                AND w.id = s.wikiid
590                AND p.title != w.firstpagetitle
591                AND p.id NOT IN (SELECT topageid FROM {$CFG->prefix}wikicode_links
                       WHERE subwikiid = ".$swid.");";
592
593        return get_records_sql($sql);
594    }
595
596    /**
597     * Search wiki title
598     * @param int $swid sub wiki id
599     * @param string $search
600     */
601    function wikicode_search_title($swid, $search) {
602
603        return get_records_select('wikicode_pages', "subwikiid = ? AND title LIKE ?",
               $swid, '%'.$search.'%');
604    }
605
606    /**
607     * Search wiki content
608     * @param int $swid sub wiki id
609     * @param string $search
610     */
611    function wikicode_search_content($swid, $search) {
612
613        return get_records_select('wikicode_pages', "subwikiid = ? AND cachedcontent
               LIKE ?", $swid, '%'.$search.'%');
614    }
615
616    /**
617     * Search wiki title and content
```

```
618     * @param int $swid sub wiki id
619     * @param string $search
620     */
621    function wikicode_search_all($swid, $search) {
622
623        return get_records_select('wikicode_pages', "subwikiid = ? AND (cachedcontent
               LIKE ? OR title LIKE ?)", $swid, '%'.$search.'%', '%'.$search.'%');
624    }
625
626    /**
627     * Get user data
628     */
629    function wikicode_get_user_info($userid) {
630            return get_record('user', 'id', $userid);
631    }
632
633    /**
634     * Increase page view nubmer
635     * @param int $page, database record
636     */
637    function wikicode_increment_pageviews($page) {
638
639        $page->pageviews++;
640        update_record('wikicode_pages', $page);
641    }
642
643    //-------------------------------------------------------------
644    //-------------------------------------------------------------
645
646    /**
647     * Text format supported by wiki module
648     */
649    function wikicode_get_formats() {
650        return array('wcode');
651    }
652
653    /**
654     * Parses a string with the wiki markup language in $markup.
655     *
656     * @return Array or false when something wrong has happened.
657     *
658     * Returned array contains the following fields:
659     *     'parsed_text', String. Contains the parsed wiki content.
660     *     'unparsed_text', String. Constains the original wiki content.
661     *     'link_count', Array of 'destination', ..., 'new', "is new?"). Contains the
           internal wiki links found in the wiki content.
662     *      'deleted_sections', the list of deleted sections.
663     *              '' =>
664     *
665     **/
666    function wikicode_parse_content($markup, $pagecontent, $options = array()) {
667        global $PAGE;
668
669        $subwiki = wikicode_get_subwiki($options['swid']);
```

```
670         $cm = get_coursemodule_from_instance("wikicode", $subwiki->wikiid);
671         $context = get_context_instance(CONTEXT_MODULE, $cm->id);
672 
673         $parser_options = array(
674             'link_callback' => '/mod/wikicode/locallib.php:wiki_parser_link',
675             'link_callback_args' => array('swid' => $options['swid']),
676             'table_callback' => '/mod/wikicode/locallib.php:wiki_parser_table',
677             'real_path_callback' => '/mod/wikicode/locallib.php:wiki_parser_real_path',
678             'real_path_callback_args' => array(
679                 'context' => $context,
680                 'component' => 'mod_wikicode',
681                 'filearea' => 'attachments',
682                 'subwikiid'=> $subwiki->id,
683                 'pageid' => $options['pageid']
684             ),
685             'pageid' => $options['pageid'],
686             'pretty_print' => (isset($options['pretty_print']) && $options['
                    pretty_print']),
687             'printable' => (isset($options['printable']) && $options['printable'])
688         );
689 
690         return wikicode_parser_proxy::parse($pagecontent, $markup, $parser_options);
691     }
692 
693     /**
694      * This function is the parser callback to parse wiki links.
695      *
696      * It returns the necesary information to print a link.
697      *
698      * NOTE: Empty pages and non-existent pages must be print in red color.
699      *
700      * @param link name of a page
701      * @param $options
702      *
703      * @return
704      *
705      * @TODO Doc return and options
706      */
707     function wikicode_parser_link($link, $options = null) {
708         global $CFG;
709 
710         if (is_object($link)) {
711             $parsedlink = array('content' => $link->title, 'url' => $CFG->wwwroot . '/
                    mod/wikicode/view.php?pageid=' . $link->id, 'new' => false, 'link_info'
                     => array('link' => $link->title, 'pageid' => $link->id, 'new' => false
                    ));
712 
713             $version = wikicode_get_current_version($link->id);
714             if ($version->version == 0) {
715                 $parsedlink['new'] = true;
716             }
717             return $parsedlink;
718         } else {
719             $swid = $options['swid'];
```

```
720
721         if ($page = wikicode_get_page_by_title($swid, $link)) {
722             $parsedlink = array('content' => $link, 'url' => $CFG->wwwroot . '/mod/
                wikicode/view.php?pageid=' . $page->id, 'new' => false, 'link_info'
                 => array('link' => $link, 'pageid' => $page->id, 'new' => false));
723
724             $version = wikicode_get_current_version($page->id);
725             if ($version->version == 0) {
726                 $parsedlink['new'] = true;
727             }
728
729             return $parsedlink;
730
731         } else {
732             return array('content' => $link, 'url' => $CFG->wwwroot . '/mod/
                wikicode/create.php?swid=' . $swid . '&amp;title=' . urlencode(
                $link) . '&amp;action=new', 'new' => true, 'link_info' => array('
                link' => $link, 'new' => true, 'pageid' => 0));
733         }
734     }
735 }
736
737 /**
738  * Returns the table fully parsed (HTML)
739  *
740  * @return HTML for the table $table
741  *
742  **/
743 function wikicode_parser_table($table) {
744     global $OUTPUT;
745
746     $htmltable = new html_table();
747
748     $headers = $table[0];
749     $htmltable->head = array();
750     foreach ($headers as $h) {
751         $htmltable->head[] = $h[1];
752     }
753
754     array_shift($table);
755     $htmltable->data = array();
756     foreach ($table as $row) {
757         $row_data = array();
758         foreach ($row as $r) {
759             $row_data[] = $r[1];
760         }
761         $htmltable->data[] = $row_data;
762     }
763
764     return html_writer::table($htmltable);
765 }
766
767 /**
768  * Returns an absolute path link, unless there is no such link.
```

```
769     *
770     * @param string $url Link's URL or filename
771     * @param stdClass $context filearea params
772     * @param string $component The component the file is associated with
773     * @param string $filearea The filearea the file is stored in
774     * @param int $swid Sub wiki id
775     *
776     * @return string URL for files full path
777     */
778
779    function wikicode_parser_real_path($url, $context, $component, $filearea, $swid) {
780        global $CFG;
781
782        if (preg_match("/^(?:http|ftp)s?\:\/\//", $url)) {
783            return $url;
784        } else {
785
786            $file = 'pluginfile.php';
787            if (!$CFG->slasharguments) {
788                $file = $file . '?file=';
789            }
790            $baseurl = "$CFG->wwwroot/$file/{$context->id}/$component/$filearea/$swid/"
                    ;
791            // it is a file in current file area
792            return $baseurl . $url;
793        }
794    }
795
796    /**
797     * Returns the token used by a wiki language to represent a given tag or "object" (
          bold -> **)
798     *
799     * @return A string when it has only one token at the beginning (f. ex. lists). An
          array composed by 2 strings when it has 2 tokens, one at the beginning and one
           at the end (f. ex. italics). Returns false otherwise.
800     **/
801    function wikicode_parser_get_token($markup, $name) {
802
803        return wikicode_parser_proxy::get_token($name, $markup);
804    }
805
806    /**
807     * Checks if current user can view a subwiki
808     *
809     * @param $subwiki
810     */
811    function wikicode_user_can_view($subwiki) {
812        global $USER;
813
814        $wiki = wikicode_get_wiki($subwiki->wikiid);
815        $cm = get_coursemodule_from_instance('wikicode', $wiki->id);
816        $context = get_context_instance(CONTEXT_MODULE, $cm->id);
817
818        // Working depending on activity groupmode
```

```
819     switch (groups_get_activity_groupmode($cm)) {
820     case NOGROUPS:
821
822         if ($wiki->wikimode == 'collaborative') {
823             // Collaborative Mode:
824             // There is one wiki for all the class.
825             //
826             // Only view capbility needed
827             return has_capability('mod/wikicode:viewpage', $context);
828         } else if ($wiki->wikimode == 'individual') {
829             // Individual Mode:
830             // Each person owns a wiki.
831             if ($subwiki->userid == $USER->id) {
832                 // Only the owner of the wiki can view it
833                 return has_capability('mod/wikicode:viewpage', $context);
834             } else { // User has special capabilities
835                 // User must have:
836                 //      mod/wiki:viewpage capability
837                 // and
838                 //      mod/wiki:managewiki capability
839                 $view = has_capability('mod/wikicode:viewpage', $context);
840                 $manage = has_capability('mod/wikicode:managewiki', $context);
841
842                 return $view && $manage;
843             }
844         } else {
845             //Error
846             return false;
847         }
848     case SEPARATEGROUPS:
849         // Collaborative and Individual Mode
850         //
851         // Collaborative Mode:
852         //      There is one wiki per group.
853         // Individual Mode:
854         //      Each person owns a wiki.
855         if ($wiki->wikimode == 'collaborative' || $wiki->wikimode == 'individual')
                {
856             // Only members of subwiki group could view that wiki
857             if ($subwiki->groupid == groups_get_activity_group($cm)) {
858                 // Only view capability needed
859                 return has_capability('mod/wikicode:viewpage', $context);
860
861             } else { // User is not part of that group
862                 // User must have:
863                 //      mod/wiki:managewiki capability
864                 // or
865                 //      moodle/site:accessallgroups capability
866                 // and
867                 //      mod/wiki:viewpage capability
868                 $view = has_capability('mod/wikicode:viewpage', $context);
869                 $manage = has_capability('mod/wikicode:managewiki', $context);
870                 $access = has_capability('moodle/site:accessallgroups', $context);
871                 return ($manage || $access) && $view;
```

```
872             }
873         } else {
874             //Error
875             return false;
876         }
877     case VISIBLEGROUPS:
878         // Collaborative and Individual Mode
879         //
880         // Collaborative Mode:
881         //      There is one wiki per group.
882         // Individual Mode:
883         //      Each person owns a wiki.
884         if ($wiki->wikimode == 'collaborative' || $wiki->wikimode == 'individual')
                {
885             // Everybody can read all wikis
886             //
887             // Only view capability needed
888             return has_capability('mod/wikicode:viewpage', $context);
889         } else {
890             //Error
891             return false;
892         }
893     default: // Error
894         return false;
895     }
896 }
897
898 /**
899  * Checks if current user can edit a subwiki
900  *
901  * @param $subwiki
902  */
903 function wikicode_user_can_edit($subwiki) {
904     global $USER;
905
906     $wiki = wikicode_get_wiki($subwiki->wikiid);
907     $cm = get_coursemodule_from_instance('wikicode', $wiki->id);
908     $context = get_context_instance(CONTEXT_MODULE, $cm->id);
909
910     // Working depending on activity groupmode
911     switch (groups_get_activity_groupmode($cm)) {
912     case NOGROUPS:
913
914         if ($wiki->wikimode == 'collaborative') {
915             // Collaborative Mode:
916             // There is a wiki for all the class.
917             //
918             // Only edit capbility needed
919             return has_capability('mod/wikicode:editpage', $context);
920         } else if ($wiki->wikimode == 'individual') {
921             // Individual Mode
922             // There is a wiki per user
923
924             // Only the owner of that wiki can edit it
```

```
925             if ($subwiki->userid == $USER->id) {
926                 return has_capability('mod/wikicode:editpage', $context);
927             } else { // Current user is not the owner of that wiki.
928 
929                 // User must have:
930                 //      mod/wiki:editpage capability
931                 // and
932                 //      mod/wiki:managewiki capability
933                 $edit = has_capability('mod/wikicode:editpage', $context);
934                 $manage = has_capability('mod/wikicode:managewiki', $context);
935 
936                 return $edit && $manage;
937             }
938         } else {
939             //Error
940             return false;
941         }
942     case SEPARATEGROUPS:
943         if ($wiki->wikimode == 'collaborative') {
944             // Collaborative Mode:
945             // There is one wiki per group.
946             //
947             // Only members of subwiki group could edit that wiki
948             if ($subwiki->groupid == groups_get_activity_group($cm)) {
949                 // Only edit capability needed
950                 return has_capability('mod/wikicode:editpage', $context);
951             } else { // User is not part of that group
952                 // User must have:
953                 //      mod/wiki:managewiki capability
954                 // and
955                 //      moodle/site:accessallgroups capability
956                 // and
957                 //      mod/wiki:editpage capability
958                 $manage = has_capability('mod/wikicode:managewiki', $context);
959                 $access = has_capability('moodle/site:accessallgroups', $context);
960                 $edit = has_capability('mod/wikicode:editpage', $context);
961                 return $manage && $access && $edit;
962             }
963         } else if ($wiki->wikimode == 'individual') {
964             // Individual Mode:
965             // Each person owns a wiki.
966             //
967             // Only the owner of that wiki can edit it
968             if ($subwiki->userid == $USER->id) {
969                 return has_capability('mod/wikicode:editpage', $context);
970             } else { // Current user is not the owner of that wiki.
971                 // User must have:
972                 //      mod/wiki:managewiki capability
973                 // and
974                 //      moodle/site:accessallgroups capability
975                 // and
976                 //      mod/wiki:editpage capability
977                 $manage = has_capability('mod/wikicode:managewiki', $context);
978                 $access = has_capability('moodle/site:accessallgroups', $context);
```

```
979                     $edit = has_capability('mod/wikicode:editpage', $context);
980                     return $manage && $access && $edit;
981                 }
982             } else {
983                 //Error
984                 return false;
985             }
986         case VISIBLEGROUPS:
987             if ($wiki->wikimode == 'collaborative') {
988                 // Collaborative Mode:
989                 // There is one wiki per group.
990                 //
991                 // Only members of subwiki group could edit that wiki
992                 if (groups_is_member($subwiki->groupid)) {
993                     // Only edit capability needed
994                     return has_capability('mod/wikicode:editpage', $context);
995                 } else { // User is not part of that group
996                     // User must have:
997                     //      mod/wiki:managewiki capability
998                     // and
999                     //      mod/wiki:editpage capability
1000                    $manage = has_capability('mod/wikicode:managewiki', $context);
1001                    $edit = has_capability('mod/wikicode:editpage', $context);
1002                    return $manage && $edit;
1003                }
1004            } else if ($wiki->wikimode == 'individual') {
1005                // Individual Mode:
1006                // Each person owns a wiki.
1007                //
1008                // Only the owner of that wiki can edit it
1009                if ($subwiki->userid == $USER->id) {
1010                    return has_capability('mod/wikicode:editpage', $context);
1011                } else { // Current user is not the owner of that wiki.
1012                    // User must have:
1013                    //      mod/wiki:managewiki capability
1014                    // and
1015                    //      mod/wiki:editpage capability
1016                    $manage = has_capability('mod/wikicode:managewiki', $context);
1017                    $edit = has_capability('mod/wikicode:editpage', $context);
1018                    return $manage && $edit;
1019                }
1020            } else {
1021                //Error
1022                return false;
1023            }
1024        default: // Error
1025            return false;
1026        }
1027    }
1028
1029    //----------------
1030    // Locks
1031    //----------------
1032
```

```
1033  /**
1034   * Checks if a page-section is locked.
1035   *
1036   * @return true if the combination of section and page is locked, FALSE otherwise.
1037   */
1038  function wikicode_is_page_section_locked($pageid, $userid, $section = null) {
1039
1040      $sql = "pageid = ".$pageid." AND lockedat > ".time()." AND userid != ".$userid;
1041
1042      if (!empty($section)) {
1043          $sql .= " AND (sectionname = ".$section." OR sectionname IS null)";
1044      }
1045
1046      return record_exists_select('wikicode_locks', $sql);
1047  }
1048
1049  /**
1050   * Inserts or updates a wikicode_locks record.
1051   */
1052  function wikicode_set_lock($pageid, $userid, $section = null, $insert = false) {
1053
1054      if (wikicode_is_page_section_locked($pageid, $userid, $section)) {
1055          return false;
1056      }
1057
1058      $lock = get_record('wikicode_locks', 'pageid', $pageid, 'userid', $userid, '
            sectionname', $section);
1059
1060      if (!empty($lock)) {
1061          update_record('wikicode_locks', array('id' => $lock->id, 'lockedat' => time
                () + LOCK_TIMEOUT));
1062      } else if ($insert) {
1063          insert_record('wikicode_locks', array('pageid' => $pageid, 'sectionname' =>
                $section, 'userid' => $userid, 'lockedat' => time() + 30));
1064      }
1065
1066      return true;
1067  }
1068
1069  /**
1070   * Deletes wikicode_locks that are not in use. (F.Ex. after submitting the changes)
        . If no userid is present, it deletes ALL the wikicode_locks of a specific
        page.
1071   */
1072  function wikicode_delete_locks($pageid, $userid = null, $section = null,
        $delete_from_db = true, $delete_section_and_page = false) {
1073
1074
1075  }
1076
1077  /**
1078   * Deletes wikicode_locks that expired 1 hour ago.
1079   */
1080  function wikicode_delete_old_locks() {
```

```
1081
1082        delete_records_select('wikicode_locks', "lockedat < ?", time() - 3600);
1083    }
1084
1085    /**
1086     * Deletes wikicode_links. It can be sepecific link or links attached in subwiki
1087     *
1088     * @global mixed $DB database object
1089     * @param int $linkid id of the link to be deleted
1090     * @param int $topageid links to the specific page
1091     * @param int $frompageid links from specific page
1092     * @param int $subwikiid links to subwiki
1093     */
1094    function wikicode_delete_links($linkid = null, $topageid = null, $frompageid = null
         , $subwikiid = null) {
1095        $params = array();
1096
1097        // if link id is givien then don't check for anything else
1098        if (!empty($linkid)) {
1099            $params['id'] = $linkid;
1100        } else {
1101            if (!empty($topageid)) {
1102                $params['topageid'] = $topageid;
1103            }
1104            if (!empty($frompageid)) {
1105                $params['frompageid'] = $frompageid;
1106            }
1107            if (!empty($subwikiid)) {
1108                $params['subwikiid'] = $subwikiid;
1109            }
1110        }
1111
1112        //Delete links if any params are passed, else nothing to delete.
1113        if (!empty($params)) {
1114            delete_records('wikicode_links', $params);
1115        }
1116    }
1117
1118    /**
1119     * Delete wiki synonyms related to subwikiid or page
1120     *
1121     * @param int $subwikiid id of sunbwiki
1122     * @param int $pageid id of page
1123     */
1124    function wikicode_delete_synonym($subwikiid, $pageid = null) {
1125
1126        if (!is_null($pageid)) {
1127            $params['pageid'] = $pageid;
1128                    delete_records('wikicode_synonyms', 'subwikiid', $subwikiid, '
                         pageid', $pageid);
1129        } else {
1130            delete_records('wikicode_synonyms', 'subwikiid', $subwikiid);
1131            }
1132    }
```

```
1133
1134  /**
1135   * Delete pages and all related data
1136   *
1137   * @param mixed $context context in which page needs to be deleted.
1138   * @param mixed $pageids id's of pages to be deleted
1139   * @param int $subwikiid id of the subwiki for which all pages should be deleted
1140   */
1141  function wikicode_delete_pages($context, $pageids = null, $subwikiid = null) {
1142
1143      if (!empty($pageids) && is_int($pageids)) {
1144         $pageids = array($pageids);
1145      } else if (!empty($subwikiid)) {
1146          $pageids = wikicode_get_page_list($subwikiid);
1147      }
1148
1149      //If there is no pageid then return as we can't delete anything.
1150      if (empty($pageids)) {
1151          return;
1152      }
1153
1154      /// Delete page and all it's relevent data
1155      foreach ($pageids as $pageid) {
1156          if (is_object($pageid)) {
1157              $pageid = $pageid->id;
1158          }
1159
1160          //Delete page comments
1161          $comments = wikicode_get_comments($context->id, $pageid);
1162          foreach ($comments as $commentid => $commentvalue) {
1163              wikicode_delete_comment($commentid, $context, $pageid);
1164          }
1165
1166          //Delete page tags
1167          $tags = tag_get_tags_array('wikicode_pages', $pageid);
1168          foreach ($tags as $tagid => $tagvalue) {
1169              tag_delete_instance('wikicode_pages', $pageid, $tagid);
1170          }
1171
1172          //Delete Synonym
1173          wikicode_delete_synonym($subwikiid, $pageid);
1174
1175          //Delete all page versions
1176          wikicode_delete_page_versions(array($pageid=>array(0)));
1177
1178          //Delete all page locks
1179          wikicode_delete_locks($pageid);
1180
1181          //Delete all page links
1182          wikicode_delete_links(null, $pageid);
1183
1184          //Delete page
1185                  delete_records('wikicode_pages', 'id', $pageid);
1186      }
```

```
1187 }
1188
1189 /**
1190  * Delete specificed versions of a page or versions created by users
1191  * if version is 0 then it will remove all versions of the page
1192  *
1193  * @param array $deleteversions delete versions for a page
1194  */
1195 function wikicode_delete_page_versions($deleteversions) {
1196
1197     /// delete page-versions
1198     foreach ($deleteversions as $id => $versions) {
1199         foreach ($versions as $version) {
1200             //If version = 0, then remove all versions of this page, else remove
1201             //specified version
1202             if ($version != 0) {
1203                 delete_records('wikicode_versions', 'pageid', $id, 'version',
                        $version);
1204             } else {
1205                 delete_records('wikicode_versions', 'pageid', $id);
1206                         }
1207         }
1208     }
1209 }
1210
1211 function wikicode_get_comment($commentid){
1212         return get_record('comments', 'id', $commentid);
1213 }
1214
1215 /**
1216  * Returns all comments by context and pageid
1217  *
1218  * @param $context. Current context
1219  * @param $pageid. Current pageid
1220  **/
1221 function wikicode_get_comments($contextid, $pageid) {
1222
1223     return get_records('comments', 'contextid', $contextid, 'itemid', $pageid, '
            commentarea', 'wikicode_page');
1224 }
1225
1226 /**
1227  * Add comments ro database
1228  *
1229  * @param object $context. Current context
1230  * @param int $pageid. Current pageid
1231  * @param string $content. Content of the comment
1232  * @param string editor. Version of editor we are using.
1233  **/
1234 function wikicode_add_comment($context, $pageid, $content, $editor) {
1235     global $CFG;
1236     require_once($CFG->dirroot . '/comment/lib.php');
1237
1238     list($context, $course, $cm) = get_context_info_array($context->id);
```

```
1239        $cmt = new stdclass();
1240        $cmt->context = $context;
1241        $cmt->itemid = $pageid;
1242        $cmt->area = 'wikicode_page';
1243        $cmt->course = $course;
1244        $cmt->component = 'mod_wikicode';
1245
1246        $manager = new comment($cmt);
1247
1248        if ($editor == 'creole') {
1249            $manager->add($content, FORMAT_CREOLE);
1250        } else if ($editor == 'html') {
1251            $manager->add($content, FORMAT_HTML);
1252        } else if ($editor == 'nwiki') {
1253            $manager->add($content, FORMAT_NWIKI);
1254        } else if ($editor == 'wcode' ) {
1255            $manager->add($content, FORMAT_WCODE);
1256        }
1257
1258    }
1259
1260    /**
1261     * Delete comments from database
1262     *
1263     * @param $idcomment. Id of comment which will be deleted
1264     * @param $context. Current context
1265     * @param $pageid. Current pageid
1266     **/
1267    function wikicode_delete_comment($idcomment, $context, $pageid) {
1268        global $CFG;
1269        require_once($CFG->dirroot . '/comment/lib.php');
1270
1271        list($context, $course, $cm) = get_context_info_array($context->id);
1272        $cmt = new stdClass();
1273        $cmt->context = $context;
1274        $cmt->itemid = $pageid;
1275        $cmt->area = 'wikicode_page';
1276        $cmt->course = $course;
1277        $cmt->component = 'mod_wikicode';
1278
1279        $manager = new comment($cmt);
1280        $manager->delete($idcomment);
1281
1282    }
1283
1284    /**
1285     * Delete al comments from wiki
1286     *
1287     **/
1288    function wikicode_delete_comments_wiki() {
1289        global $PAGE;
1290
1291        $cm = $PAGE->cm;
1292        $context = get_context_instance(CONTEXT_MODULE, $cm->id);
```

```
1293
1294        $table = 'comments';
1295        $select = 'contextid = ?';
1296
1297        delete_records_select($table, $select, $context->id);
1298
1299    }
1300
1301    function wikicode_add_progress($pageid, $oldversionid, $versionid, $progress) {
1302            for ($v = $oldversionid + 1; $v <= $versionid; $v++) {
1303            $user = wikicode_get_wikicode_page_id($pageid, $v);
1304
1305            insert_record('wikicode_progress', array('userid' => $user->userid, 'pageid
                ' => $pageid, 'versionid' => $v, 'progress' => $progress));
1306        }
1307    }
1308
1309    function wikicode_get_wikicode_page_id($pageid, $id) {
1310            return get_record('wikicode_versions', 'pageid', $pageid, 'id', $id);
1311    }
1312
1313    function wikicode_print_page_content($page, $context, $subwikiid) {
1314        global $OUTPUT, $CFG;
1315
1316        if ($page->timerendered + WIKI_REFRESH_CACHE_TIME < time()) {
1317            $content = wikicode_refresh_cachedcontent($page);
1318            $page = $content['page'];
1319        }
1320
1321        if (isset($content)) {
1322            $box = '';
1323            foreach ($content['sections'] as $s) {
1324                $box .= '<p>' . get_string('repeatedsection', 'wikicode', $s) . '</p>';
1325            }
1326
1327            if (!empty($box)) {
1328                echo $OUTPUT->box($box);
1329            }
1330        }
1331
1332        $html = format_text(wikicode_remove_tags($page->cachedcontent), FORMAT_PLAIN,
            array('overflowdiv'=>true));
1333
1334            print_simple_box_start('center','70%','','20');
1335        print_box($html);
1336            print_simple_box_end();
1337
1338        wikicode_increment_pageviews($page);
1339
1340    }
1341
1342    /**
1343     * This function trims any given text and returns it with some dots at the end
1344     *
```

```
1345     * @param string $text
1346     * @param string $limit
1347     *
1348     * @return string
1349     */
1350    function wikicode_trim_string($text, $limit = 25) {
1351
1352        if (textlib::strlen($text) > $limit) {
1353            $text = textlib::substr($text, 0, $limit) . '...';
1354        }
1355
1356        return $text;
1357    }
1358
1359    /**
1360     * Prints default edit form fields and buttons
1361     *
1362     * @param string $format Edit form format (html, creole...)
1363     * @param integer $version Version number. A negative number means no versioning.
1364     */
1365
1366    function wikicode_print_edit_form_default_fields($format, $pageid, $version = -1,
         $upload = false, $deleteuploads = array()) {
1367        global $CFG, $PAGE, $OUTPUT;
1368
1369        echo '<input type="hidden" name="sesskey" value="' . sesskey() . '" />';
1370
1371        if ($version >= 0) {
1372            echo '<input type="hidden" name="version" value="' . $version . '" />';
1373        }
1374
1375        echo '<input type="hidden" name="format" value="' . $format . '"/>';
1376
1377        //attachments
1378        require_once($CFG->dirroot . '/lib/form/filemanager.php');
1379
1380        $filemanager = new MoodleQuickForm_filemanager('attachments', get_string('
             wikiattachments', 'wikicode'), array('id' => 'attachments'), array('subdirs
             ' => false, 'maxfiles' => 99, 'maxbytes' => $CFG->maxbytes));
1381
1382        $value = file_get_submitted_draft_itemid('attachments');
1383        if (!empty($value) && !$upload) {
1384            $filemanager->setValue($value);
1385        }
1386
1387        echo "<fieldset class=\"wiki-upload-section clearfix\"><legend class=\"ftoggler
             \">" . get_string("uploadtitle", 'wikicode') . "</legend>";
1388
1389        echo $OUTPUT->container_start('mdl-align wiki-form-center aaaaa');
1390        print $filemanager->toHtml();
1391        echo $OUTPUT->container_end();
1392
1393        $cm = $PAGE->cm;
1394        $context = get_context_instance(CONTEXT_MODULE, $cm->id);
```

```
1395
1396        echo $OUTPUT->container_start('mdl-align wiki-form-center wiki-upload-table');
1397        wikicode_print_upload_table($context, 'wikicode_upload', $pageid,
                $deleteuploads);
1398        echo $OUTPUT->container_end();
1399
1400        echo "</fieldset>";
1401
1402        echo '<input class="wiki_button" type="submit" name="editoption" value="' .
                get_string('save', 'wikicode') . '"/>';
1403        echo '<input class="wiki_button" type="submit" name="editoption" value="' .
                get_string('upload', 'wikicode') . '"/>';
1404        echo '<input class="wiki_button" type="submit" name="editoption" value="' .
                get_string('preview') . '"/>';
1405        echo '<input class="wiki_button" type="submit" name="editoption" value="' .
                get_string('cancel') . '" />';
1406    }
1407
1408    /**
1409     * Prints a table with the files attached to a wiki page
1410     * @param object $context
1411     * @param string $filearea
1412     * @param int $fileitemid
1413     * @param array deleteuploads
1414     */
1415    function wikicode_print_upload_table($context, $filearea, $fileitemid,
            $deleteuploads = array()) {
1416        global $CFG, $OUTPUT;
1417
1418        $htmltable = new html_table();
1419
1420        $htmltable->head = array(get_string('deleteupload', 'wikicode'), get_string('
                uploadname', 'wikicode'), get_string('uploadactions', 'wiki'));
1421
1422        $fs = get_file_storage();
1423        $files = $fs->get_area_files($context->id, 'mod_wikicode', $filearea,
                $fileitemid); //TODO: this is weird (skodak)
1424
1425        foreach ($files as $file) {
1426            if (!$file->is_directory()) {
1427                $checkbox = '<input type="checkbox" name="deleteupload[]", value="' .
                        $file->get_pathnamehash() . '"';
1428
1429                if (in_array($file->get_pathnamehash(), $deleteuploads)) {
1430                    $checkbox .= ' checked="checked"';
1431                }
1432
1433                $checkbox .= " />";
1434
1435                $htmltable->data[] = array($checkbox, '<a href="' . file_encode_url(
                        $CFG->wwwroot . '/pluginfile.php', '/' . $context->id . '/
                        wikicode_upload/' . $fileitemid . '/' . $file->get_filename()) . '
                        ">' . $file->get_filename() . '</a>', "");
1436            }
```

```
1437        }
1438
1439        print '<h3 class="upload-table-title">' . get_string('uploadfiletitle', '
                wikicode') . "</h3>";
1440        print html_writer::table($htmltable);
1441    }
1442
1443    /**
1444     * Generate wiki's page tree
1445     *
1446     * @param $page. A wiki page object
1447     * @param $node. Starting navigation_node
1448     * @param $keys. An array to store keys
1449     * @return an array with all tree nodes
1450     */
1451    function wikicode_build_tree($page, $node, &$keys) {
1452        $content = array();
1453        static $icon;
1454        $icon = new pix_icon('f/odt', '');
1455        $pages = wikicode_get_linked_pages($page->id);
1456        foreach ($pages as $p) {
1457            $key = $page->id . ':' . $p->id;
1458            if (in_array($key, $keys)) {
1459                break;
1460            }
1461            array_push($keys, $key);
1462            $l = wikicode_parser_link($p);
1463            $link = new moodle_url('/mod/wikicode/view.php', array('pageid' => $p->id))
                    ;
1464            $nodeaux = $node->add($p->title, $link, null, null, null, $icon);
1465            if ($l['new']) {
1466                $nodeaux->add_class('wikicode_newentry');
1467            }
1468            wikicode_build_tree($p, $nodeaux, $keys);
1469        }
1470        $content[] = $node;
1471        return $content;
1472    }
1473
1474    /**
1475     * Get linked pages from page
1476     * @param int $pageid
1477     */
1478    function wikicode_get_linked_pages($pageid) {
1479
1480            global $CFG;
1481
1482        $sql = "SELECT p.id, p.title
1483                FROM {$CFG->prefix}wikicode_pages p
1484                JOIN {$CFG->prefix}wikicode_links l ON l.topageid = p.id
1485                WHERE l.frompageid = ".$pageid."
1486                ORDER BY p.title ASC";
1487        return get_records_sql($sql);
1488    }
```

```
1489
1490  /**
1491   * Get updated pages from wiki
1492   * @param int $pageid
1493   */
1494  function wikicode_get_updated_pages_by_subwiki($swid) {
1495      global $USER, $CFG;
1496
1497      $sql = "SELECT *
1498              FROM {$CFG->prefix}wikicode_pages
1499              WHERE subwikiid = ".$swid." AND timemodified > ".$USER->lastlogin."
1500              ORDER BY timemodified DESC";
1501      return get_records_sql($sql);
1502  }
```

Capítulo 3

Módulo para Moodle 2.x

3.1. Front-End

3.1.1. admin.php

```
<?php

/**
 * Delete wiki pages or versions
 *
 * This will show options for deleting wiki pages or purging page versions
 * If user have wiki:managewiki ability then only this page will show delete
 * options
 *
 * @package mod-wikicode
 * @license http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
 */

require_once('../../config.php');
require_once($CFG->dirroot . '/mod/wikicode/lib.php');
require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
require_once($CFG->dirroot . '/mod/wikicode/pagelib.php');

$pageid = required_param('pageid', PARAM_INT); // Page ID
$delete = optional_param('delete', 0, PARAM_INT); // ID of the page to be deleted.
$option = optional_param('option', 1, PARAM_INT); // Option ID
$listall = optional_param('listall', 0, PARAM_INT); // list all pages
$toversion = optional_param('toversion', 0, PARAM_INT); // max version to be
    deleted
$fromversion = optional_param('fromversion', 0, PARAM_INT); // min version to be
    deleted

if (!$page = wikicode_get_page($pageid)) {
    print_error('incorrectpageid', 'wikicode');
```

```
28  }
29  if (!$subwiki = wikicode_get_subwiki($page->subwikiid)) {
30      print_error('incorrectsubwikiid', 'wikicode');
31  }
32  if (!$cm = get_coursemodule_from_instance("wikicode", $subwiki->wikiid)) {
33      print_error('invalidcoursemodule');
34  }
35  $course = $DB->get_record('course', array('id' => $cm->course), '*', MUST_EXIST);
36  if (!$wiki = wikicode_get_wiki($subwiki->wikiid)) {
37      print_error('incorrectwikiid', 'wikicode');
38  }
39
40  require_login($course->id, true, $cm);
41
42
43  $context = get_context_instance(CONTEXT_MODULE, $cm->id);
44  require_capability('mod/wikicode:managewiki', $context);
45  add_to_log($course->id, "wikicode", "admin", "admin.php?id=$cm->id", "$wiki->id");
46
47  //Delete page if a page ID to delete was supplied
48  if (!empty($delete) && confirm_sesskey()) {
49      wikicode_delete_pages($context, $delete, $page->subwikiid);
50      //when current wiki page is deleted, then redirect user to create that page, as
51      //current pageid is invalid after deletion.
52      if ($pageid == $delete) {
53          $params = array('swid' => $page->subwikiid, 'title' => $page->title);
54          $url = new moodle_url('/mod/wikicode/create.php', $params);
55          redirect($url);
56      }
57  }
58
59  //delete version if toversion and fromversion are set.
60  if (!empty($toversion) && !empty($fromversion) && confirm_sesskey()) {
61      //make sure all versions should not be deleted...
62      $versioncount = wikicode_count_wikicode_page_versions($pageid);
63      $versioncount -= 1; //ignore version 0
64      $totalversionstodelete = $toversion - $fromversion;
65      $totalversionstodelete += 1; //added 1 as toversion should be included
66
67      if (($totalversionstodelete >= $versioncount) || ($versioncount <= 1)) {
68          print_error('incorrectdeleteversions', 'wikicode');
69      } else {
70          $versions = array();
71          for ($i = $fromversion; $i <= $toversion; $i++) {
72              //Add all version to deletion list which exist
73              if (wikicode_get_wikicode_page_version($pageid, $i)) {
74                  array_push($versions, $i);
75              }
76          }
77          $purgeversions[$pageid] = $versions;
78          wikicode_delete_page_versions($purgeversions);
79      }
80  }
81
```

```
82  //show actual page
83  $wikipage = new page_wikicode_admin($wiki, $subwiki, $cm);
84
85  $wikipage->set_page($page);
86  $wikipage->print_header();
87  $wikipage->set_view($option, empty($listall)?true:false);
88  $wikipage->print_content();
89
90  $wikipage->print_footer();
```

3.1.2. comments.php

```
1   <?php
2   require_once('../../config.php');
3
4   require_once($CFG->dirroot . '/mod/wikicode/lib.php');
5   require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
6   require_once($CFG->dirroot . '/mod/wikicode/pagelib.php');
7
8   $pageid = required_param('pageid', PARAM_TEXT);
9
10  if (!$page = wikicode_get_page($pageid)) {
11      print_error('incorrectpageid', 'wikicode');
12  }
13
14  if (!$subwiki = wikicode_get_subwiki($page->subwikiid)) {
15      print_error('incorrectsubwikiid', 'wikicode');
16  }
17
18  if (!$wiki = wikicode_get_wiki($subwiki->wikiid)) {
19      print_error('incorrectwikiid', 'wikicode');
20  }
21
22  if (!$cm = get_coursemodule_from_instance('wikicode', $wiki->id)) {
23      print_error('invalidcoursemodule');
24  }
25
26  $course = $DB->get_record('course', array('id' => $cm->course), '*', MUST_EXIST);
27
28  require_login($course->id, true, $cm);
29
30  add_to_log($course->id, 'wikicode', 'comments', 'comments.php?id=' . $cm->id, $wiki
        ->id);
31
32  /// Print the page header
33  $wikipage = new page_wikicode_comments($wiki, $subwiki, $cm);
34
35  $wikipage->set_page($page);
36
37  $wikipage->print_header();
38  $wikipage->print_content();
```

```
39 $wikipage->print_footer();
```

3.1.3. comments_form.php

```
1  <?php
2
3  if (!defined('MOODLE_INTERNAL')) {
4      die('Direct access to this script is forbidden.');    ///  It must be included
           from a Moodle page
5  }
6
7  require_once($CFG->dirroot . '/lib/formslib.php');
8
9  class mod_wikicode_comments_form extends moodleform {
10     function definition() {
11         $pageid = optional_param('pageid', 0, PARAM_INT);
12         $mform =& $this->_form;
13
14         $current = $this->_customdata['current'];
15         $commentoptions = $this->_customdata['commentoptions'];
16
17         // visible elements
18         $mform->addElement('editor', 'entrycomment_editor', get_string('comment', '
               glossary'), null, $commentoptions);
19         $mform->addRule('entrycomment_editor', get_string('required'), 'required',
               null, 'client');
20         $mform->setType('entrycomment_editor', PARAM_RAW); // processed by trust
               text or cleaned before the display
21
22         // hidden optional params
23         $mform->addElement('hidden', 'id', '');
24         $mform->setType('id', PARAM_INT);
25
26         $mform->addElement('hidden', 'action', '');
27         $mform->setType('action', PARAM_ACTION);
28
29         //
               ---------------------------------------------------------------------------

30         // buttons
31         $this->add_action_buttons(false);
32
33         //
               ---------------------------------------------------------------------------

34         $this->set_data($current);
35     }
36
37     public function edit_definition($current, $commentoptions) {
38         $this->set_data($current);
39         $this->set_data($commentoptions);
```

```
40      }
41  }
```

3.1.4. create.php

```
1   <?php
2
3   require_once('../../config.php');
4   require_once(dirname(__FILE__) . '/create_form.php');
5   require_once($CFG->dirroot . '/mod/wikicode/lib.php');
6   require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
7   require_once($CFG->dirroot . '/mod/wikicode/pagelib.php');
8
9   // this page accepts two actions: new and create
10  // 'new' action will display a form contains page title and page format
11  // selections
12  // 'create' action will create a new page in db, and redirect to
13  // page editing page.
14  $action = optional_param('action', 'new', PARAM_TEXT);
15  // The title of the new page, can be empty
16  $title = optional_param('title', '', PARAM_TEXT);
17  $wid = optional_param('wid', 0, PARAM_INT);
18  $swid = optional_param('swid', 0, PARAM_INT);
19  $gid = optional_param('gid', 0, PARAM_INT);
20  $uid = optional_param('uid', 0, PARAM_INT);
21
22  // 'create' action must be submitted by moodle form
23  // so sesskey must be checked
24  if ($action == 'create') {
25      if (!confirm_sesskey()) {
26          print_error('invalidsesskey');
27      }
28  }
29
30  if (!empty($swid)) {
31      $subwiki = wikicode_get_subwiki($swid);
32
33      if (!$wiki = wikicode_get_wiki($subwiki->wikiid)) {
34          print_error('invalidwikiid', 'wikicode');
35      }
36
37  } else {
38      $subwiki = wikicode_get_subwiki_by_group($wid, $gid, $uid);
39
40      if (!$wiki = wikicode_get_wiki($wid)) {
41          print_error('invalidwikiid', 'wikicode');
42      }
43
44  }
45
46  if (!$cm = get_coursemodule_from_instance('wikicode', $wiki->id)) {
```

```
47      print_error('invalidcoursemoduleid', 'wikicode');
48  }
49
50  $course = $DB->get_record('course', array('id' => $cm->course), '*', MUST_EXIST);
51
52  require_login($course->id, true, $cm);
53
54  add_to_log($course->id, 'createpage', 'createpage', 'view.php?id=' . $cm->id, $wiki
        ->id);
55
56  $wikipage = new page_wikicode_create($wiki, $subwiki, $cm);
57
58  if (!empty($swid)) {
59      $wikipage->set_gid($subwiki->groupid);
60      $wikipage->set_uid($subwiki->userid);
61      $wikipage->set_swid($swid);
62  } else {
63      $wikipage->set_wid($wid);
64      $wikipage->set_gid($gid);
65      $wikipage->set_uid($uid);
66  }
67
68  if (!empty($title)) {
69      $wikipage->set_title($title);
70  } else {
71      $wikipage->set_title(get_string('newpage', 'wikicode'));
72  }
73
74  // set page action, and initialise moodle form
75  $wikipage->set_action($action);
76
77  switch ($action) {
78  case 'create':
79      $wikipage->create_page($title);
80      break;
81  case 'new':
82      if ((int)$wiki->forceformat == 1 && !empty($title)) {
83          $wikipage->create_page($title);
84      } else {
85          // create link from moodle navigation block without pagetitle
86          $wikipage->print_header();
87          // new page without page title
88          $wikipage->print_content($title);
89      }
90      $wikipage->print_footer();
91      break;
92  }
```

3.1.5. create_form.php

```
1   <?php
```

```
2
3   require_once($CFG->libdir.'/formslib.php');
4
5   class mod_wikicode_create_form extends moodleform {
6
7       protected function definition() {
8           global $CFG;
9           $mform =& $this->_form;
10
11          $formats = $this->_customdata['formats'];
12          $defaultformat = $this->_customdata['defaultformat'];
13          $forceformat = $this->_customdata['forceformat'];
14
15          $mform->addElement('header', 'general', get_string('createpage', 'wikicode'
                ));
16
17          $textoptions = array();
18          if (!empty($this->_customdata['disable_pagetitle'])) {
19              $textoptions = array('readonly'=>'readonly');
20          }
21          $mform->addElement('text', 'pagetitle', get_string('newpagetitle', '
                wikicode'), $textoptions);
22
23          if ($forceformat) {
24              $mform->addElement('hidden', 'pageformat', $defaultformat);
25          } else {
26              $mform->addElement('static', 'format', get_string('format', 'wikicode')
                    );
27              $mform->addHelpButton('format', 'format', 'wikicode');
28              foreach ($formats as $format) {
29                  if ($format == $defaultformat) {
30                      $attr = array('checked'=>'checked');
31                  }else if (!empty($forceformat)){
32                      $attr = array('disabled'=>'disabled');
33                  } else {
34                      $attr = array();
35                  }
36                  $mform->addElement('radio', 'pageformat', '', get_string('format'.
                        $format, 'wikicode'), $format, $attr);
37              }
38          }
39
40          //hiddens
41          $mform->addElement('hidden', 'action');
42          $mform->setDefault('action', 'create');
43
44          $this->add_action_buttons(false, get_string('createpage', 'wikicode'));
45      }
46  }
```

3.1.6. diff.php

```
<?php

require_once('../../config.php');

require_once($CFG->dirroot . '/mod/wikicode/lib.php');
require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
require_once($CFG->dirroot . '/mod/wikicode/pagelib.php');

require_once($CFG->dirroot . '/mod/wikicode/diff/difflib.php');
require_once($CFG->dirroot . '/mod/wikicode/diff/diff_nwiki.php');

$pageid = required_param('pageid', PARAM_TEXT);
$compare = required_param('compare', PARAM_INT);
$comparewith = required_param('comparewith', PARAM_INT);

if (!$page = wikicode_get_page($pageid)) {
    print_error('incorrectpageid', 'wikicode');
}

if (!$subwiki = wikicode_get_subwiki($page->subwikiid)) {
    print_error('incorrectsubwikiid', 'wikicode');
}

if (!$wiki = wikicode_get_wiki($subwiki->wikiid)) {
    print_error('incorrectwikiid', 'wikicode');
}

if (!$cm = get_coursemodule_from_instance('wikicode', $wiki->id)) {
    print_error('invalidcoursemodule');
}

$course = $DB->get_record('course', array('id' => $cm->course), '*', MUST_EXIST);

if ($compare >= $comparewith) {
    print_error("A page version can only be compared with an older version.");
}

require_login($course->id, true, $cm);
add_to_log($course->id, "wikicode", "diff", "diff.php?id=$cm->id", "$wiki->id");

$wikipage = new page_wikicode_diff($wiki, $subwiki, $cm);

$wikipage->set_page($page);
$wikipage->set_comparison($compare, $comparewith);

$wikipage->print_header();

$wikipage->print_content();

$wikipage->print_footer();
```

3.1.7. edit.php

```
<?php

require_once('../../config.php');

require_once($CFG->dirroot . '/mod/wikicode/lib.php');
require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
require_once($CFG->dirroot . '/mod/wikicode/pagelib.php');

$pageid = required_param('pageid', PARAM_INT);
$contentformat = optional_param('contentformat', '', PARAM_ALPHA);
$option = optional_param('editoption', '', PARAM_TEXT);
$section = optional_param('section', "", PARAM_TEXT);
$version = optional_param('version', -1, PARAM_INT);
$attachments = optional_param('attachments', 0, PARAM_INT);
$deleteuploads = optional_param('deleteuploads', 0, PARAM_RAW);
$compiled = optional_param('compiled', 0, PARAM_INT);

$newcontent = '';
if (!empty($newcontent) && is_array($newcontent)) {
    $newcontent = $newcontent['text'];
}

if (!empty($option) && is_array($option)) {
    $option = $option['editoption'];
}

if (!$page = wikicode_get_page($pageid)) {
    print_error('incorrectpageid', 'wikicode');
}

if (!$subwiki = wikicode_get_subwiki($page->subwikiid)) {
    print_error('incorrectsubwikiid', 'wikicode');
}

if (!$wiki = wikicode_get_wiki($subwiki->wikiid)) {
    print_error('incorrectwikiid', 'wikicode');
}

if (!$cm = get_coursemodule_from_instance('wikicode', $wiki->id)) {
    print_error('invalidcoursemodule');
}

$course = $DB->get_record('course', array('id' => $cm->course), '*', MUST_EXIST);

if (!empty($section) && !$sectioncontent = wikicode_get_section_page($page,
    $section)) {
    print_error('invalidsection', 'wikicode');
}

require_login($course, true, $cm);

```

```
51  $context = get_context_instance(CONTEXT_MODULE, $cm->id);
52  require_capability('mod/wikicode:editpage', $context);
53
54  add_to_log($course->id, 'wikicode', 'edit', "edit.php?id=$cm->id", "$wiki->id");
55
56  if ($option == get_string('save', 'wikicode')) {
57      if (!confirm_sesskey()) {
58          print_error(get_string('invalidsesskey', 'wikicode'));
59      }
60      $wikipage = new page_wikicode_save($wiki, $subwiki, $cm);
61      $wikipage->set_page($page);
62      $wikipage->set_newcontent($newcontent);
63      $wikipage->set_upload(true);
64  } else {
65      if ($option == 'Compile' or $option == 'Download') {
66          if (!confirm_sesskey()) {
67              print_error(get_string('invalidsesskey', 'wikicode'));
68          }
69          $wikipage = new page_wikicode_compile($wiki, $subwiki, $cm);
70          $wikipage->set_page($page);
71                  $wikipage->set_download(($option == 'Download'));
72      }
73          else {
74          if ($option == get_string('cancel')) {
75              //delete lock
76              wikicode_delete_locks($page->id, $USER->id, $section);
77
78              redirect($CFG->wwwroot . '/mod/wikicode/view.php?pageid=' . $pageid);
79          } else {
80              $wikipage = new page_wikicode_edit($wiki, $subwiki, $cm);
81              $wikipage->set_page($page);
82              $wikipage->set_upload($option == get_string('upload', 'wikicode'));
83                          $wikipage->set_compiled($compiled);
84          }
85      }
86
87      if (has_capability('mod/wikicode:overridelock', $context)) {
88          $wikipage->set_overridelock(true);
89      }
90  }
91
92  if ($version >= 0) {
93      $wikipage->set_versionnumber($version);
94  }
95
96  if (!empty($section)) {
97      $wikipage->set_section($sectioncontent, $section);
98  }
99
100 if (!empty($attachments)) {
101     $wikipage->set_attachments($attachments);
102 }
103
104 if (!empty($deleteuploads)) {
```

```
105     $wikipage->set_deleteuploads($deleteuploads);
106 }
107
108 if (!empty($contentformat)) {
109     $wikipage->set_format($contentformat);
110 }
111
112 $wikipage->print_header();
113
114 $wikipage->print_content();
115
116 $wikipage->print_footer();
```

3.1.8. edit_form.php

```
1  <?php
2
3  if (!defined('MOODLE_INTERNAL')) {
4      die('Direct access to this script is forbidden.');    ///  It must be included
           from a Moodle page
5  }
6
7  require_once($CFG->dirroot . '/mod/wikicode/editors/wikieditor.php');
8  require_once($CFG->dirroot . '/mod/wikicode/chat/wikichat.php');
9
10 class mod_wikicode_edit_form extends moodleform {
11
12     protected function definition() {
13         global $CFG, $USER, $DB;
14
15         $mform =& $this->_form;
16
17         $version = $this->_customdata['version'];
18         $format  = $this->_customdata['format'];
19         $tags    = !isset($this->_customdata['tags'])?"":$this->_customdata['tags'
               ];
20
21         if ($format != 'html') {
22             $contextid  = $this->_customdata['contextid'];
23             $filearea   = $this->_customdata['filearea'];
24             $fileitemid = $this->_customdata['fileitemid'];
25         }
26
27         if (isset($this->_customdata['pagetitle'])) {
28             $pagetitle = get_string('editingpage', 'wikicode', $this->_customdata['
                   pagetitle']);
29         } else {
30             $pagetitle = get_string('editing', 'wikicode');
31         }
32
33         //editor
```

```
34          $mform->addElement('header', 'general', $pagetitle);
35
36          $fieldname = get_string('format' . $format, 'wikicode');
37          if ($format != 'html') {
38              // Use wiki editor
39              $extensions = file_get_typegroup('extension', 'web_image');
40              $fs = get_file_storage();
41              $tree = $fs->get_area_tree($contextid, 'mod_wikicode', 'attachments',
                    $fileitemid);
42              $files = array();
43              foreach ($tree['files'] as $file) {
44                  $filename = $file->get_filename();
45                  foreach ($extensions as $ext) {
46                      if (preg_match('#'.$ext.'$#i', $filename)) {
47                          $files[] = $filename;
48                      }
49                  }
50              }
51                          $buttoncommands=array();
52                          $buttoncommands[] =& $mform->createElement('button','
                                editoption','Unlock', array('id' => 'btnunlock', 'class
                                ' => 'btnunlock'));
53                          $buttoncommands[] =& $mform->createElement('button','
                                editoption','Refresh', array('id' => 'btnref', 'class'
                                => 'btnref'));
54                          $buttoncommands[] =& $mform->createElement('submit', '
                                editoption', 'Save', array('id' => 'save'));
55                          $mform->addGroup($buttoncommands, 'editoption', 'Actions:',
                                 '', true);
56              $mform->addElement('wikicodeeditor', 'newcontent', $fieldname, array('
                    cols' => 150, 'rows' => 30, 'Wiki_format' => $format, 'files'=>
                    $files));
57          } else {
58              $mform->addElement('editor', 'newcontent_editor', $fieldname, null,
                    page_wikicode_edit::$attachmentoptions);
59              $mform->addHelpButton('newcontent_editor', 'formathtml', 'wikicode');
60          }
61
62                  //chat
63                  $mform->addElement('header','chat','Chat');
64                  $mform->addElement('wikicodechat', 'wikicodechat', null, array('
                        itemid'=>$fileitemid));
65
66                  //compiler
67                  $mform->addElement('header','compiler', 'Compiler');
68                  $mform->addElement('textarea', 'textCompiler', '', 'wrap="virtual"
                        rows="3" cols="150" readonly="readonly" ');
69                  //$mform->addElement('submit','editoption','Compilar', array('id'
                        => 'compile'));
70                  //$mform->addElement('submit','editoption','Descargar', array('id'
                        => 'download'));
71
72                  $buttonarray=array();
```

```
73                  $buttonarray[] =& $mform->createElement('submit','editoption','
                        Compile', array('id' => 'compile'));
74                  $buttonarray[] =& $mform->createElement('submit','editoption','
                        Download', array('id' => 'compile'));
75                  $mform->addGroup($buttonarray, 'editoption', 'Options:', '', true);
76
77          //hiddens
78          if ($version >= 0) {
79              $mform->addElement('hidden', 'version');
80              $mform->setDefault('version', $version);
81          }
82
83          $mform->addElement('hidden', 'contentformat');
84          $mform->setDefault('contentformat', $format);
85
86                  $mform->addElement('hidden', 'insert');
87                  $mform->setDefault('insert', 1);
88
89      }
90
91  }
```

3.1.9. files.php

```
1   <?php
2   require_once('../../config.php');
3   require_once($CFG->dirroot . '/mod/wikicode/lib.php');
4   require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
5
6   $pageid       = required_param('pageid', PARAM_INT); // Page ID
7   $wid          = optional_param('wid', 0, PARAM_INT); // Wiki ID
8   $currentgroup = optional_param('group', 0, PARAM_INT); // Group ID
9   $userid       = optional_param('uid', 0, PARAM_INT); // User ID
10  $groupanduser = optional_param('groupanduser', null, PARAM_TEXT);
11
12  if (!$page = wikicode_get_page($pageid)) {
13      print_error('incorrectpageid', 'wikicode');
14  }
15
16  if ($groupanduser) {
17      list($currentgroup, $userid) = explode('-', $groupanduser);
18      $currentgroup = clean_param($currentgroup, PARAM_INT);
19      $userid       = clean_param($userid, PARAM_INT);
20  }
21
22  if ($wid) {
23      // in group mode
24      if (!$wiki = wikicode_get_wiki($wid)) {
25          print_error('incorrectwikiid', 'wikicode');
26      }
```

```
27      if (!$subwiki = wikicode_get_subwiki_by_group($wiki->id, $currentgroup, $userid
            )) {
28          // create subwiki if doesn't exist
29          $subwikiid = wikicode_add_subwiki($wiki->id, $currentgroup, $userid);
30          $subwiki = wikicode_get_subwiki($subwikiid);
31      }
32  } else {
33      // no group
34      if (!$subwiki = wikicode_get_subwiki($page->subwikiid)) {
35          print_error('incorrectsubwikiid', 'wikicode');
36      }
37
38      // Checking wiki instance of that subwiki
39      if (!$wiki = wikicode_get_wiki($subwiki->wikiid)) {
40          print_error('incorrectwikiid', 'wikicode');
41      }
42  }
43
44  // Checking course module instance
45  if (!$cm = get_coursemodule_from_instance("wikicode", $subwiki->wikiid)) {
46      print_error('invalidcoursemodule');
47  }
48
49  // Checking course instance
50  $course = $DB->get_record('course', array('id' => $cm->course), '*', MUST_EXIST);
51
52  $context = get_context_instance(CONTEXT_MODULE, $cm->id);
53
54
55  $PAGE->set_url('/mod/wikicode/files.php', array('pageid'=>$pageid));
56  require_login($course, true, $cm);
57  $PAGE->set_context($context);
58  $PAGE->set_title(get_string('wikifiles', 'wikicode'));
59  $PAGE->set_heading(get_string('wikifiles', 'wikicode'));
60  $PAGE->navbar->add(format_string(get_string('wikifiles', 'wikicode')));
61  echo $OUTPUT->header();
62
63  $renderer = $PAGE->get_renderer('mod_wikicode');
64
65  $tabitems = array('view' => 'view', 'edit' => 'edit', 'comments' => 'comments', '
        history' => 'history', 'map' => 'map', 'files' => 'files');
66
67  $options = array('activetab'=>'files');
68  echo $renderer->tabs($page, $tabitems, $options);
69
70
71  echo $OUTPUT->box_start('generalbox');
72  if (has_capability('mod/wikicode:viewpage', $context)) {
73      echo $renderer->wikicode_print_subwiki_selector($PAGE->activityrecord, $subwiki
            , $page, 'files');
74      echo $renderer->wikicode_files_tree($context, $subwiki);
75  } else {
76      echo $OUTPUT->notification(get_string('cannotviewfiles', 'wikicode'));
77  }
```

```
78  echo $OUTPUT->box_end();
79
80  if (has_capability('mod/wikicode:managefiles', $context)) {
81      echo $OUTPUT->single_button(new moodle_url('/mod/wikicode/filesedit.php', array
            ('subwiki'=>$subwiki->id, 'pageid'=>$pageid)), get_string('editfiles', '
            wikicode'), 'get');
82  }
83  echo $OUTPUT->footer();
```

3.1.10. filesedit.php

```
1   <?php
2
3   /**
4    * Manage files in wiki
5    *
6    * @package   mod-wikicode
7    * @copyright 2011 Dongsheng Cai <dongsheng@moodle.com>
8    * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
9    */
10
11  require_once(dirname(dirname(dirname(__FILE__))) . '/config.php');
12  require_once('lib.php');
13  require_once('locallib.php');
14  require_once("$CFG->dirroot/mod/wikicode/filesedit_form.php");
15  require_once("$CFG->dirroot/repository/lib.php");
16
17  $subwikiid = required_param('subwiki', PARAM_INT);
18  // not being used for file management, we use it to generate navbar link
19  $pageid    = optional_param('pageid', 0, PARAM_INT);
20  $returnurl = optional_param('returnurl', '', PARAM_URL);
21
22  if (!$subwiki = wikicode_get_subwiki($subwikiid)) {
23      print_error('incorrectsubwikiid', 'wikicode');
24  }
25
26  // Checking wiki instance of that subwiki
27  if (!$wiki = wikicode_get_wiki($subwiki->wikiid)) {
28      print_error('incorrectwikiid', 'wikicode');
29  }
30
31  // Checking course module instance
32  if (!$cm = get_coursemodule_from_instance("wikicode", $subwiki->wikiid)) {
33      print_error('invalidcoursemodule');
34  }
35
36  // Checking course instance
37  $course = $DB->get_record('course', array('id' => $cm->course), '*', MUST_EXIST);
38
39  $context = get_context_instance(CONTEXT_MODULE, $cm->id);
40
```

```
41  require_login($course, true, $cm);
42  require_capability('mod/wikicode:managefiles', $context);
43
44  if (empty($returnurl)) {
45      if (!empty($_SERVER["HTTP_REFERER"])) {
46          $returnurl = $_SERVER["HTTP_REFERER"];
47      } else {
48          $returnurl = new moodle_url('/mod/wikicode/files.php', array('subwiki'=>
                $subwiki->id));
49      }
50  }
51
52  $title = get_string('editfiles', 'wikicode');
53
54  $struser = get_string('user');
55  $url = new moodle_url('/mod/wikicode/filesedit.php', array('subwiki'=>$subwiki->id,
        'pageid'=>$pageid));
56  $PAGE->set_url($url);
57  $PAGE->set_context($context);
58  $PAGE->set_title($title);
59  $PAGE->set_heading($title);
60  $PAGE->navbar->add(format_string(get_string('wikifiles', 'wikicode')), $CFG->
        wwwroot . '/mod/wikicode/files.php?pageid=' . $pageid);
61  $PAGE->navbar->add(format_string($title));
62
63  $data = new stdClass();
64  $data->returnurl = $returnurl;
65  $data->subwikiid = $subwiki->id;
66  $maxbytes = get_max_upload_file_size($CFG->maxbytes, $COURSE->maxbytes);
67  $options = array('subdirs'=>0, 'maxbytes'=>$maxbytes, 'maxfiles'=>-1, '
        accepted_types'=>'*', 'return_types'=>FILE_INTERNAL);
68  file_prepare_standard_filemanager($data, 'files', $options, $context, 'mod_wiki', '
        attachments', $subwiki->id);
69
70  $mform = new mod_wikicode_filesedit_form(null, array('data'=>$data, 'options'=>
        $options));
71
72  if ($mform->is_cancelled()) {
73      redirect($returnurl);
74  } else if ($formdata = $mform->get_data()) {
75      $formdata = file_postupdate_standard_filemanager($formdata, 'files', $options,
            $context, 'mod_wikicode', 'attachments', $subwiki->id);
76      redirect($returnurl);
77  }
78
79  echo $OUTPUT->header();
80  echo $OUTPUT->box_start('generalbox');
81  $mform->display();
82  echo $OUTPUT->box_end();
83  echo $OUTPUT->footer();
```

3.1.11. filesedit_form.php

```php
<?php

/**
 * Edit wiki files form
 *
 * @package   mod-wikicode
 * @copyright 2011 Dongsheng Cai <dongsheng@moodle.com>
 * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
 */

defined('MOODLE_INTERNAL') || die();

require_once("$CFG->libdir/formslib.php");

class mod_wikicode_filesedit_form extends moodleform {
    function definition() {
        $mform = $this->_form;

        $data    = $this->_customdata['data'];
        $options = $this->_customdata['options'];

        $mform->addElement('filemanager', 'files_filemanager', get_string('files'),
             null, $options);
        $mform->addElement('hidden', 'returnurl', $data->returnurl);
        $mform->addElement('hidden', 'subwiki', $data->subwikiid);

        $this->add_action_buttons(true, get_string('savechanges'));

        $this->set_data($data);
    }
}
```

3.1.12. history.php

```php
<?php

require_once('../../config.php');

require_once($CFG->dirroot . '/mod/wikicode/lib.php');
require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
require_once($CFG->dirroot . '/mod/wikicode/pagelib.php');

$pageid = required_param('pageid', PARAM_TEXT);
$paging = optional_param('page', 0, PARAM_INT);
$allversion = optional_param('allversion', 0, PARAM_INT);

if (!$page = wikicode_get_page($pageid)) {
    print_error('incorrectpageid', 'wikicode');
```

```
15  }
16
17  if (!$subwiki = wikicode_get_subwiki($page->subwikiid)) {
18      print_error('incorrectsubwikiid', 'wikicode');
19  }
20
21  if (!$wiki = wikicode_get_wiki($subwiki->wikiid)) {
22      print_error('incorrectwikiid', 'wikicode');
23  }
24
25  if (!$cm = get_coursemodule_from_instance('wikicode', $wiki->id)) {
26      print_error('invalidcoursemodule');
27  }
28
29  $course = $DB->get_record('course', array('id' => $cm->course), '*', MUST_EXIST);
30
31  require_login($course->id, true, $cm);
32  $context = get_context_instance(CONTEXT_MODULE, $cm->id);
33  require_capability('mod/wikicode:viewpage', $context);
34  add_to_log($course->id, 'wikicode', 'history', 'history.php?id=' . $cm->id, $wiki->
        id);
35
36  /// Print the page header
37  $wikipage = new page_wikicode_history($wiki, $subwiki, $cm);
38
39  $wikipage->set_page($page);
40  $wikipage->set_paging($paging);
41  $wikipage->set_allversion($allversion);
42
43  $wikipage->print_header();
44  $wikipage->print_content();
45
46  $wikipage->print_footer();
```

3.1.13. index.php

```
1   <?php
2
3   require_once('../../config.php');
4   require_once('lib.php');
5
6   $id = required_param('id', PARAM_INT); // course
7   $PAGE->set_url('/mod/wikicode/index.php', array('id' => $id));
8
9   if (!$course = $DB->get_record('course', array('id' => $id))) {
10      print_error('invalidcourseid');
11  }
12
13  require_login($course->id, true);
14  $PAGE->set_pagelayout('incourse');
15  $context = get_context_instance(CONTEXT_COURSE, $course->id);
```

```
16
17  add_to_log($course->id, 'wikicode', 'view all', "index.php?id=$course->id", "");
18
19  /// Get all required stringswiki
20  $strwikis = get_string("modulenameplural", "wikicode");
21  $strwiki = get_string("modulename", "wikicode");
22
23  /// Print the header
24  $PAGE->navbar->add($strwikis, "index.php?id=$course->id");
25  $PAGE->set_title($strwikis);
26  $PAGE->set_heading($course->fullname);
27  echo $OUTPUT->header();
28
29  /// Get all the appropriate data
30  if (!$wikis = get_all_instances_in_course("wikicode", $course)) {
31      notice("There are no wikis", "../../course/view.php?id=$course->id");
32      die;
33  }
34
35  $usesections = course_format_uses_sections($course->format);
36  if ($usesections) {
37      $sections = get_all_sections($course->id);
38  }
39
40  /// Print the list of instances (your module will probably extend this)
41
42  $timenow = time();
43  $strsectionname = get_string('sectionname', 'format_' . $course->format);
44  $strname = get_string("name");
45  $table = new html_table();
46
47  if ($usesections) {
48      $table->head = array($strsectionname, $strname);
49  } else {
50      $table->head = array($strname);
51  }
52
53  foreach ($wikis as $wiki) {
54      $linkcss = null;
55      if (!$wiki->visible) {
56          $linkcss = array('class' => 'dimmed');
57      }
58      $link = html_writer::link(new moodle_url('/mod/wikicode/view.php', array('id'
            => $wiki->coursemodule)), $wiki->name, $linkcss);
59
60      if ($usesections) {
61          $table->data[] = array(get_section_name($course, $sections[$wiki->section])
                , $link);
62      } else {
63          $table->data[] = array($link);
64      }
65  }
66
67  echo html_writer::table($table);
```

```
68
69  /// Finish the page
70  echo $OUTPUT->footer();
```

3.1.14. log.php

```
1   <?php
2
3   require_once('../../config.php');
4
5   require_once($CFG->dirroot . '/mod/wikicode/lib.php');
6   require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
7   require_once($CFG->dirroot . '/mod/wikicode/pagelib.php');
8
9   $pageid = required_param('pageid', PARAM_INT);
10  $section = optional_param('section', "", PARAM_TEXT);
11  $version = optional_param('version', -1, PARAM_INT);
12
13  if (!$page = wikicode_get_page($pageid)) {
14      print_error('incorrectpageid', 'wikicode');
15  }
16
17  if (!$subwiki = wikicode_get_subwiki($page->subwikiid)) {
18      print_error('incorrectsubwikiid', 'wikicode');
19  }
20
21  if (!$wiki = wikicode_get_wiki($subwiki->wikiid)) {
22      print_error('incorrectwikiid', 'wikicode');
23  }
24
25  if (!$cm = get_coursemodule_from_instance('wikicode', $wiki->id)) {
26      print_error('invalidcoursemodule');
27  }
28
29  $course = $DB->get_record('course', array('id' => $cm->course), '*', MUST_EXIST);
30
31  if (!empty($section) && !$sectioncontent = wikicode_get_section_page($page,
        $section)) {
32      print_error('invalidsection', 'wikicode');
33  }
34
35  require_login($course, true, $cm);
36
37  $context = get_context_instance(CONTEXT_MODULE, $cm->id);
38  require_capability('mod/wikicode:editpage', $context);
39
40  add_to_log($course->id, 'wikicode', 'log', "log.php?id=$cm->id", "$wiki->id");
41
42  $wikipage = new page_wikicode_log($wiki, $subwiki, $cm);
43  $wikipage->set_page($page);
44
```

```
45 $wikipage->print_header();
46
47 $wikipage->print_content();
48
49 $wikipage->print_footer();
```

3.1.15. log_form.php

```
1  <?php
2
3  if (!defined('MOODLE_INTERNAL')) {
4      die('Direct access to this script is forbidden.');    ///  It must be included
           from a Moodle page
5  }
6
7  class mod_wikicode_log_form extends moodleform {
8
9      protected function definition() {
10         global $CFG, $USER, $DB;
11
12         $mform =& $this->_form;
13
14         $version = $this->_customdata['version'];
15         $format  = $this->_customdata['format'];
16         $tags    = !isset($this->_customdata['tags'])?"":$this->_customdata['tags'
               ];
17
18         if ($format != 'html') {
19             $contextid  = $this->_customdata['contextid'];
20             $filearea   = $this->_customdata['filearea'];
21             $fileitemid = $this->_customdata['fileitemid'];
22         }
23
24         if (isset($this->_customdata['pagetitle'])) {
25             $pagetitle = get_string('logpage', 'wikicode', $this->_customdata['
                   pagetitle']);
26         } else {
27             $pagetitle = get_string('loging', 'wikicode');
28         }
29
30                 //Time
31                 $time = $this->_customdata['page']->timeendedit - $this->
                       _customdata['page']->timestartedit;
32                 $seconds = $time % 60;
33                 $time = ($time - $seconds) / 60;
34                 $minutes = $time % 60;
35                 $hours = ($time - $minutes) / 60;
36
37                 //Stats
38                 $attr = array('size' => '75', 'readonly' => 1);
39                 $mform->addElement('header','stats', 'Stats');
```

```
40                  $attr['value'] = $hours . " hours, " . $minutes . " minutes, " .
                        $seconds . " seconds";
41                  $mform->addElement('text', 'timeedit', 'Editing Time', $attr);
42                  $mform->addHelpButton('timeedit', 'timeedit', 'wikicode');
43                  $attr['value'] = $this->_customdata['page']->errorcompile;
44                  $mform->addElement('text', 'errorscompilation', 'Compilation Errors
                        ', $attr);
45                  $mform->addHelpButton('errorscompilation', 'errorscompilation', '
                        wikicode');
46
47
48          $mform->addElement('hidden', 'contentformat');
49          $mform->setDefault('contentformat', $format);
50
51                  $mform->addElement('hidden', 'insert');
52                  $mform->setDefault('insert', 1);
53
54      }
55
56  }
```

3.1.16. mod_form.php

```
1   <?php
2
3   if (!defined('MOODLE_INTERNAL')) {
4       die('Direct access to this script is forbidden.');    ///  It must be included
            from a Moodle page
5   }
6
7   require_once('moodleform_mod.php');
8   require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
9   require_once($CFG->dirroot . '/lib/datalib.php');
10
11  class mod_wikicode_mod_form extends moodleform_mod {
12
13      function definition() {
14
15          global $COURSE;
16          $mform =& $this->_form;
17
18          //
                ------------------------------------------------------------------------------

19          /// Adding the "general" fieldset, where all the common settings are showed
20          $mform->addElement('header', 'general', get_string('general', 'form'));
21          /// Adding the standard "name" field
22          $mform->addElement('text', 'name', get_string('wikiname', 'wikicode'),
                array('size' => '64'));
23          $mform->setType('name', PARAM_TEXT);
24          $mform->addRule('name', null, 'required', null, 'client');
```

```
25          /// Adding the optional "intro" and "introformat" pair of fields
26          //      $mform->addElement('htmleditor', 'intro', get_string('wikiintro', '
                wiki'));
27          //              $mform->setType('intro', PARAM_RAW);
28          //              $mform->addRule('intro', get_string('required'), 'required
                ', null, 'client');
29          //
30          //        $mform->addElement('format', 'introformat', get_string('format'))
                ;
31          $this->add_intro_editor(true, get_string('wikiintro', 'wikicode'));
32          //
                ------------------------------------------------------------------------

33          /// Adding the rest of wiki settings, spreeading all them into this
                fieldset
34          /// or adding more fieldsets ('header' elements) if needed for better logic
35
36          $mform->addElement('header', 'wikifieldset', get_string('wikisettings', '
                wikicode'));
37
38          $attr = array('size' => '20');
39          if (!empty($this->_instance)) {
40              $attr['disabled'] = 'disabled';
41          } else {
42              $attr['value'] = get_string('firstpagetitle', 'wikicode');
43          }
44
45          $mform->addElement('text', 'firstpagetitle', get_string('firstpagetitle', '
                wikicode'), $attr);
46          $mform->addHelpButton('firstpagetitle', 'firstpagetitle', 'wikicode');
47
48          if (empty($this->_instance)) {
49              $mform->addRule('firstpagetitle', null, 'required', null, 'client');
50          }
51
52                  $attr = array('size' => '180');
53
54                  $gccvalue = wikicode_get_gccpath();
55                  if ($gccvalue->gccpath != "") {
56                      $attr['value'] = $gccvalue->gccpath;
57                  } else {
58                          $attr['value'] = 'gcc';
59                  }
60
61                  $mform->addElement('text', 'gccpath', 'Unix Compiler Path', $attr);
62                  $mform->addHelpButton('gccpath', 'gccpath', 'wikicode');
63
64                  $mingwvalue = wikicode_get_mingwpath();
65                  if ($mingwvalue->mingwpath != "") {
66                      $attr['value'] = $mingwvalue->mingwpath;
67                  } else {
68                          $attr['value'] = 'mingw32-gcc';
69                  }
70
```

```
71                  $mform->addElement('text', 'mingwpath', 'Windows Compiler Path',
                        $attr);
72                  $mform->addHelpButton('mingwpath', 'mingwpath', 'wikicode');
73
74          $wikimodeoptions = array ('collaborative' => get_string('
                wikimodecollaborative', 'wikicode'), 'individual' => get_string('
                wikimodeindividual', 'wikicode'));
75          // don't allow to change wiki type once is set
76          $wikitype_attr = array();
77          if (!empty($this->_instance)) {
78              $wikitype_attr['disabled'] = 'disabled';
79          }
80          $mform->addElement('select', 'wikimode', get_string('wikimode', 'wikicode')
                , $wikimodeoptions, $wikitype_attr);
81          $mform->addHelpButton('wikimode', 'wikimode', 'wikicode');
82
83          $formats = wikicode_get_formats();
84          $editoroptions = array();
85          foreach ($formats as $format) {
86              $editoroptions[$format] = get_string($format, 'wikicode');
87          }
88          $mform->addElement('select', 'defaultformat', get_string('defaultformat', '
                wikicode'), $editoroptions);
89          $mform->addHelpButton('defaultformat', 'defaultformat', 'wikicode');
90          $mform->addElement('checkbox', 'forceformat', get_string('forceformat', '
                wikicode'));
91          $mform->addHelpButton('forceformat', 'forceformat', 'wikicode');
92
93          //
                ---------------------------------------------------------------------------

94          // add standard elements, common to all modules
95          $this->standard_coursemodule_elements();
96          //
                ---------------------------------------------------------------------------

97          // add standard buttons, common to all modules
98          $this->add_action_buttons();
99
100     }
101 }
```

3.1.17. pagelib.php

```
1   <?php
2
3   require_once($CFG->dirroot . '/mod/wikicode/edit_form.php');
4   require_once($CFG->dirroot . '/mod/wikicode/log_form.php');
5   require_once($CFG->dirroot . '/tag/lib.php');
6
7   /**
```

```
8    * Class page_wikicode contains the common code between all pages
9    *
10   * @license    http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
11   */
12  abstract class page_wikicode {
13
14      /**
15       * @var object Current subwiki
16       */
17      protected $subwiki;
18
19      /**
20       * @var int Current page
21       */
22      protected $page;
23
24      /**
25       * @var string Current page title
26       */
27      protected $title;
28
29      /**
30       * @var int Current group ID
31       */
32      protected $gid;
33
34      /**
35       * @var object module context object
36       */
37      protected $modcontext;
38
39      /**
40       * @var int Current user ID
41       */
42      protected $uid;
43      /**
44       * @var array The tabs set used in wiki module
45       */
46      protected $tabs = array('view' => 'view', 'edit' => 'edit', 'history' => '
            history', 'log' => 'log', 'admin' => 'admin');
47      /**
48       * @var array tabs options
49       */
50      protected $tabs_options = array();
51      /**
52       * @var object wiki renderer
53       */
54      protected $wikioutput;
55
56      /**
57       * page_wikicode constructor
58       *
59       * @param $wiki. Current wiki
60       * @param $subwiki. Current subwiki.
```

```
61          * @param $cm. Current course_module.
62          */
63         function __construct($wiki, $subwiki, $cm) {
64             global $PAGE, $CFG;
65             $this->subwiki = $subwiki;
66             $this->modcontext = get_context_instance(CONTEXT_MODULE, $PAGE->cm->id);
67 
68             // initialise wiki renderer
69             $this->wikioutput = $PAGE->get_renderer('mod_wikicode');
70             $PAGE->set_cacheable(true);
71             $PAGE->set_cm($cm);
72             $PAGE->set_activity_record($wiki);
73             // the search box
74             $PAGE->set_button(wikicode_search_form($cm));
75         }
76 
77         /**
78          * This method prints the top of the page.
79          */
80         function print_header() {
81             global $OUTPUT, $PAGE, $CFG, $USER, $SESSION;
82 
83             $PAGE->set_heading(format_string($PAGE->course->fullname));
84 
85             $this->set_url();
86 
87             if (isset($SESSION->wikipreviousurl) && is_array($SESSION->wikipreviousurl)
                    ) {
88                 $this->process_session_url();
89             }
90             $this->set_session_url();
91 
92             $this->create_navbar();
93             $this->setup_tabs();
94 
95             echo $OUTPUT->header();
96 
97             echo $this->wikioutput->wikicode_info();
98 
99             // tabs are associated with pageid, so if page is empty, tabs should be
                    disabled
100            if (!empty($this->page) && !empty($this->tabs)) {
101                echo $this->wikioutput->tabs($this->page, $this->tabs, $this->
                        tabs_options);
102            }
103        }
104 
105        /**
106         * Protected method to print current page title.
107         */
108        protected function print_pagetitle() {
109            global $OUTPUT;
110            $html = '';
111 
```

```
112             $html .= $OUTPUT->container_start();
113             $html .= $OUTPUT->heading(format_string($this->title), 2, '
                    wikicode_headingtitle');
114             $html .= $OUTPUT->container_end();
115             echo $html;
116         }
117
118         /**
119          * Setup page tabs, if options is empty, will set up active tab automatically
120          * @param array $options, tabs options
121          */
122         protected function setup_tabs($options = array()) {
123             global $CFG, $PAGE;
124             $groupmode = groups_get_activity_groupmode($PAGE->cm);
125
126             if (empty($CFG->usecomments) || !has_capability('mod/wikicode:viewcomment',
                     $PAGE->context)){
127                 unset($this->tabs['comments']);
128             }
129
130             if (!has_capability('mod/wikicode:editpage', $PAGE->context)){
131                 unset($this->tabs['edit']);
132             }
133
134             if ($groupmode and $groupmode == VISIBLEGROUPS) {
135                 $currentgroup = groups_get_activity_group($PAGE->cm);
136                 $manage = has_capability('mod/wikicode:managewiki', $PAGE->cm->context)
                        ;
137                 $edit = has_capability('mod/wikicode:editpage', $PAGE->context);
138                 if (!$manage and !($edit and groups_is_member($currentgroup))) {
139                     unset($this->tabs['edit']);
140                 }
141             } else {
142                 if (!has_capability('mod/wikicode:editpage', $PAGE->context)) {
143                     unset($this->tabs['edit']);
144                 }
145             }
146
147
148             if (empty($options)) {
149                 $this->tabs_options = array('activetab' => substr(get_class($this), 10)
                        );
150             } else {
151                 $this->tabs_options = $options;
152             }
153
154         }
155
156         /**
157          * This method must be overwritten to print the page content.
158          */
159         function print_content() {
160             throw new coding_exception('Page wiki class does not implement method
                    print_content()');
```

```
161     }
162
163     /**
164      * Method to set the current page
165      *
166      * @param object $page Current page
167      */
168     function set_page($page) {
169         global $PAGE;
170
171         $this->page = $page;
172         $this->title = $page->title;
173         $PAGE->set_title($this->title);
174     }
175
176     /**
177      * Method to set the current page title.
178      * This method must be called when the current page is not created yet.
179      * @param string $title Current page title.
180      */
181     function set_title($title) {
182         global $PAGE;
183
184         $this->page = null;
185         $this->title = $title;
186         $PAGE->set_title($this->title);
187     }
188
189     /**
190      * Method to set current group id
191      * @param int $gid Current group id
192      */
193     function set_gid($gid) {
194         $this->gid = $gid;
195     }
196
197     /**
198      * Method to set current user id
199      * @param int $uid Current user id
200      */
201     function set_uid($uid) {
202         $this->uid = $uid;
203     }
204
205     /**
206      * Method to set the URL of the page.
207      * This method must be overwritten by every type of page.
208      */
209     protected function set_url() {
210         throw new coding_exception('Page wiki class does not implement method
                set_url()');
211     }
212
213     /**
```

```
214         * Protected method to create the common items of the navbar in every page type
                .
215         */
216        protected function create_navbar() {
217            global $PAGE, $CFG;
218
219            $PAGE->navbar->add(format_string($this->title), $CFG->wwwroot . '/mod/
                   wikicode/view.php?pageid=' . $this->page->id);
220        }
221
222        /**
223         * This method print the footer of the page.
224         */
225        function print_footer() {
226            global $OUTPUT;
227            echo $OUTPUT->footer();
228        }
229
230        protected function process_session_url() {
231            global $USER, $SESSION;
232
233            //delete locks if edit
234            $url = $SESSION->wikipreviousurl;
235            switch ($url['page']) {
236            case 'edit':
237                wikicode_delete_locks($url['params']['pageid'], $USER->id, $url['params
                       ']['section'], false);
238                break;
239            }
240        }
241
242        protected function set_session_url() {
243            global $SESSION;
244            unset($SESSION->wikipreviousurl);
245        }
246
247    }
248
249    /**
250     * View a wiki page
251     *
252     * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
253     */
254    class page_wikicode_view extends page_wikicode {
255        /**
256         * @var int the coursemodule id
257         */
258        private $coursemodule;
259
260        function print_header() {
261            global $PAGE;
262
263            parent::print_header();
264
```

```
265             $this->wikioutput->wikicode_print_subwiki_selector($PAGE->activityrecord,
                    $this->subwiki, $this->page, 'view');
266
267             //if (!empty($this->page)) {
268             //    echo $this->wikioutput->prettyview_link($this->page);
269             //}
270
271             //echo $this->wikioutput->page_index();
272
273             $this->print_pagetitle();
274         }
275
276         function print_content() {
277             global $PAGE, $CFG;
278
279             if (wikicode_user_can_view($this->subwiki)) {
280
281                 if (!empty($this->page)) {
282                     wikicode_print_page_content($this->page, $this->modcontext, $this->
                            subwiki->id);
283                     $wiki = $PAGE->activityrecord;
284                 } else {
285                     print_string('nocontent', 'wikicode');
286                     // TODO: fix this part
287                     $swid = 0;
288                     if (!empty($this->subwiki)) {
289                         $swid = $this->subwiki->id;
290                     }
291                 }
292             } else {
293                 // @TODO: Tranlate it
294                 echo "You can not view this page";
295             }
296         }
297
298         function set_url() {
299             global $PAGE, $CFG;
300             $params = array();
301
302             if (isset($this->coursemodule)) {
303                 $params['id'] = $this->coursemodule;
304             } else if (!empty($this->page) and $this->page != null) {
305                 $params['pageid'] = $this->page->id;
306             } else if (!empty($this->gid)) {
307                 $params['wid'] = $PAGE->cm->instance;
308                 $params['group'] = $this->gid;
309             } else if (!empty($this->title)) {
310                 $params['swid'] = $this->subwiki->id;
311                 $params['title'] = $this->title;
312             } else {
313                 print_error(get_string('invalidparameters', 'wikicode'));
314             }
315
316             $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/view.php', $params);
```

```
317         }
318
319         function set_coursemodule($id) {
320             $this->coursemodule = $id;
321         }
322
323         protected function create_navbar() {
324             global $PAGE, $CFG;
325
326             $PAGE->navbar->add(format_string($this->title));
327             $PAGE->navbar->add(get_string('view', 'wikicode'));
328         }
329     }
330
331     /**
332      * Wiki page editing page
333      *
334      * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
335      */
336     class page_wikicode_edit extends page_wikicode {
337
338         public static $attachmentoptions;
339
340         protected $sectioncontent;
341         /** @var string the section name needed to be edited */
342         protected $section;
343         protected $overridelock = false;
344         protected $versionnumber = -1;
345         protected $upload = false;
346         protected $attachments = 0;
347         protected $deleteuploads = array();
348         protected $format;
349             protected $compiled = 0;
350
351         function __construct($wiki, $subwiki, $cm) {
352             global $CFG, $PAGE;
353             parent::__construct($wiki, $subwiki, $cm);
354             self::$attachmentoptions = array('subdirs' => false, 'maxfiles' => -1, '
                    maxbytes' => $CFG->maxbytes, 'accepted_types' => '*');
355             //$PAGE->requires->js_init_call('M.mod_wikicode.renew_lock', null, true);
356             $PAGE->requires->yui2_lib('connection');
357         }
358
359         protected function print_pagetitle() {
360             global $OUTPUT;
361
362             $title = $this->title;
363             if (isset($this->section)) {
364                 $title .= ' : ' . $this->section;
365             }
366             echo $OUTPUT->container_start('wikicode_clear');
367             echo $OUTPUT->heading(format_string($title), 2, 'wikicode_headingtitle');
368                     echo "<script src=\"js/codemirror.js\" type=\"text/javascript\"></
                            script>";
```

```
369             echo $OUTPUT->container_end();
370         }
371
372         function print_header() {
373             global $OUTPUT, $PAGE;
374             //$PAGE->requires->data_for_js('wikicode', array('renew_lock_timeout' =>
                    LOCK_TIMEOUT - 5, 'pageid' => $this->page->id, 'section' => $this->
                    section));
375
376             parent::print_header();
377
378             $this->print_pagetitle();
379
380             print '<noscript>' . $OUTPUT->box(get_string('javascriptdisabledlocks', '
                    wikicode'), 'errorbox') . '</noscript>';
381         }
382
383         function print_content() {
384             global $PAGE;
385
386             if (wikicode_user_can_edit($this->subwiki)) {
387                 $this->print_edit(null, $compile);
388             } else {
389                 // @TODO: Translate it
390                 echo "You can not edit this page";
391             }
392         }
393
394         protected function set_url() {
395             global $PAGE, $CFG;
396
397             $params = array('pageid' => $this->page->id);
398
399             if (isset($this->section)) {
400                 $params['section'] = $this->section;
401             }
402
403             $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/edit.php', $params);
404         }
405
406         protected function set_session_url() {
407             global $SESSION;
408
409             $SESSION->wikipreviousurl = array('page' => 'edit', 'params' => array('
                    pageid' => $this->page->id, 'section' => $this->section));
410         }
411
412         protected function process_session_url() {
413         }
414
415         function set_section($sectioncontent, $section) {
416             $this->sectioncontent = $sectioncontent;
417             $this->section = $section;
418         }
```

```
419
420     public function set_versionnumber($versionnumber) {
421         $this->versionnumber = $versionnumber;
422     }
423
424     public function set_overridelock($override) {
425         $this->overridelock = $override;
426     }
427
428     function set_format($format) {
429         $this->format = $format;
430     }
431
432     public function set_upload($upload) {
433         $this->upload = $upload;
434     }
435
436     public function set_attachments($attachments) {
437         $this->attachments = $attachments;
438     }
439
440     public function set_deleteuploads($deleteuploads) {
441         $this->deleteuploads = $deleteuploads;
442     }
443
444         public function set_compiled($compiled) {
445                 $this->compiled = $compiled;
446         }
447
448     protected function create_navbar() {
449         global $PAGE, $CFG;
450
451         parent::create_navbar();
452
453         $PAGE->navbar->add(get_string('edit', 'wikicode'));
454     }
455
456     protected function check_locks() {
457         global $OUTPUT, $USER, $CFG;
458
459         /*if (!wikicode_set_lock($this->page->id, $USER->id, $this->section, true))
                 {
460             print $OUTPUT->box(get_string('pageislocked', 'wikicode'), 'generalbox
                 boxwidthnormal boxaligncenter');
461
462             if ($this->overridelock) {
463                 $params = 'pageid=' . $this->page->id;
464
465                 if ($this->section) {
466                     $params .= '&section=' . urlencode($this->section);
467                 }
468
469                 $form = '<form method="post" action="' . $CFG->wwwroot . '/mod/
                     wikicode/overridelocks.php?' . $params . '">';
```

```
470                 $form .= '<input type="hidden" name="sesskey" value="' . sesskey()
                        . '" />';
471                 $form .= '<input type="submit" value="' . get_string('overridelocks
                        ', 'wikicode') . '" />';
472                 $form .= '</form>';
473 
474                 print $OUTPUT->box($form, 'generalbox boxwidthnormal boxaligncenter
                        ');
475             }
476             return false;
477         }*/
478         return true;
479     }
480 
481     protected function print_edit($content = null) {
482         global $CFG, $OUTPUT, $USER, $PAGE;
483 
484         if (!$this->check_locks()) {
485             return;
486         }
487 
488         //delete old locks (> 1 hour)
489         wikicode_delete_old_locks();
490 
491         $version = wikicode_get_current_version($this->page->id);
492                 $page = wikicode_get_page($this->page->id);
493 
494         $format = $version->contentformat;
495 
496         if ($content == null) {
497             if (empty($this->section)) {
498                 $content = $version->content;
499             } else {
500                 $content = $this->sectioncontent;
501             }
502         }
503 
504         $versionnumber = $version->version;
505         if ($this->versionnumber >= 0) {
506             if ($version->version != $this->versionnumber) {
507                 print $OUTPUT->box(get_string('wrongversionlock', 'wikicode'), '
                        errorbox');
508                 $versionnumber = $this->versionnumber;
509             }
510         }
511 
512         $url = $CFG->wwwroot . '/mod/wikicode/edit.php?pageid=' . $this->page->id;
513         if (!empty($this->section)) {
514             $url .= "&section=" . urlencode($this->section);
515         }
516 
517         $params = array('attachmentoptions' => page_wikicode_edit::
                $attachmentoptions, 'format' => $version->contentformat, 'version' =>
                $versionnumber, 'pagetitle'=>$this->page->title);
```

```
518
519         $data = new StdClass();
520         $data->newcontent = wikicode_remove_tags_owner($content);
521         $data->version = $versionnumber;
522         $data->format = $format;
523
524         switch ($format) {
525         case 'html':
526             $data->newcontentformat = FORMAT_HTML;
527             // Append editor context to editor options, giving preference to
                    existing context.
528             page_wikicode_edit::$attachmentoptions = array_merge(array('context' =>
                     $this->modcontext), page_wikicode_edit::$attachmentoptions);
529             $data = file_prepare_standard_editor($data, 'newcontent',
                    page_wikicode_edit::$attachmentoptions, $this->modcontext, '
                    mod_wikicode', 'attachments', $this->subwiki->id);
530             break;
531         default:
532             break;
533             }
534
535         if ($version->contentformat != 'html') {
536             $params['fileitemid'] = $this->subwiki->id;
537             $params['contextid']  = $this->modcontext->id;
538             $params['component']  = 'mod_wikicode';
539             $params['filearea']   = 'attachments';
540         }
541
542         if (!empty($CFG->usetags)) {
543             $params['tags'] = tag_get_tags_csv('wikicode_pages', $this->page->id,
                    TAG_RETURN_TEXT);
544         }
545
546         $form = new mod_wikicode_edit_form($url, $params);
547
548         if ($formdata = $form->get_data()) {
549             if (!empty($CFG->usetags)) {
550                 $data->tags = $formdata->tags;
551             }
552         } else {
553             if (!empty($CFG->usetags)) {
554                 $data->tags = tag_get_tags_array('wikicode', $this->page->id);
555             }
556         }
557
558                 if ( $this->compiled == 1 ) {
559                         $data->newcontent = wikicode_remove_tags_owner($page->
                                cachedcompile);
560                         $data->textCompiler = $page->cachedgcc;
561                 }
562
563         $form->set_data($data);
564         $form->display();
565     }
```

```
566
567 }
568
569 /**
570  * Wiki page editing page
571  *
572  * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
573  */
574 class page_wikicode_log extends page_wikicode {
575
576         protected $sectioncontent;
577     /** @var string the section name needed to be edited */
578     protected $section;
579     protected $overridelock = false;
580     protected $versionnumber = -1;
581
582     function __construct($wiki, $subwiki, $cm) {
583         global $CFG, $PAGE;
584         parent::__construct($wiki, $subwiki, $cm);
585         //$PAGE->requires->js_init_call('M.mod_wikicode.renew_lock', null, true);
586         $PAGE->requires->yui2_lib('connection');
587     }
588
589     protected function print_pagetitle() {
590         global $OUTPUT;
591
592         $title = $this->title;
593         if (isset($this->section)) {
594             $title .= ' : ' . $this->section;
595         }
596         echo $OUTPUT->container_start('wikicode_clear');
597         echo $OUTPUT->heading(format_string($title), 2, 'wikicode_headingtitle');
598         echo $OUTPUT->container_end();
599     }
600
601     function print_header() {
602         global $OUTPUT, $PAGE;
603
604         parent::print_header();
605
606         $this->print_pagetitle();
607     }
608
609     function print_content() {
610         global $PAGE;
611
612         if (wikicode_user_can_edit($this->subwiki)) {
613             $this->print_log();
614         } else {
615             // @TODO: Translate it
616             echo "You can not edit this page";
617         }
618     }
619
```

```
620     protected function set_url() {
621         global $PAGE, $CFG;
622 
623         $params = array('pageid' => $this->page->id);
624 
625         if (isset($this->section)) {
626             $params['section'] = $this->section;
627         }
628 
629         $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/log.php', $params);
630     }
631 
632     protected function set_session_url() {
633         global $SESSION;
634 
635         $SESSION->wikipreviousurl = array('page' => 'log', 'params' => array('
                pageid' => $this->page->id, 'section' => $this->section));
636     }
637 
638     protected function process_session_url() {
639     }
640 
641     protected function create_navbar() {
642         global $PAGE, $CFG;
643 
644         parent::create_navbar();
645 
646         $PAGE->navbar->add(get_string('log', 'wikicode'));
647     }
648 
649     protected function check_locks() {
650         global $OUTPUT, $USER, $CFG;
651 
652         return true;
653     }
654 
655     protected function print_log() {
656         global $CFG, $OUTPUT, $USER, $PAGE;
657 
658         if (!$this->check_locks()) {
659             return;
660         }
661 
662         //delete old locks (> 1 hour)
663         wikicode_delete_old_locks();
664 
665         $version = wikicode_get_current_version($this->page->id);
666                 $page = wikicode_get_page($this->page->id);
667 
668         $format = $version->contentformat;
669 
670         $params = array('attachmentoptions' => page_wikicode_edit::
                $attachmentoptions, 'format' => $version->contentformat, 'version' =>
                $versionnumber, 'pagetitle'=>$this->page->title);
```

```
671
672         $data = new StdClass();
673                 $params['page'] = $this->page;
674
675         $form = new mod_wikicode_log_form($url, $params);
676
677         $form->set_data($data);
678         $form->display();
679     }
680
681 }
682
683 /**
684  * Class that models the behavior of wiki's view comments page
685  *
686  * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
687  */
688 class page_wikicode_comments extends page_wikicode {
689
690     function print_header() {
691
692         parent::print_header();
693
694         $this->print_pagetitle();
695
696     }
697
698     function print_content() {
699         global $CFG, $OUTPUT, $USER, $PAGE;
700         require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
701
702         $page = $this->page;
703         $subwiki = $this->subwiki;
704         $wiki = $PAGE->activityrecord;
705         list($context, $course, $cm) = get_context_info_array($this->modcontext->id
                ) ;
706
707         require_capability('mod/wikicode:viewcomment', $this->modcontext, NULL,
                true, 'noviewcommentpermission', 'wikicode');
708
709         $comments = wikicode_get_comments($this->modcontext->id, $page->id);
710
711         if (has_capability('mod/wikicode:editcomment', $this->modcontext)) {
712             echo '<div class="midpad"><a href="' . $CFG->wwwroot . '/mod/wikicode/
                    editcomments.php?action=add&amp;pageid=' . $page->id . '">' .
                    get_string('addcomment', 'wikicode') . '</a></div>';
713         }
714
715         $options = array('swid' => $this->page->subwikiid, 'pageid' => $page->id);
716         $version = wikicode_get_current_version($this->page->id);
717         $format = $version->contentformat;
718
719         if (empty($comments)) {
720             echo $OUTPUT->heading(get_string('nocomments', 'wikicode'));
```

```
721         }
722
723         foreach ($comments as $comment) {
724
725             $user = wikicode_get_user_info($comment->userid);
726
727             $fullname = fullname($user, has_capability('moodle/site:viewfullnames',
                    get_context_instance(CONTEXT_COURSE, $course->id)));
728             $by = new stdclass();
729             $by->name = '<a href="' . $CFG->wwwroot . '/user/view.php?id=' . $user
                    ->id . '&amp;course=' . $course->id . '">' . $fullname . '</a>';
730             $by->date = userdate($comment->timecreated);
731
732             $t = new html_table();
733             $cell1 = new html_table_cell($OUTPUT->user_picture($user, array('popup'
                    => true)));
734             $cell2 = new html_table_cell(get_string('bynameondate', 'forum', $by));
735             $cell3 = new html_table_cell();
736             $cell3->atributtes ['width'] = "80%";
737             $cell4 = new html_table_cell();
738             $cell5 = new html_table_cell();
739
740             $row1 = new html_table_row();
741             $row1->cells[] = $cell1;
742             $row1->cells[] = $cell2;
743             $row2 = new html_table_row();
744             $row2->cells[] = $cell3;
745
746             if ($format != 'html') {
747                 if ($format == 'creole') {
748                     $parsedcontent = wikicode_parse_content('creole', $comment->
                            content, $options);
749                 } else if ($format == 'nwiki') {
750                     $parsedcontent = wikicode_parse_content('nwiki', $comment->
                            content, $options);
751                 }
752
753                 $cell4->text = format_text(html_entity_decode($parsedcontent['
                        parsed_text']), FORMAT_HTML);
754             } else {
755                 $cell4->text = format_text($comment->content, FORMAT_HTML);
756             }
757
758             $row2->cells[] = $cell4;
759
760             $t->data = array($row1, $row2);
761
762             $actionicons = false;
763             if ((has_capability('mod/wikicode:managecomment', $this->modcontext)))
                    {
764                 $urledit = new moodle_url('/mod/wikicode/editcomments.php', array('
                        commentid' => $comment->id, 'pageid' => $page->id, 'action' =>
                        'edit'));
```

```
765                 $urldelet = new moodle_url('/mod/wikicode/instancecomments.php',
                        array('commentid' => $comment->id, 'pageid' => $page->id, '
                        action' => 'delete'));
766                 $actionicons = true;
767             } else if ((has_capability('mod/wikicode:editcomment', $this->
                    modcontext)) and ($USER->id == $user->id)) {
768                 $urledit = new moodle_url('/mod/wikicode/editcomments.php', array('
                        commentid' => $comment->id, 'pageid' => $page->id, 'action' =>
                        'edit'));
769                 $urldelet = new moodle_url('/mod/wikicode/instancecomments.php',
                        array('commentid' => $comment->id, 'pageid' => $page->id, '
                        action' => 'delete'));
770                 $actionicons = true;
771             }
772
773             if ($actionicons) {
774                 $cell6 = new html_table_cell($OUTPUT->action_icon($urledit, new
                        pix_icon('t/edit', get_string('edit'))) . $OUTPUT->action_icon(
                        $urldelet, new pix_icon('t/delete', get_string('delete'))));
775                 $row3 = new html_table_row();
776                 $row3->cells[] = $cell5;
777                 $row3->cells[] = $cell6;
778                 $t->data[] = $row3;
779             }
780
781             echo html_writer::tag('div', html_writer::table($t), array('class'=>'no
                    -overflow'));
782
783         }
784     }
785
786     function set_url() {
787         global $PAGE, $CFG;
788         $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/comments.php', array('pageid'
                 => $this->page->id));
789     }
790
791     protected function create_navbar() {
792         global $PAGE, $CFG;
793
794         parent::create_navbar();
795         $PAGE->navbar->add(get_string('comments', 'wikicode'));
796     }
797
798 }
799
800 /**
801  * Class that models the behavior of wiki's edit comment
802  *
803  * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
804  */
805 class page_wikicode_editcomment extends page_wikicode {
806     private $comment;
807     private $action;
```

```
808     private $form;
809     private $format;
810 
811     function set_url() {
812         global $PAGE, $CFG;
813         $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/comments.php', array('pageid'
                => $this->page->id));
814     }
815 
816     function print_header() {
817         parent::print_header();
818         $this->print_pagetitle();
819     }
820 
821     function print_content() {
822         global $PAGE;
823 
824         require_capability('mod/wikicode:editcomment', $this->modcontext, NULL,
                true, 'noeditcommentpermission', 'wikicode');
825 
826         if ($this->action == 'add') {
827             $this->add_comment_form();
828         } else if ($this->action == 'edit') {
829             $this->edit_comment_form($this->comment);
830         }
831     }
832 
833     function set_action($action, $comment) {
834         global $CFG;
835         require_once($CFG->dirroot . '/mod/wikicode/comments_form.php');
836 
837         $this->action = $action;
838         $this->comment = $comment;
839         $version = wikicode_get_current_version($this->page->id);
840         $this->format = $version->contentformat;
841 
842         if ($this->format == 'html') {
843             $destination = $CFG->wwwroot . '/mod/wikicode/instancecomments.php?
                    pageid=' . $this->page->id;
844             $this->form = new mod_wikicode_comments_form($destination);
845         }
846     }
847 
848     protected function create_navbar() {
849         global $PAGE, $CFG;
850 
851         $PAGE->navbar->add(get_string('comments', 'wikicode'), $CFG->wwwroot . '/
                mod/wikicode/comments.php?pageid=' . $this->page->id);
852 
853         if ($this->action == 'add') {
854             $PAGE->navbar->add(get_string('insertcomment', 'wikicode'));
855         } else {
856             $PAGE->navbar->add(get_string('editcomment', 'wikicode'));
857         }
```

```
858     }
859
860     protected function setup_tabs() {
861         parent::setup_tabs(array('linkedwhenactive' => 'comments', 'activetab' => '
                comments'));
862     }
863
864     private function add_comment_form() {
865         global $CFG;
866         require_once($CFG->dirroot . '/mod/wikicode/editors/wiki_editor.php');
867
868         $pageid = $this->page->id;
869
870         if ($this->format == 'html') {
871             $com = new stdClass();
872             $com->action = 'add';
873             $com->commentoptions = array('trusttext' => true, 'maxfiles' => 0);
874             $this->form->set_data($com);
875             $this->form->display();
876         } else {
877             wikicode_print_editor_wiki($this->page->id, null, $this->format, -1,
                    null, false, null, 'addcomments');
878         }
879     }
880
881     private function edit_comment_form($com) {
882         global $CFG;
883         require_once($CFG->dirroot . '/mod/wikicode/comments_form.php');
884         require_once($CFG->dirroot . '/mod/wikicode/editors/wiki_editor.php');
885
886         if ($this->format == 'html') {
887             $com->action = 'edit';
888             $com->entrycomment_editor['text'] = $com->content;
889             $com->commentoptions = array('trusttext' => true, 'maxfiles' => 0);
890
891             $this->form->set_data($com);
892             $this->form->display();
893         } else {
894             wikicode_print_editor_wiki($this->page->id, $com->content, $this->
                    format, -1, null, false, array(), 'editcomments', $com->id);
895         }
896
897     }
898
899 }
900
901 /**
902  * Wiki page search page
903  *
904  * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
905  */
906 class page_wikicode_search extends page_wikicode {
907     private $search_result;
908
```

```
909     protected function create_navbar() {
910         global $PAGE, $CFG;
911 
912         $PAGE->navbar->add(format_string($this->title));
913     }
914 
915     function set_search_string($search, $searchcontent) {
916         $swid = $this->subwiki->id;
917         if ($searchcontent) {
918             $this->search_result = wikicode_search_all($swid, $search);
919         } else {
920             $this->search_result = wikicode_search_title($swid, $search);
921         }
922 
923     }
924 
925     function set_url() {
926         global $PAGE, $CFG;
927         $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/search.php');
928     }
929     function print_content() {
930         global $PAGE;
931 
932         require_capability('mod/wikicode:viewpage', $this->modcontext, NULL, true,
                'noviewpagepermission', 'wikicode');
933 
934         echo $this->wikioutput->search_result($this->search_result, $this->subwiki)
                ;
935     }
936 }
937 
938 /**
939  *
940  * Class that models the behavior of wiki's
941  * create page
942  *
943  */
944 class page_wikicode_create extends page_wikicode {
945 
946     private $format;
947     private $swid;
948     private $wid;
949     private $action;
950     private $mform;
951 
952     function print_header() {
953         $this->set_url();
954         parent::print_header();
955     }
956 
957     function set_url() {
958         global $PAGE, $CFG;
959 
960         $params = array();
```

```
961             if ($this->action == 'new') {
962                 $params['action'] = 'new';
963                 $params['swid'] = $this->swid;
964                 $params['wid'] = $this->wid;
965                 if ($this->title != get_string('newpage', 'wikicode')) {
966                     $params['title'] = $this->title;
967                 }
968                 $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/create.php', $params);
969             } else {
970                 $params['action'] = 'create';
971                 $params['swid'] = $this->swid;
972                 $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/create.php', $params);
973             }
974         }
975 
976         function set_format($format) {
977             $this->format = $format;
978         }
979 
980         function set_wid($wid) {
981             $this->wid = $wid;
982         }
983 
984         function set_swid($swid) {
985             $this->swid = $swid;
986         }
987 
988         function set_action($action) {
989             global $PAGE;
990             $this->action = $action;
991 
992             require_once(dirname(__FILE__) . '/create_form.php');
993             $url = new moodle_url('/mod/wikicode/create.php', array('action' => 'create
                    ', 'wid' => $PAGE->activityrecord->id, 'gid' => $this->gid, 'uid' =>
                    $this->uid));
994             $formats = wikicode_get_formats();
995             $options = array('formats' => $formats, 'defaultformat' => $PAGE->
                    activityrecord->defaultformat, 'forceformat' => $PAGE->activityrecord->
                    forceformat);
996             if ($this->title != get_string('newpage', 'wikicode')) {
997                 $options['disable_pagetitle'] = true;
998             }
999             $this->mform = new mod_wikicode_create_form($url->out(false), $options);
1000        }
1001 
1002        protected function create_navbar() {
1003            global $PAGE;
1004 
1005            $PAGE->navbar->add($this->title);
1006        }
1007 
1008        function print_content($pagetitle = '') {
1009            global $PAGE;
1010 
```

```
1011            // @TODO: Change this to has_capability and show an alternative interface.
1012            require_capability('mod/wikicode:createpage', $this->modcontext, NULL, true
                    , 'nocreatepermission', 'wikicode');
1013            $data = new stdClass();
1014            if (!empty($pagetitle)) {
1015                $data->pagetitle = $pagetitle;
1016            }
1017            $data->pageformat = $PAGE->activityrecord->defaultformat;
1018
1019            $this->mform->set_data($data);
1020            $this->mform->display();
1021        }
1022
1023        function create_page($pagetitle) {
1024            global $USER, $CFG, $PAGE;
1025            $data = $this->mform->get_data();
1026            if (empty($this->subwiki)) {
1027                $swid = wikicode_add_subwiki($PAGE->activityrecord->id, $this->gid,
                        $this->uid);
1028                $this->subwiki = wikicode_get_subwiki($swid);
1029            }
1030            if ($data) {
1031                $id = wikicode_create_page($this->subwiki->id, $data->pagetitle, $data
                        ->pageformat, $USER->id);
1032            } else {
1033                $id = wikicode_create_page($this->subwiki->id, $pagetitle, $PAGE->
                        activityrecord->defaultformat, $USER->id);
1034            }
1035            redirect($CFG->wwwroot . '/mod/wikicode/edit.php?pageid=' . $id);
1036        }
1037    }
1038
1039    /**
1040     * Class that models the behavior of wiki's
1041     * compile page
1042     *
1043     */
1044    class page_wikicode_compile extends page_wikicode_edit {
1045
1046        private $newcontent, $download;
1047
1048        function print_header() {
1049        }
1050
1051        function print_content() {
1052            global $PAGE;
1053
1054            $context = get_context_instance(CONTEXT_MODULE, $PAGE->cm->id);
1055            require_capability('mod/wikicode:editpage', $context, NULL, true, '
                    noeditpermission', 'wikicode');
1056
1057            $this->print_compile();
1058        }
1059
```

```
1060        function set_newcontent($newcontent) {
1061            $this->newcontent = $newcontent;
1062        }
1063
1064            function set_download($download) {
1065                    $this->download = $download;
1066            }
1067
1068        protected function set_session_url() {
1069        }
1070
1071        protected function print_compile() {
1072            global $CFG, $USER, $OUTPUT, $PAGE;
1073
1074            $url = $CFG->wwwroot . '/mod/wikicode/edit.php?pageid=' . $this->page->id;
1075            if (!empty($this->section)) {
1076                $url .= "&section=" . urlencode($this->section);
1077            }
1078
1079            $params = array('attachmentoptions' => page_wikicode_edit::
                $attachmentoptions, 'format' => $this->format, 'version' => $this->
                versionnumber);
1080
1081            if ($this->format != 'html') {
1082                $params['fileitemid'] = $this->page->id;
1083                $params['contextid']  = $this->modcontext->id;
1084                $params['component']  = 'mod_wikicode';
1085                $params['filearea']   = 'attachments';
1086            }
1087
1088            $form = new mod_wikicode_edit_form($url, $params);
1089
1090            $save = false;
1091            $data = false;
1092            if ($data = $form->get_data()) {
1093
1094                $save = wikicode_compile_page($this->page, $data->newcontent, $USER->id
                    , $this->download);
1095
1096                //deleting old locks
1097                wikicode_delete_locks($this->page->id, $USER->id, $this->section);
1098
1099                redirect($CFG->wwwroot . '/mod/wikicode/edit.php?compiled=1&pageid=' .
                    $this->page->id);
1100            } else {
1101                print_error('savingerror', 'wikicode');
1102            }
1103        }
1104    }
1105
1106    /**
1107     *
1108     * Class that models the behavior of wiki's
1109     * view differences
```

```
1110     *
1111     */
1112    class page_wikicode_diff extends page_wikicode {
1113
1114        private $compare;
1115        private $comparewith;
1116
1117        function print_header() {
1118            global $OUTPUT;
1119
1120            parent::print_header();
1121
1122            $this->print_pagetitle();
1123            $vstring = new stdClass();
1124            $vstring->old = $this->compare;
1125            $vstring->new = $this->comparewith;
1126            echo $OUTPUT->heading(get_string('comparewith', 'wikicode', $vstring));
1127        }
1128
1129        /**
1130         * Print the diff view
1131         */
1132        function print_content() {
1133            global $PAGE;
1134
1135            require_capability('mod/wikicode:viewpage', $this->modcontext, NULL, true,
                    'noviewpagepermission', 'wikicode');
1136
1137            $this->print_diff_content();
1138        }
1139
1140        function set_url() {
1141            global $PAGE, $CFG;
1142
1143            $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/diff.php', array('pageid' =>
                    $this->page->id, 'comparewith' => $this->comparewith, 'compare' =>
                    $this->compare));
1144        }
1145
1146        function set_comparison($compare, $comparewith) {
1147            $this->compare = $compare;
1148            $this->comparewith = $comparewith;
1149        }
1150
1151        protected function create_navbar() {
1152            global $PAGE, $CFG;
1153
1154            parent::create_navbar();
1155            $PAGE->navbar->add(get_string('history', 'wikicode'), $CFG->wwwroot . '/mod
                    /wikicode/history.php?pageid' . $this->page->id);
1156            $PAGE->navbar->add(get_string('diff', 'wikicode'));
1157        }
1158
1159        protected function setup_tabs() {
```

```
1160            parent::setup_tabs(array('linkedwhenactive' => 'history', 'activetab' => '
                    history'));
1161        }
1162
1163        /**
1164         * Given two versions of a page, prints a page displaying the differences
                 between them.
1165         *
1166         * @global object $CFG
1167         * @global object $OUTPUT
1168         * @global object $PAGE
1169         */
1170        private function print_diff_content() {
1171            global $CFG, $OUTPUT, $PAGE;
1172
1173            $pageid = $this->page->id;
1174            $total = wikicode_count_wikicode_page_versions($pageid) - 1;
1175
1176            $oldversion = wikicode_get_wikicode_page_version($pageid, $this->compare);
1177
1178            $newversion = wikicode_get_wikicode_page_version($pageid, $this->
                    comparewith);
1179
1180            if ($oldversion && $newversion) {
1181
1182                $oldtext = format_text(wikicode_remove_tags($oldversion->content),
                        FORMAT_PLAIN, array('overflowdiv'=>true));
1183                        $newtext = format_text(wikicode_remove_tags($newversion->
                            content), FORMAT_PLAIN, array('overflowdiv'=>true));
1184                list($diff1, $diff2) = ouwiki_diff_html($oldtext, $newtext);
1185                $oldversion->diff = $diff1;
1186                $oldversion->user = wikicode_get_user_info($oldversion->userid);
1187                $newversion->diff = $diff2;
1188                $newversion->user = wikicode_get_user_info($newversion->userid);
1189
1190                echo $this->wikioutput->diff($pageid, $oldversion, $newversion, array('
                        total' => $total));
1191            } else {
1192                print_error('versionerror', 'wikicode');
1193            }
1194        }
1195    }
1196
1197    /**
1198     *
1199     * Class that models the behavior of wiki's history page
1200     *
1201     */
1202    class page_wikicode_history extends page_wikicode {
1203        /**
1204         * @var int $paging current page
1205         */
1206        private $paging;
1207
```

```
1208        /**
1209         * @var int @rowsperpage Items per page
1210         */
1211        private $rowsperpage = 10;
1212
1213        /**
1214         * @var int $allversion if $allversion != 0, all versions will be printed in a
                 signle table
1215         */
1216        private $allversion;
1217
1218        function __construct($wiki, $subwiki, $cm) {
1219            global $PAGE;
1220            parent::__construct($wiki, $subwiki, $cm);
1221            $PAGE->requires->js_init_call('M.mod_wikicode.history', null, true);
1222        }
1223
1224        function print_header() {
1225            parent::print_header();
1226            $this->print_pagetitle();
1227        }
1228
1229        function print_pagetitle() {
1230            global $OUTPUT;
1231            $html = '';
1232
1233            $html .= $OUTPUT->container_start();
1234            $html .= $OUTPUT->heading_with_help(format_string($this->title), 'history',
                     'wikicode');
1235            $html .= $OUTPUT->container_end();
1236            echo $html;
1237        }
1238
1239        function print_content() {
1240            global $PAGE;
1241
1242            require_capability('mod/wikicode:viewpage', $this->modcontext, NULL, true,
                    'noviewpagepermission', 'wikicode');
1243
1244            $this->print_history_content();
1245        }
1246
1247        function set_url() {
1248            global $PAGE, $CFG;
1249            $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/history.php', array('pageid'
                    => $this->page->id));
1250        }
1251
1252        function set_paging($paging) {
1253            $this->paging = $paging;
1254        }
1255
1256        function set_allversion($allversion) {
1257            $this->allversion = $allversion;
```

```
1258        }
1259
1260        protected function create_navbar() {
1261            global $PAGE, $CFG;
1262
1263            parent::create_navbar();
1264            $PAGE->navbar->add(get_string('history', 'wikicode'));
1265        }
1266
1267        /**
1268         * Prints the history for a given wiki page
1269         *
1270         * @global object $CFG
1271         * @global object $OUTPUT
1272         * @global object $PAGE
1273         */
1274        private function print_history_content() {
1275            global $CFG, $OUTPUT, $PAGE;
1276
1277            $pageid = $this->page->id;
1278            $offset = $this->paging * $this->rowsperpage;
1279            // vcount is the latest version
1280            $vcount = wikicode_count_wikicode_page_versions($pageid) - 1;
1281            if ($this->allversion) {
1282                $versions = wikicode_get_wikicode_page_versions($pageid, 0, $vcount);
1283            } else {
1284                $versions = wikicode_get_wikicode_page_versions($pageid, $offset, $this
                        ->rowsperpage);
1285            }
1286            // We don't want version 0 to be displayed
1287            // version 0 is blank page
1288            if (end($versions)->version == 0) {
1289                array_pop($versions);
1290            }
1291
1292            $contents = array();
1293
1294            $version0page = wikicode_get_wikicode_page_version($this->page->id, 0);
1295            $creator = wikicode_get_user_info($version0page->userid);
1296            $a = new StdClass;
1297            $a->date = userdate($this->page->timecreated, get_string('
                    strftimedaydatetime', 'langconfig'));
1298            $a->username = fullname($creator);
1299            echo $OUTPUT->heading(get_string('createddate', 'wikicode', $a), 4, '
                    wikicode_headingtime');
1300            if ($vcount > 0) {
1301
1302                /// If there is only one version, we don't need radios nor forms
1303                if (count($versions) == 1) {
1304
1305                    $row = array_shift($versions);
1306
1307                    $username = wikicode_get_user_info($row->userid);
1308                    $picture = $OUTPUT->user_picture($username);
```

```
1309                    $date = userdate($row->timecreated, get_string('strftimedate', '
                            langconfig'));
1310                    $time = userdate($row->timecreated, get_string('strftimetime', '
                            langconfig'));
1311                    $versionid = wikicode_get_version($row->id);
1312                    $versionlink = new moodle_url('/mod/wikicode/viewversion.php',
                            array('pageid' => $pageid, 'versionid' => $versionid->id));
1313                    $userlink = new moodle_url('/user/view.php', array('id' =>
                            $username->id));
1314                    $contents[] = array('', html_writer::link($versionlink->out(false),
                             $row->version), $picture . html_writer::link($userlink->out(
                            false), fullname($username)), $time, $OUTPUT->container($date,
                            'wikicode_histdate'));
1315
1316                    $table = new html_table();
1317                    $table->head = array('', get_string('version'), get_string('user'),
                             get_string('modified'), '');
1318                    $table->data = $contents;
1319                    $table->attributes['class'] = 'mdl-align';
1320
1321                    echo html_writer::table($table);
1322
1323                } else {
1324
1325                    $checked = $vcount - $offset;
1326                    $rowclass = array();
1327
1328                    foreach ($versions as $version) {
1329                        $user = wikicode_get_user_info($version->userid);
1330                        $picture = $OUTPUT->user_picture($user, array('popup' => true))
                                ;
1331                        $date = userdate($version->timecreated, get_string('
                                strftimedate'));
1332                        $rowclass[] = 'wikicode_histnewdate';
1333                        $time = userdate($version->timecreated, get_string('
                                strftimetime', 'langconfig'));
1334                        $versionid = wikicode_get_version($version->id);
1335                        if ($versionid) {
1336                            $url = new moodle_url('/mod/wikicode/viewversion.php',
                                    array('pageid' => $pageid, 'versionid' => $versionid->
                                    id));
1337                            $viewlink = html_writer::link($url->out(false), $version->
                                    version);
1338                        } else {
1339                            $viewlink = $version->version;
1340                        }
1341                        $userlink = new moodle_url('/user/view.php', array('id' =>
                                $version->userid));
1342                        $contents[] = array($this->choose_from_radio(array($version->
                                version  => null), 'compare', 'M.mod_wikicode.history()',
                                $checked - 1, true) . $this->choose_from_radio(array(
                                $version->version  => null), 'comparewith', 'M.mod_wikicode
                                .history()', $checked, true), $viewlink, $picture .
```

```
                        html_writer::link($userlink->out(false), fullname($user)),
                        $time, $OUTPUT->container($date, 'wikicode_histdate'));
1343                }
1344
1345                $table = new html_table();
1346
1347                $icon = $OUTPUT->help_icon('diff', 'wikicode');
1348
1349                $table->head = array(get_string('diff', 'wikicode') . $icon,
                        get_string('version'), get_string('user'), get_string('modified
                        '), '');
1350                $table->data = $contents;
1351                $table->attributes['class'] = 'generaltable mdl-align';
1352                $table->rowclasses = $rowclass;
1353
1354                /*$table = new StdClass();
1355                 $table->head = array(helpbutton('diff', 'diff', 'wikicode', true,
                            false, '', true, ''),
1356                 get_string('version'),
1357                 get_string('user'),
1358                 get_string('modified'),
1359                 '');
1360                 $table->data = $contents;
1361                 $table->class = 'mdl-align';
1362                 $table->rowclass = $rowclass;*/
1363
1364                ///Print the form
1365                echo html_writer::start_tag('form', array('action'=>new moodle_url(
                        '/mod/wikicode/diff.php'), 'method'=>'get', 'id'=>'diff'));
1366                echo html_writer::tag('div', html_writer::empty_tag('input', array(
                        'type'=>'hidden', 'name'=>'pageid', 'value'=>$pageid)));
1367                echo html_writer::table($table);
1368                echo html_writer::start_tag('div', array('class'=>'mdl-align'));
1369                echo html_writer::empty_tag('input', array('type'=>'submit', 'class
                        '=>'wikicode_form-button', 'value'=>get_string('comparesel', '
                        wikicode')));
1370                echo html_writer::end_tag('div');
1371                echo html_writer::end_tag('form');
1372            }
1373        } else {
1374            print_string('nohistory', 'wikicode');
1375        }
1376        if (!$this->allversion) {
1377            //$pagingbar = moodle_paging_bar::make($vcount, $this->paging, $this->
                    rowsperpage, $CFG->wwwroot.'/mod/wikicode/history.php?pageid='.
                    $pageid.'&amp;');
1378            // $pagingbar->pagevar = $pagevar;
1379            echo $OUTPUT->paging_bar($vcount, $this->paging, $this->rowsperpage,
                    $CFG->wwwroot . '/mod/wikicode/history.php?pageid=' . $pageid . '&
                    amp;');
1380            //print_paging_bar($vcount, $paging, $rowsperpage,$CFG->wwwroot.'/mod/
                    wikicode/history.php?pageid='.$pageid.'&amp;','paging');
1381            } else {
```

```
1382                $link = new moodle_url('/mod/wikicode/history.php', array('pageid' =>
                        $pageid));
1383                $OUTPUT->container(html_writer::link($link->out(false), get_string('
                        viewperpage', 'wikicode', $this->rowsperpage)), 'mdl-align');
1384            }
1385            if ($vcount > $this->rowsperpage && !$this->allversion) {
1386                $link = new moodle_url('/mod/wikicode/history.php', array('pageid' =>
                        $pageid, 'allversion' => 1));
1387                $OUTPUT->container(html_writer::link($link->out(false), get_string('
                        viewallhistory', 'wikicode')), 'mdl-align');
1388            }
1389        }
1390
1391        /**
1392         * Given an array of values, creates a group of radio buttons to be part of a
                form
1393         *
1394         * @param array  $options  An array of value-label pairs for the radio group (
                values as keys).
1395         * @param string $name     Name of the radiogroup (unique in the form).
1396         * @param string $onclick  Function to be executed when the radios are clicked.
1397         * @param string $checked  The value that is already checked.
1398         * @param bool   $return   If true, return the HTML as a string, otherwise
                print it.
1399         *
1400         * @return mixed If $return is false, returns nothing, otherwise returns a
                string of HTML.
1401         */
1402        private function choose_from_radio($options, $name, $onclick = '', $checked = '
                ', $return = false) {
1403
1404            static $idcounter = 0;
1405
1406            if (!$name) {
1407                $name = 'unnamed';
1408            }
1409
1410            $output = '<span class="radiogroup ' . $name . "\">\n";
1411
1412            if (!empty($options)) {
1413                $currentradio = 0;
1414                foreach ($options as $value => $label) {
1415                    $htmlid = 'auto-rb' . sprintf('%04d', ++$idcounter);
1416                    $output .= ' <span class="radioelement ' . $name . ' rb' .
                            $currentradio . "\">";
1417                    $output .= '<input name="' . $name . '" id="' . $htmlid . '" type="
                            radio" value="' . $value . '"';
1418                    if ($value == $checked) {
1419                        $output .= ' checked="checked"';
1420                    }
1421                    if ($onclick) {
1422                        $output .= ' onclick="' . $onclick . '"';
1423                    }
1424                    if ($label === '') {
```

```
1425                        $output .= ' /> <label for="' . $htmlid . '">' . $value . '</
                                label></span>' . "\n";
1426                    } else {
1427                        $output .= ' /> <label for="' . $htmlid . '">' . $label . '</
                                label></span>' . "\n";
1428                    }
1429                    $currentradio = ($currentradio + 1) % 2;
1430                }
1431            }
1432
1433            $output .= '</span>' . "\n";
1434
1435            if ($return) {
1436                return $output;
1437            } else {
1438                echo $output;
1439            }
1440        }
1441    }
1442
1443    /**
1444     * Class that models the behavior of wiki's map page
1445     *
1446     */
1447    class page_wikicode_map extends page_wikicode {
1448
1449        /**
1450         * @var int wiki view option
1451         */
1452        private $view;
1453
1454        function print_header() {
1455            parent::print_header();
1456            $this->print_pagetitle();
1457        }
1458
1459        function print_content() {
1460            global $CFG, $PAGE;
1461
1462            require_capability('mod/wikicode:viewpage', $this->modcontext, NULL, true,
                    'noviewpagepermission', 'wikicode');
1463
1464            if ($this->view > 0) {
1465                //echo '<div><a href="' . $CFG->wwwroot . '/mod/wikicode/map.php?pageid
                        =' . $this->page->id . '">' . get_string('backtomapmenu', 'wikicode
                        ') . '</a></div>';
1466            }
1467
1468            switch ($this->view) {
1469            case 1:
1470                echo $this->wikioutput->menu_map($this->page->id, $this->view);
1471                $this->print_contributions_content();
1472                break;
1473            case 2:
```

```
1474                echo $this->wikioutput->menu_map($this->page->id, $this->view);
1475                $this->print_navigation_content();
1476                break;
1477            case 3:
1478                echo $this->wikioutput->menu_map($this->page->id, $this->view);
1479                $this->print_orphaned_content();
1480                break;
1481            case 4:
1482                echo $this->wikioutput->menu_map($this->page->id, $this->view);
1483                $this->print_index_content();
1484                break;
1485            case 5:
1486                echo $this->wikioutput->menu_map($this->page->id, $this->view);
1487                $this->print_page_list_content();
1488                break;
1489            case 6:
1490                echo $this->wikioutput->menu_map($this->page->id, $this->view);
1491                $this->print_updated_content();
1492                break;
1493            default:
1494                echo $this->wikioutput->menu_map($this->page->id, $this->view);
1495                $this->print_page_list_content();
1496            }
1497        }
1498
1499        function set_view($option) {
1500            $this->view = $option;
1501        }
1502
1503        function set_url() {
1504            global $PAGE, $CFG;
1505            $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/map.php', array('pageid' =>
                    $this->page->id));
1506        }
1507
1508        protected function create_navbar() {
1509            global $PAGE;
1510
1511            parent::create_navbar();
1512            $PAGE->navbar->add(get_string('map', 'wikicode'));
1513        }
1514
1515        /**
1516         * Prints the contributions tab content
1517         *
1518         * @uses $OUTPUT, $USER
1519         *
1520         */
1521        private function print_contributions_content() {
1522            global $CFG, $OUTPUT, $USER;
1523            $page = $this->page;
1524
1525            if ($page->timerendered + wikicode_REFRESH_CACHE_TIME < time()) {
1526                $fresh = wikicode_refresh_cachedcontent($page);
```

```
1527                $page = $fresh['page'];
1528            }
1529
1530            $swid = $this->subwiki->id;
1531
1532            $table = new html_table();
1533            $table->head = array(get_string('contributions', 'wikicode') . $OUTPUT->
                    help_icon('contributions', 'wikicode'));
1534            $table->attributes['class'] = 'wikicode_editor generalbox';
1535            $table->data = array();
1536            $table->rowclasses = array();
1537
1538            $lastversions = array();
1539            $pages = array();
1540            $users = array();
1541
1542            if ($contribs = wikicode_get_contributions($swid, $USER->id)) {
1543                foreach ($contribs as $contrib) {
1544                    if (!array_key_exists($contrib->pageid, $pages)) {
1545                        $page = wikicode_get_page($contrib->pageid);
1546                        $pages[$contrib->pageid] = $page;
1547                    } else {
1548                        continue;
1549                    }
1550
1551                    if (!array_key_exists($page->id, $lastversions)) {
1552                        $version = wikicode_get_last_version($page->id);
1553                        $lastversions[$page->id] = $version;
1554                    } else {
1555                        $version = $lastversions[$page->id];
1556                    }
1557
1558                    if (!array_key_exists($version->userid, $users)) {
1559                        $user = wikicode_get_user_info($version->userid);
1560                        $users[$version->userid] = $user;
1561                    } else {
1562                        $user = $users[$version->userid];
1563                    }
1564
1565                    $link = wikicode_parser_link(format_string($page->title), array('
                            swid' => $swid));
1566                    $class = ($link['new']) ? 'class="wiki_newentry"' : '';
1567
1568                    $linkpage = '<a href="' . $link['url'] . '"' . $class . '>' . $link
                            ['content'] . '</a>';
1569                    $icon = $OUTPUT->user_picture($user, array('popup' => true));
1570
1571                    $table->data[] = array("$icon $linkpage");
1572                }
1573            } else {
1574                $table->data[] = array(get_string('nocontribs', 'wikicode'));
1575            }
1576            echo html_writer::table($table);
1577        }
```

```
1578
1579        /**
1580         * Prints the navigation tab content
1581         *
1582         * @uses $OUTPUT
1583         *
1584         */
1585        private function print_navigation_content() {
1586            global $OUTPUT;
1587            $page = $this->page;
1588
1589            if ($page->timerendered + wikicode_REFRESH_CACHE_TIME < time()) {
1590                $fresh = wikicode_refresh_cachedcontent($page);
1591                $page = $fresh['page'];
1592            }
1593
1594            $tolinks = wikicode_get_linked_to_pages($page->id);
1595            $fromlinks = wikicode_get_linked_from_pages($page->id);
1596
1597            $table = new html_table();
1598            $table->attributes['class'] = 'wikicode_navigation_from';
1599            $table->head = array(get_string('navigationfrom', 'wikicode') . $OUTPUT->
                help_icon('navigationfrom', 'wikicode') . ':');
1600            $table->data = array();
1601            $table->rowclasses = array();
1602            foreach ($fromlinks as $link) {
1603                $lpage = wikicode_get_page($link->frompageid);
1604                $link = new moodle_url('/mod/wikicode/view.php', array('pageid' =>
                    $lpage->id));
1605                $table->data[] = array(html_writer::link($link->out(false),
                    format_string($lpage->title)));
1606                $table->rowclasses[] = 'mdl-align';
1607            }
1608
1609            $table_left = html_writer::table($table);
1610
1611            $table = new html_table();
1612            $table->attributes['class'] = 'wikicode_navigation_to';
1613            $table->head = array(get_string('navigationto', 'wikicode') . $OUTPUT->
                help_icon('navigationto', 'wikicode') . ':');
1614            $table->data = array();
1615            $table->rowclasses = array();
1616            foreach ($tolinks as $link) {
1617                if ($link->tomissingpage) {
1618                    $viewlink = new moodle_url('/mod/wikicode/create.php', array('swid'
                        => $page->subwikiid, 'title' => $link->tomissingpage, 'action'
                        => 'new'));
1619                    $table->data[] = array(html_writer::link($viewlink->out(false),
                        format_string($link->tomissingpage), array('class' => '
                        wikicode_newentry')));
1620                } else {
1621                    $lpage = wikicode_get_page($link->topageid);
1622                    $viewlink = new moodle_url('/mod/wikicode/view.php', array('pageid'
                        => $lpage->id));
```

```
1623                    $table->data[] = array(html_writer::link($viewlink->out(false),
                            format_string($lpage->title)));
1624                }
1625                $table->rowclasses[] = 'mdl-align';
1626            }
1627            $table_right = html_writer::table($table);
1628            echo $OUTPUT->container($table_left . $table_right, '
                    wikicode_navigation_container');
1629        }
1630
1631        /**
1632         * Prints the index page tab content
1633         *
1634         *
1635         */
1636        private function print_index_content() {
1637            global $OUTPUT;
1638            $page = $this->page;
1639
1640            if ($page->timerendered + wikicode_REFRESH_CACHE_TIME < time()) {
1641                $fresh = wikicode_refresh_cachedcontent($page);
1642                $page = $fresh['page'];
1643            }
1644
1645            $node = new navigation_node($page->title);
1646
1647            $keys = array();
1648            $tree = array();
1649            $tree = wikicode_build_tree($page, $node, $keys);
1650
1651            $table = new html_table();
1652            $table->head = array(get_string('pageindex', 'wikicode') . $OUTPUT->
                    help_icon('pageindex', 'wikicode'));
1653            $table->attributes['class'] = 'wikicode_editor generalbox';
1654            $table->data[] = array($this->render_navigation_node($tree));
1655
1656            echo html_writer::table($table);
1657        }
1658
1659        /**
1660         * Prints the page list tab content
1661         *
1662         *
1663         */
1664        private function print_page_list_content() {
1665            global $OUTPUT;
1666            $page = $this->page;
1667
1668            if ($page->timerendered + wikicode_REFRESH_CACHE_TIME < time()) {
1669                $fresh = wikicode_refresh_cachedcontent($page);
1670                $page = $fresh['page'];
1671            }
1672
1673            $pages = wikicode_get_page_list($this->subwiki->id);
```

```
1674
1675            $stdaux = new stdClass();
1676            $strspecial = get_string('special', 'wikicode');
1677
1678            foreach ($pages as $page) {
1679                $letter = textlib::strtoupper(textlib::substr($page->title, 0, 1));
1680                if (preg_match('/[A-Z]/', $letter)) {
1681                    $stdaux->{
1682                        $letter}
1683                    [] = wikicode_parser_link($page);
1684                } else {
1685                    $stdaux->{
1686                        $strspecial}
1687                    [] = wikicode_parser_link($page);
1688                }
1689            }
1690
1691            $table = new html_table();
1692            $table->head = array(get_string('pagelist', 'wikicode') . $OUTPUT->
                help_icon('pagelist', 'wikicode'));
1693            $table->attributes['class'] = 'wikicode_editor generalbox';
1694            $table->align = array('center');
1695            foreach ($stdaux as $key => $elem) {
1696                $table->data[] = array($key);
1697                foreach ($elem as $e) {
1698                    $table->data[] = array(html_writer::link($e['url'], $e['content']))
                        ;
1699                }
1700            }
1701            echo html_writer::table($table);
1702        }
1703
1704        /**
1705         * Prints the orphaned tab content
1706         *
1707         *
1708         */
1709        private function print_orphaned_content() {
1710            global $OUTPUT;
1711
1712            $page = $this->page;
1713
1714            if ($page->timerendered + wikicode_REFRESH_CACHE_TIME < time()) {
1715                $fresh = wikicode_refresh_cachedcontent($page);
1716                $page = $fresh['page'];
1717            }
1718
1719            $swid = $this->subwiki->id;
1720
1721            $table = new html_table();
1722            $table->head = array(get_string('orphaned', 'wikicode') . $OUTPUT->
                help_icon('orphaned', 'wikicode'));
1723            $table->attributes['class'] = 'wikicode_editor generalbox';
1724            $table->data = array();
```

```
1725            $table->rowclasses = array();
1726
1727            if ($orphanedpages = wikicode_get_orphaned_pages($swid)) {
1728                foreach ($orphanedpages as $page) {
1729                    $link = wikicode_parser_link($page->title, array('swid' => $swid));
1730                    $class = ($link['new']) ? 'class="wiki_newentry"' : '';
1731                    $table->data[] = array('<a href="' . $link['url'] . '"' . $class .
                        '>' . format_string($link['content']) . '</a>');
1732                }
1733            } else {
1734                $table->data[] = array(get_string('noorphanedpages', 'wikicode'));
1735            }
1736
1737            echo html_writer::table($table);
1738        }
1739
1740        /**
1741         * Prints the updated tab content
1742         *
1743         * @uses $COURSE, $OUTPUT
1744         *
1745         */
1746        private function print_updated_content() {
1747            global $COURSE, $OUTPUT;
1748            $page = $this->page;
1749
1750            if ($page->timerendered + wikicode_REFRESH_CACHE_TIME < time()) {
1751                $fresh = wikicode_refresh_cachedcontent($page);
1752                $page = $fresh['page'];
1753            }
1754
1755            $swid = $this->subwiki->id;
1756
1757            $table = new html_table();
1758            $table->head = array(get_string('updatedpages', 'wikicode') . $OUTPUT->
                help_icon('updatedpages', 'wikicode'));
1759            $table->attributes['class'] = 'wikicode_editor generalbox';
1760            $table->data = array();
1761            $table->rowclasses = array();
1762
1763            if ($pages = wikicode_get_updated_pages_by_subwiki($swid)) {
1764                $strdataux = '';
1765                foreach ($pages as $page) {
1766                    $user = wikicode_get_user_info($page->userid);
1767                    $strdata = strftime('%d %b %Y', $page->timemodified);
1768                    if ($strdata != $strdataux) {
1769                        $table->data[] = array($OUTPUT->heading($strdata, 4));
1770                        $strdataux = $strdata;
1771                    }
1772                    $link = wikicode_parser_link($page->title, array('swid' => $swid));
1773                    $class = ($link['new']) ? 'class="wiki_newentry"' : '';
1774
1775                    $linkpage = '<a href="' . $link['url'] . '"' . $class . '>' .
                        format_string($link['content']) . '</a>';
```

```
1776                    $icon = $OUTPUT->user_picture($user, array($COURSE->id));
1777                    $table->data[] = array("$icon $linkpage");
1778                }
1779            } else {
1780                $table->data[] = array(get_string('noupdatedpages', 'wikicode'));
1781            }
1782
1783            echo html_writer::table($table);
1784        }
1785
1786        protected function render_navigation_node($items, $attrs = array(),
                $expansionlimit = null, $depth = 1) {
1787
1788            // exit if empty, we don't want an empty ul element
1789            if (count($items) == 0) {
1790                return '';
1791            }
1792
1793            // array of nested li elements
1794            $lis = array();
1795            foreach ($items as $item) {
1796                if (!$item->display) {
1797                    continue;
1798                }
1799                $content = $item->get_content();
1800                $title = $item->get_title();
1801                if ($item->icon instanceof renderable) {
1802                    $icon = $this->wikioutput->render($item->icon);
1803                    $content = $icon . ' ' . $content; // use CSS for spacing of
                        icons
1804                    }
1805                if ($item->helpbutton !== null) {
1806                    $content = trim($item->helpbutton) . html_writer::tag('span',
                        $content, array('class' => 'clearhelpbutton'));
1807                }
1808
1809                if ($content === '') {
1810                    continue;
1811                }
1812
1813                if ($item->action instanceof action_link) {
1814                    //TODO: to be replaced with something else
1815                    $link = $item->action;
1816                    if ($item->hidden) {
1817                        $link->add_class('dimmed');
1818                    }
1819                    $content = $this->output->render($link);
1820                } else if ($item->action instanceof moodle_url) {
1821                    $attributes = array();
1822                    if ($title !== '') {
1823                        $attributes['title'] = $title;
1824                    }
1825                    if ($item->hidden) {
1826                        $attributes['class'] = 'dimmed_text';
```

```
1827                    }
1828                    $content = html_writer::link($item->action, $content, $attributes);
1829
1830                } else if (is_string($item->action) || empty($item->action)) {
1831                    $attributes = array();
1832                    if ($title !== '') {
1833                        $attributes['title'] = $title;
1834                    }
1835                    if ($item->hidden) {
1836                        $attributes['class'] = 'dimmed_text';
1837                    }
1838                    $content = html_writer::tag('span', $content, $attributes);
1839                }
1840
1841                // this applies to the li item which contains all child lists too
1842                $liclasses = array($item->get_css_type(), 'depth_' . $depth);
1843                if ($item->has_children() && (!$item->forceopen || $item->collapse)) {
1844                    $liclasses[] = 'collapsed';
1845                }
1846                if ($item->isactive === true) {
1847                    $liclasses[] = 'current_branch';
1848                }
1849                $liattr = array('class' => join(' ', $liclasses));
1850                // class attribute on the div item which only contains the item content
1851                $divclasses = array('tree_item');
1852                if ((empty($expansionlimit) || $item->type != $expansionlimit) && (
                        $item->children->count() > 0 || ($item->nodetype == navigation_node
                        ::NODETYPE_BRANCH && $item->children->count() == 0 && isloggedin())
                        )) {
1853                    $divclasses[] = 'branch';
1854                } else {
1855                    $divclasses[] = 'leaf';
1856                }
1857                if (!empty($item->classes) && count($item->classes) > 0) {
1858                    $divclasses[] = join(' ', $item->classes);
1859                }
1860                $divattr = array('class' => join(' ', $divclasses));
1861                if (!empty($item->id)) {
1862                    $divattr['id'] = $item->id;
1863                }
1864                $content = html_writer::tag('p', $content, $divattr) . $this->
                        render_navigation_node($item->children, array(), $expansionlimit,
                        $depth + 1);
1865                if (!empty($item->preceedwithhr) && $item->preceedwithhr === true) {
1866                    $content = html_writer::empty_tag('hr') . $content;
1867                }
1868                $content = html_writer::tag('li', $content, $liattr);
1869                $lis[] = $content;
1870            }
1871
1872            if (count($lis)) {
1873                return html_writer::tag('ul', implode("\n", $lis), $attrs);
1874            } else {
1875                return '';
```

```
1876            }
1877        }
1878
1879    }
1880
1881    /**
1882     * Class that models the behavior of wiki's restore version page
1883     *
1884     */
1885    class page_wikicode_restoreversion extends page_wikicode {
1886        private $version;
1887
1888        function print_header() {
1889            parent::print_header();
1890            $this->print_pagetitle();
1891        }
1892
1893        function print_content() {
1894            global $CFG, $PAGE;
1895
1896            require_capability('mod/wikicode:managewiki', $this->modcontext, NULL, true
                    , 'nomanagewikipermission', 'wikicode');
1897
1898            $this->print_restoreversion();
1899        }
1900
1901        function set_url() {
1902            global $PAGE, $CFG;
1903            $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/viewversion.php', array('
                    pageid' => $this->page->id, 'versionid' => $this->version->id));
1904        }
1905
1906        function set_versionid($versionid) {
1907            $this->version = wikicode_get_version($versionid);
1908        }
1909
1910        protected function create_navbar() {
1911            global $PAGE, $CFG;
1912
1913            parent::create_navbar();
1914            $PAGE->navbar->add(get_string('restoreversion', 'wikicode'));
1915        }
1916
1917        protected function setup_tabs() {
1918            parent::setup_tabs(array('linkedwhenactive' => 'history', 'activetab' => '
                    history'));
1919        }
1920
1921        /**
1922         * Prints the restore version content
1923         *
1924         * @uses $CFG
1925         *
1926         * @param page $page The page whose version will be restored
```

```
1927        * @param int  $versionid The version to be restored
1928        * @param bool $confirm If false, shows a yes/no confirmation page.
1929        *     If true, restores the old version and redirects the user to the 'view'
                tab.
1930        */
1931       private function print_restoreversion() {
1932           global $OUTPUT;
1933
1934           $version = wikicode_get_version($this->version->id);
1935
1936           $optionsyes = array('confirm'=>1, 'pageid'=>$this->page->id, 'versionid'=>
                $version->id, 'sesskey'=>sesskey());
1937           $restoreurl = new moodle_url('/mod/wikicode/restoreversion.php',
                $optionsyes);
1938           $return = new moodle_url('/mod/wikicode/viewversion.php', array('pageid'=>
                $this->page->id, 'versionid'=>$version->id));
1939
1940           echo $OUTPUT->heading(get_string('restoreconfirm', 'wikicode', $version->
                version), 2);
1941           print_container_start(false, 'wikicode_restoreform');
1942           echo '<form class="wiki_restore_yes" action="' . $restoreurl . '" method="
                post" id="restoreversion">';
1943           echo '<div><input type="submit" name="confirm" value="' . get_string('yes')
                . '" /></div>';
1944           echo '</form>';
1945           echo '<form class="wiki_restore_no" action="' . $return . '" method="post">
                ';
1946           echo '<div><input type="submit" name="norestore" value="' . get_string('no'
                ) . '" /></div>';
1947           echo '</form>';
1948           print_container_end();
1949       }
1950   }
1951   /**
1952    * Class that models the behavior of wiki's delete comment confirmation page
1953    *
1954    */
1955   class page_wikicode_deletecomment extends page_wikicode {
1956       private $commentid;
1957
1958       function print_header() {
1959           parent::print_header();
1960           $this->print_pagetitle();
1961       }
1962
1963       function print_content() {
1964           $this->printconfirmdelete();
1965       }
1966
1967       function set_url() {
1968           global $PAGE;
1969           $PAGE->set_url('/mod/wikicode/instancecomments.php', array('pageid' =>
                $this->page->id, 'commentid' => $this->commentid));
1970       }
```

```
1971
1972        public function set_action($action, $commentid, $content) {
1973            $this->action = $action;
1974            $this->commentid = $commentid;
1975            $this->content = $content;
1976        }
1977
1978        protected function create_navbar() {
1979            global $PAGE;
1980
1981            parent::create_navbar();
1982            $PAGE->navbar->add(get_string('deletecommentcheck', 'wikicode'));
1983        }
1984
1985        protected function setup_tabs() {
1986            parent::setup_tabs(array('linkedwhenactive' => 'comments', 'activetab' => '
                    comments'));
1987        }
1988
1989        /**
1990         * Prints the comment deletion confirmation form
1991         *
1992         * @param page $page The page whose version will be restored
1993         * @param int  $versionid The version to be restored
1994         * @param bool $confirm If false, shows a yes/no confirmation page.
1995         *     If true, restores the old version and redirects the user to the 'view'
                 tab.
1996         */
1997        private function printconfirmdelete() {
1998            global $OUTPUT;
1999
2000            $strdeletecheck = get_string('deletecommentcheck', 'wikicode');
2001            $strdeletecheckfull = get_string('deletecommentcheckfull', 'wikicode');
2002
2003            //ask confirmation
2004            $optionsyes = array('confirm'=>1, 'pageid'=>$this->page->id, 'action'=>'
                    delete', 'commentid'=>$this->commentid, 'sesskey'=>sesskey());
2005            $deleteurl = new moodle_url('/mod/wikicode/instancecomments.php',
                    $optionsyes);
2006            $return = new moodle_url('/mod/wikicode/comments.php', array('pageid'=>
                    $this->page->id));
2007
2008            echo $OUTPUT->heading($strdeletecheckfull);
2009            print_container_start(false, 'wikicode_deletecommentform');
2010            echo '<form class="wiki_deletecomment_yes" action="' . $deleteurl . '"
                    method="post" id="deletecomment">';
2011            echo '<div><input type="submit" name="confirmdeletecomment" value="' .
                    get_string('yes') . '" /></div>';
2012            echo '</form>';
2013            echo '<form class="wiki_deletecomment_no" action="' . $return . '" method="
                    post">';
2014            echo '<div><input type="submit" name="norestore" value="' . get_string('no'
                    ) . '" /></div>';
2015            echo '</form>';
```

```
2016            print_container_end();
2017        }
2018    }
2019
2020    /**
2021     * Class that models the behavior of wiki's
2022     * save page
2023     *
2024     */
2025    class page_wikicode_save extends page_wikicode_edit {
2026
2027        private $newcontent;
2028
2029        function print_header() {
2030        }
2031
2032        function print_content() {
2033            global $PAGE;
2034
2035            $context = get_context_instance(CONTEXT_MODULE, $PAGE->cm->id);
2036            require_capability('mod/wikicode:editpage', $context, NULL, true, '
                    noeditpermission', 'wikicode');
2037
2038            $this->print_save();
2039        }
2040
2041        function set_newcontent($newcontent) {
2042            $this->newcontent = $newcontent;
2043        }
2044
2045        protected function set_session_url() {
2046        }
2047
2048        protected function print_save() {
2049            global $CFG, $USER, $OUTPUT, $PAGE;
2050
2051            $url = $CFG->wwwroot . '/mod/wikicode/edit.php?pageid=' . $this->page->id;
2052            if (!empty($this->section)) {
2053                $url .= "&section=" . urlencode($this->section);
2054            }
2055
2056            $params = array('attachmentoptions' => page_wikicode_edit::
                    $attachmentoptions, 'format' => $this->format, 'version' => $this->
                    versionnumber);
2057
2058            if ($this->format != 'html') {
2059                $params['fileitemid'] = $this->page->id;
2060                $params['contextid']  = $this->modcontext->id;
2061                $params['component']  = 'mod_wikicode';
2062                $params['filearea']   = 'attachments';
2063            }
2064
2065            $form = new mod_wikicode_edit_form($url, $params);
2066
```

```
2067            $save = false;
2068            $data = false;
2069            if ($data = $form->get_data()) {
2070                if ($this->format == 'html') {
2071                    $data = file_postupdate_standard_editor($data, 'newcontent',
                            page_wikicode_edit::$attachmentoptions, $this->modcontext, '
                            mod_wikicode', 'attachments', $this->subwiki->id);
2072                }
2073
2074                if (isset($this->section)) {
2075                    $save = wikicode_save_section($this->page, $this->section, $data->
                            newcontent, $USER->id);
2076                } else {
2077                    $save = wikicode_save_page($this->page, $data->newcontent, $USER->
                            id);
2078                }
2079            }
2080
2081            if ($save && $data) {
2082                if (!empty($CFG->usetags)) {
2083                    tag_set('wikicode_pages', $this->page->id, $data->tags);
2084                }
2085
2086                $message = '<p>' . get_string('saving', 'wikicode') . '</p>';
2087
2088                if (!empty($save['sections'])) {
2089                    foreach ($save['sections'] as $s) {
2090                        $message .= '<p>' . get_string('repeatedsection', 'wikicode',
                                $s) . '</p>';
2091                    }
2092                }
2093
2094                if ($this->versionnumber + 1 != $save['version']) {
2095                    $message .= '<p>' . get_string('wrongversionsave', 'wikicode') . '
                            </p>';
2096                }
2097
2098                if (isset($errors) && !empty($errors)) {
2099                    foreach ($errors as $e) {
2100                        $message .= "<p>" . get_string('filenotuploadederror', '
                                wikicode', $e->get_filename()) . "</p>";
2101                    }
2102                }
2103
2104                //deleting old locks
2105                wikicode_delete_locks($this->page->id, $USER->id, $this->section);
2106
2107                redirect($CFG->wwwroot . '/mod/wikicode/edit.php?pageid=' . $this->page
                        ->id);
2108            } else {
2109                print_error('savingerror', 'wikicode');
2110            }
2111        }
2112    }
```

```
2113
2114  /**
2115   * Class that models the behavior of wiki's view an old version of a page
2116   *
2117   */
2118  class page_wikicode_viewversion extends page_wikicode {
2119
2120      private $version;
2121
2122      function print_header() {
2123          parent::print_header();
2124          $this->print_pagetitle();
2125      }
2126
2127      function print_content() {
2128          global $PAGE;
2129
2130          require_capability('mod/wikicode:viewpage', $this->modcontext, NULL, true,
                  'noviewpagepermission', 'wikicode');
2131
2132          $this->print_version_view();
2133      }
2134
2135      function set_url() {
2136          global $PAGE, $CFG;
2137          $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/viewversion.php', array('
                  pageid' => $this->page->id, 'versionid' => $this->version->id));
2138      }
2139
2140      function set_versionid($versionid) {
2141          $this->version = wikicode_get_version($versionid);
2142      }
2143
2144      protected function create_navbar() {
2145          global $PAGE, $CFG;
2146
2147          parent::create_navbar();
2148          $PAGE->navbar->add(get_string('history', 'wikicode'), $CFG->wwwroot . '/mod
                  /wikicode/history.php?pageid' . $this->page->id);
2149          $PAGE->navbar->add(get_string('versionnum', 'wikicode', $this->version->
                  version));
2150      }
2151
2152      protected function setup_tabs() {
2153          parent::setup_tabs(array('linkedwhenactive' => 'history', 'activetab' => '
                  history', 'inactivetabs' => array('edit')));
2154      }
2155
2156      /**
2157       * Given an old page version, output the version content
2158       *
2159       * @global object $CFG
2160       * @global object $OUTPUT
2161       * @global object $PAGE
```

```
2162         */
2163     private function print_version_view() {
2164         global $CFG, $OUTPUT, $PAGE;
2165         $pageversion = wikicode_get_version($this->version->id);
2166 
2167         if ($pageversion) {
2168             $restorelink = new moodle_url('/mod/wikicode/restoreversion.php', array
                    ('pageid' => $this->page->id, 'versionid' => $this->version->id));
2169             echo $OUTPUT->heading(get_string('viewversion', 'wikicode',
                    $pageversion->version) . '<br />' . html_writer::link($restorelink
                    ->out(false), '(' . get_string('restorethis', 'wikicode') . ')',
                    array('class' => 'wikicode_restore')) . ' ', 4);
2170             $userinfo = wikicode_get_user_info($pageversion->userid);
2171             $heading = '<p><strong>' . get_string('modified', 'wikicode') . ':</
                    strong> ' . userdate($pageversion->timecreated, get_string('
                    strftimedatetime', 'langconfig'));
2172             $viewlink = new moodle_url('/user/view.php', array('id' => $userinfo->
                    id));
2173             $heading .= '   <strong>' . get_string('user') . ':</
                    strong> ' . html_writer::link($viewlink->out(false), fullname(
                    $userinfo));
2174             $heading .= '  &rarr; ' . $OUTPUT->user_picture(
                    wikicode_get_user_info($pageversion->userid), array('popup' => true
                    )) . '</p>';
2175             print_container($heading, false, 'mdl-align wikicode_modifieduser
                    wikicode_headingtime');
2176             $options = array('swid' => $this->subwiki->id, 'pretty_print' => true,
                    'pageid' => $this->page->id);
2177 
2178             $pageversion->content = file_rewrite_pluginfile_urls(
                    wikicode_remove_tags($pageversion->content), 'pluginfile.php',
                    $this->modcontext->id, 'mod_wikicode', 'attachments', $this->
                    subwiki->id);
2179                 $content = format_text($pageversion->content, FORMAT_PLAIN, array('
                        overflowdiv'=>true));
2180 
2181             echo $OUTPUT->box($content, 'generalbox wikicode_contentbox');
2182 
2183         } else {
2184             print_error('versionerror', 'wikicode');
2185         }
2186     }
2187 }
2188 
2189 class page_wikicode_confirmrestore extends page_wikicode_save {
2190 
2191     private $version;
2192 
2193     function set_url() {
2194         global $PAGE, $CFG;
2195         $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/viewversion.php', array('
                pageid' => $this->page->id, 'versionid' => $this->version->id));
2196     }
2197 
```

```
2198        function print_content() {
2199            global $CFG, $PAGE;
2200
2201            require_capability('mod/wikicode:managewiki', $this->modcontext, NULL, true
                    , 'nomanagewikipermission', 'wikicode');
2202
2203            $version = wikicode_get_version($this->version->id);
2204            if (wikicode_restore_page($this->page, $version->content, $version->userid)
                    ) {
2205                redirect($CFG->wwwroot . '/mod/wikicode/view.php?pageid=' . $this->page
                        ->id, get_string('restoring', 'wikicode', $version->version), 3);
2206            } else {
2207                print_error('restoreerror', 'wikicode', $version->version);
2208            }
2209        }
2210
2211        function set_versionid($versionid) {
2212            $this->version = wikicode_get_version($versionid);
2213        }
2214    }
2215
2216    class page_wikicode_prettyview extends page_wikicode {
2217
2218        function print_header() {
2219            global $CFG, $PAGE, $OUTPUT;
2220            $PAGE->set_pagelayout('embedded');
2221            echo $OUTPUT->header();
2222
2223            echo '<h1 id="wiki_printable_title">' . format_string($this->title) . '</h1
                    >';
2224        }
2225
2226        function print_content() {
2227            global $PAGE;
2228
2229            require_capability('mod/wikicode:viewpage', $this->modcontext, NULL, true,
                    'noviewpagepermission', 'wikicode');
2230
2231            $this->print_pretty_view();
2232        }
2233
2234        function set_url() {
2235            global $PAGE, $CFG;
2236
2237            $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/prettyview.php', array('
                    pageid' => $this->page->id));
2238        }
2239
2240        private function print_pretty_view() {
2241            $version = wikicode_get_current_version($this->page->id);
2242
2243            $content = wikicode_parse_content($version->contentformat, $version->
                    content, array('printable' => true, 'swid' => $this->subwiki->id, '
                    pageid' => $this->page->id, 'pretty_print' => true));
```

```
2244
2245            echo '<div id="wiki_printable_content">';
2246            echo format_text($content['parsed_text'], FORMAT_HTML);
2247            echo '</div>';
2248        }
2249    }
2250
2251    class page_wikicode_handlecomments extends page_wikicode {
2252        private $action;
2253        private $content;
2254        private $commentid;
2255        private $format;
2256
2257        function print_header() {
2258            $this->set_url();
2259        }
2260
2261        public function print_content() {
2262            global $CFG, $PAGE, $USER;
2263
2264            if ($this->action == 'add') {
2265                if (has_capability('mod/wikicode:editcomment', $this->modcontext)) {
2266                    $this->add_comment($this->content, $this->commentid);
2267                }
2268            } else if ($this->action == 'edit') {
2269                $comment = wikicode_get_comment($this->commentid);
2270                $edit = has_capability('mod/wikicode:editcomment', $this->modcontext);
2271                $owner = ($comment->userid == $USER->id);
2272                if ($owner && $edit) {
2273                    $this->add_comment($this->content, $this->commentid);
2274                }
2275            } else if ($this->action == 'delete') {
2276                $comment = wikicode_get_comment($this->commentid);
2277                $manage = has_capability('mod/wikicode:managecomment', $this->
                        modcontext);
2278                $owner = ($comment->userid == $USER->id);
2279                if ($owner || $manage) {
2280                    $this->delete_comment($this->commentid);
2281                    redirect($CFG->wwwroot . '/mod/wikicode/comments.php?pageid=' .
                            $this->page->id, get_string('deletecomment', 'wikicode'), 2);
2282                }
2283            }
2284
2285        }
2286
2287        public function set_url() {
2288            global $PAGE, $CFG;
2289            $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/comments.php', array('pageid'
                 => $this->page->id));
2290        }
2291
2292        public function set_action($action, $commentid, $content) {
2293            $this->action = $action;
2294            $this->commentid = $commentid;
```

```
2295            $this->content = $content;
2296
2297            $version = wikicode_get_current_version($this->page->id);
2298            $format = $version->contentformat;
2299
2300            $this->format = $format;
2301        }
2302
2303        private function add_comment($content, $idcomment) {
2304            global $CFG, $PAGE;
2305            require_once($CFG->dirroot . "/mod/wikicode/locallib.php");
2306
2307            $pageid = $this->page->id;
2308
2309            wikicode_add_comment($this->modcontext, $pageid, $content, $this->format);
2310
2311            if (!$idcomment) {
2312                redirect($CFG->wwwroot . '/mod/wikicode/comments.php?pageid=' . $pageid
                        , get_string('createcomment', 'wikicode'), 2);
2313            } else {
2314                $this->delete_comment($idcomment);
2315                redirect($CFG->wwwroot . '/mod/wikicode/comments.php?pageid=' . $pageid
                        , get_string('editingcomment', 'wikicode'), 2);
2316            }
2317        }
2318
2319        private function delete_comment($commentid) {
2320            global $CFG, $PAGE;
2321
2322            $pageid = $this->page->id;
2323
2324            wikicode_delete_comment($commentid, $this->modcontext, $pageid);
2325        }
2326
2327    }
2328
2329    class page_wikicode_lock extends page_wikicode_edit {
2330
2331        public function print_header() {
2332            $this->set_url();
2333        }
2334
2335        protected function set_url() {
2336            global $PAGE, $CFG;
2337
2338            $params = array('pageid' => $this->page->id);
2339
2340            if ($this->section) {
2341                $params['section'] = $this->section;
2342            }
2343
2344            $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/lock.php', $params);
2345        }
2346
```

```
2347        protected function set_session_url() {
2348        }
2349
2350        public function print_content() {
2351            global $USER, $PAGE;
2352
2353            require_capability('mod/wikicode:editpage', $this->modcontext, NULL, true,
                    'noeditpermission', 'wikicode');
2354
2355            wikicode_set_lock($this->page->id, $USER->id, $this->section);
2356        }
2357
2358        public function print_footer() {
2359        }
2360    }
2361
2362    class page_wikicode_overridelocks extends page_wikicode_edit {
2363        function print_header() {
2364            $this->set_url();
2365        }
2366
2367        function print_content() {
2368            global $CFG, $PAGE;
2369
2370            require_capability('mod/wikicode:overridelock', $this->modcontext, NULL,
                    true, 'nooverridelockpermission', 'wikicode');
2371
2372            wikicode_delete_locks($this->page->id, null, $this->section, true, true);
2373
2374            $args = "pageid=" . $this->page->id;
2375
2376            if (!empty($this->section)) {
2377                $args .= "&section=" . urlencode($this->section);
2378            }
2379
2380            redirect($CFG->wwwroot . '/mod/wikicode/edit.php?' . $args, get_string('
                    overridinglocks', 'wikicode'), 2);
2381        }
2382
2383        function set_url() {
2384            global $PAGE, $CFG;
2385
2386            $params = array('pageid' => $this->page->id);
2387
2388            if (!empty($this->section)) {
2389                $params['section'] = $this->section;
2390            }
2391
2392            $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/overridelocks.php', $params);
2393        }
2394
2395        protected function set_session_url() {
2396        }
2397
```

```
2398        private function print_overridelocks() {
2399            global $CFG;
2400
2401            wikicode_delete_locks($this->page->id, null, $this->section, true, true);
2402
2403            $args = "pageid=" . $this->page->id;
2404
2405            if (!empty($this->section)) {
2406                $args .= "&section=" . urlencode($this->section);
2407            }
2408
2409            redirect($CFG->wwwroot . '/mod/wikicode/edit.php?' . $args, get_string('
                    overridinglocks', 'wikicode'), 2);
2410        }
2411
2412    }
2413
2414    /**
2415     * This class will let user to delete wiki pages and page versions
2416     *
2417     */
2418    class page_wikicode_admin extends page_wikicode {
2419
2420        public $view, $action;
2421        public $listorphan = false;
2422
2423        /**
2424         * Constructor
2425         *
2426         * @global object $PAGE
2427         * @param mixed $wiki instance of wiki
2428         * @param mixed $subwiki instance of subwiki
2429         * @param stdClass $cm course module
2430         */
2431        function __construct($wiki, $subwiki, $cm) {
2432            global $PAGE;
2433            parent::__construct($wiki, $subwiki, $cm);
2434            $PAGE->requires->js_init_call('M.mod_wikicode.deleteversion', null, true);
2435        }
2436
2437        /**
2438         * Prints header for wiki page
2439         */
2440        function print_header() {
2441            parent::print_header();
2442            $this->print_pagetitle();
2443        }
2444
2445        /**
2446         * This function will display administration view to users with managewiki
                 capability
2447         */
2448        function print_content() {
2449            //make sure anyone trying to access this page has managewiki capabilities
```

```
2450            require_capability('mod/wikicode:managewiki', $this->modcontext, NULL, true
                    , 'noviewpagepermission', 'wikicode');
2451
2452            //update wiki cache if timedout
2453            $page = $this->page;
2454            if ($page->timerendered + wikicode_REFRESH_CACHE_TIME < time()) {
2455                $fresh = wikicode_refresh_cachedcontent($page);
2456                $page = $fresh['page'];
2457            }
2458
2459            //dispaly admin menu
2460            echo $this->wikioutput->menu_admin($this->page->id, $this->view);
2461
2462            //Display appropriate admin view
2463            switch ($this->view) {
2464                case 1: //delete page view
2465                    $this->print_delete_content($this->listorphan);
2466                    break;
2467                case 2: //delete version view
2468                    $this->print_delete_version();
2469                    break;
2470                default: //default is delete view
2471                    $this->print_delete_content($this->listorphan);
2472                    break;
2473            }
2474        }
2475
2476        /**
2477         * Sets admin view option
2478         *
2479         * @param int $view page view id
2480         * @param bool $listorphan is only valid for view 1.
2481         */
2482        public function set_view($view, $listorphan = true) {
2483            $this->view = $view;
2484            $this->listorphan = $listorphan;
2485        }
2486
2487        /**
2488         * Sets page url
2489         *
2490         * @global object $PAGE
2491         * @global object $CFG
2492         */
2493        function set_url() {
2494            global $PAGE, $CFG;
2495            $PAGE->set_url($CFG->wwwroot . '/mod/wikicode/admin.php', array('pageid' =>
                        $this->page->id));
2496        }
2497
2498        /**
2499         * sets navigation bar for the page
2500         *
2501         * @global object $PAGE
```

```
2502         */
2503        protected function create_navbar() {
2504            global $PAGE;
2505
2506            parent::create_navbar();
2507            $PAGE->navbar->add(get_string('admin', 'wikicode'));
2508        }
2509
2510        /**
2511         * Show wiki page delete options
2512         *
2513         * @param bool $showorphan
2514         */
2515        protected function print_delete_content($showorphan = true) {
2516            $contents = array();
2517            $table = new html_table();
2518            $table->head = array('','Page name');
2519            $table->attributes['class'] = 'generaltable mdl-align';
2520            $swid = $this->subwiki->id;
2521            if ($showorphan) {
2522                if ($orphanedpages = wikicode_get_orphaned_pages($swid)) {
2523                    $this->add_page_delete_options($orphanedpages, $swid, $table);
2524                } else {
2525                    $table->data[] = array('', get_string('noorphanedpages', 'wikicode'
                        ));
2526                }
2527            } else {
2528                if ($pages = wikicode_get_page_list($swid)) {
2529                    $this->add_page_delete_options($pages, $swid, $table);
2530                } else {
2531                    $table->data[] = array('', get_string('nopages', 'wikicode'));
2532                }
2533            }
2534
2535            ///Print the form
2536            echo html_writer::start_tag('form', array(
2537                                                    'action' => new moodle_url('/mod/
                                                        wikicode/admin.php'),
2538                                                    'method' => 'post'));
2539            echo html_writer::tag('div', html_writer::empty_tag('input', array(
2540                                                                                'type'  =>
                                                                                    '
                                                                                    hidden
                                                                                    ',
2541                                                                                'name'  =>
                                                                                     '
                                                                                    pageid
                                                                                    ',
2542                                                                                'value' =>
                                                                                     $this
                                                                                    ->page
                                                                                    ->id))
                                                                                    );
2543
```

```
2544            echo html_writer::empty_tag('input', array('type' => 'hidden', 'name' => '
                    option', 'value' => $this->view));
2545            echo html_writer::table($table);
2546            echo html_writer::start_tag('div', array('class' => 'mdl-align'));
2547            if (!$showorphan) {
2548                echo html_writer::empty_tag('input', array(
2549                                                        'type'    => 'submit',
2550                                                        'class'   => 'wikicode_form-
                                                            button',
2551                                                        'value'   => get_string('
                                                            listorphan', 'wikicode'),
2552                                                        'sesskey' => sesskey()));
2553            } else {
2554                echo html_writer::empty_tag('input', array('type'=>'hidden', 'name'=>'
                        listall', 'value'=>'1'));
2555                echo html_writer::empty_tag('input', array(
2556                                                        'type'    => 'submit',
2557                                                        'class'   => 'wikicode_form-
                                                            button',
2558                                                        'value'   => get_string('
                                                            listall', 'wikicode'),
2559                                                        'sesskey' => sesskey()));
2560            }
2561            echo html_writer::end_tag('div');
2562            echo html_writer::end_tag('form');
2563        }
2564
2565        /**
2566         * helper function for print_delete_content. This will add data to the table.
2567         *
2568         * @global object $OUTPUT
2569         * @param array $pages objects of wiki pages in subwiki
2570         * @param int $swid id of subwiki
2571         * @param object $table reference to the table in which data needs to be added
2572         */
2573        protected function add_page_delete_options($pages, $swid, &$table) {
2574            global $OUTPUT;
2575            foreach ($pages as $page) {
2576                $link = wikicode_parser_link($page->title, array('swid' => $swid));
2577                $class = ($link['new']) ? 'class="wiki_newentry"' : '';
2578                $pagelink = '<a href="' . $link['url'] . '"' . $class . '>' .
                        format_string($link['content']) . '</a>';
2579                $urledit = new moodle_url('/mod/wikicode/edit.php', array('pageid' =>
                        $page->id, 'sesskey' => sesskey()));
2580                $urldelete = new moodle_url('/mod/wikicode/admin.php', array(
2581                                                                    'pageid'  =>
                                                                        $this->page
                                                                        ->id,
2582                                                                    'delete'  =>
                                                                        $page->id,
2583                                                                    'option'  =>
                                                                        $this->view,
2584                                                                    'listall' => !
                                                                        $this->
```

```
                                                                    listorphan?'
                                                                    1': '',
2585                                                              'sesskey' =>
                                                                    sesskey()));
2586
2587            $editlinks = $OUTPUT->action_icon($urledit, new pix_icon('t/edit',
                    get_string('edit')));
2588            $editlinks .= $OUTPUT->action_icon($urldelete, new pix_icon('t/delete',
                     get_string('delete')));
2589            $table->data[] = array($editlinks, $pagelink);
2590        }
2591    }
2592
2593    /**
2594     * Prints lists of versions which can be deleted
2595     *
2596     * @global object $OUTPUT
2597     */
2598    private function print_delete_version() {
2599        global $OUTPUT;
2600        $pageid = $this->page->id;
2601
2602        // versioncount is the latest version
2603        $versioncount = wikicode_count_wikicode_page_versions($pageid) - 1;
2604        $versions = wikicode_get_wikicode_page_versions($pageid, 0, $versioncount);
2605
2606        // We don't want version 0 to be displayed
2607        // version 0 is blank page
2608        if (end($versions)->version == 0) {
2609            array_pop($versions);
2610        }
2611
2612        $contents = array();
2613        $version0page = wikicode_get_wikicode_page_version($this->page->id, 0);
2614        $creator = wikicode_get_user_info($version0page->userid);
2615        $a = new stdClass();
2616        $a->date = userdate($this->page->timecreated, get_string('
                strftimedaydatetime', 'langconfig'));
2617        $a->username = fullname($creator);
2618        echo $OUTPUT->heading(get_string('createddate', 'wikicode', $a), 4, '
                wikicode_headingtime');
2619        if ($versioncount > 0) {
2620            /// If there is only one version, we don't need radios nor forms
2621            if (count($versions) == 1) {
2622                $row = array_shift($versions);
2623                $username = wikicode_get_user_info($row->userid);
2624                $picture = $OUTPUT->user_picture($username);
2625                $date = userdate($row->timecreated, get_string('strftimedate', '
                        langconfig'));
2626                $time = userdate($row->timecreated, get_string('strftimetime', '
                        langconfig'));
2627                $versionid = wikicode_get_version($row->id);
2628                $versionlink = new moodle_url('/mod/wikicode/viewversion.php',
                        array('pageid' => $pageid, 'versionid' => $versionid->id));
```

```
2629                    $userlink = new moodle_url('/user/view.php', array('id' =>
                            $username->id));
2630                    $picturelink = $picture . html_writer::link($userlink->out(false),
                            fullname($username));
2631                    $historydate = $OUTPUT->container($date, 'wikicode_histdate');
2632                    $contents[] = array('', html_writer::link($versionlink->out(false),
                             $row->version), $picturelink, $time, $historydate);
2633
2634                    //Show current version
2635                    $table = new html_table();
2636                    $table->head = array('', get_string('version'), get_string('user'),
                             get_string('modified'), '');
2637                    $table->data = $contents;
2638                    $table->attributes['class'] = 'mdl-align';
2639
2640                    echo html_writer::table($table);
2641                } else {
2642                    $lastdate = '';
2643                    $rowclass = array();
2644
2645                    foreach ($versions as $version) {
2646                        $user = wikicode_get_user_info($version->userid);
2647                        $picture = $OUTPUT->user_picture($user, array('popup' => true))
                                ;
2648                        $date = userdate($version->timecreated, get_string('
                                strftimedate'));
2649                        if ($date == $lastdate) {
2650                            $date = '';
2651                            $rowclass[] = '';
2652                        } else {
2653                            $lastdate = $date;
2654                            $rowclass[] = 'wikicode_histnewdate';
2655                        }
2656
2657                        $time = userdate($version->timecreated, get_string('
                                strftimetime', 'langconfig'));
2658                        $versionid = wikicode_get_version($version->id);
2659                        if ($versionid) {
2660                            $url = new moodle_url('/mod/wikicode/viewversion.php',
                                    array('pageid' => $pageid, 'versionid' => $versionid->
                                    id));
2661                            $viewlink = html_writer::link($url->out(false), $version->
                                    version);
2662                        } else {
2663                            $viewlink = $version->version;
2664                        }
2665
2666                        $userlink = new moodle_url('/user/view.php', array('id' =>
                                $version->userid));
2667                        $picturelink = $picture . html_writer::link($userlink->out(
                                false), fullname($user));
2668                        $historydate = $OUTPUT->container($date, 'wikicode_histdate');
```

```
2669                        $radiofromelement = $this->choose_from_radio(array($version->
                                version  => null), 'fromversion', 'M.mod_wikicode.
                                deleteversion()', $versioncount, true);
2670                        $radiotoelement = $this->choose_from_radio(array($version->
                                version  => null), 'toversion', 'M.mod_wikicode.
                                deleteversion()', $versioncount, true);
2671                        $contents[] = array( $radiofromelement . $radiotoelement,
                                $viewlink, $picturelink, $time, $historydate);
2672                    }
2673
2674                    $table = new html_table();
2675                    $table->head = array(get_string('deleteversions', 'wikicode'),
                            get_string('version'), get_string('user'), get_string('modified
                            '), '');
2676                    $table->data = $contents;
2677                    $table->attributes['class'] = 'generaltable mdl-align';
2678                    $table->rowclasses = $rowclass;
2679
2680                    ///Print the form
2681                    echo html_writer::start_tag('form', array('action'=>new moodle_url(
                            '/mod/wikicode/admin.php'), 'method' => 'post'));
2682                    echo html_writer::tag('div', html_writer::empty_tag('input', array(
                            'type' => 'hidden', 'name' => 'pageid', 'value' => $pageid)));
2683                    echo html_writer::empty_tag('input', array('type' => 'hidden', '
                            name' => 'option', 'value' => $this->view));
2684                    echo html_writer::empty_tag('input', array('type' => 'hidden', '
                            name' => 'sesskey', 'value' =>  sesskey()));
2685                    echo html_writer::table($table);
2686                    echo html_writer::start_tag('div', array('class' => 'mdl-align'));
2687                    echo html_writer::empty_tag('input', array('type' => 'submit', '
                            class' => 'wikicode_form-button', 'value' => get_string('
                            deleteversions', 'wikicode')));
2688                    echo html_writer::end_tag('div');
2689                    echo html_writer::end_tag('form');
2690                }
2691            } else {
2692                print_string('nohistory', 'wikicode');
2693            }
2694        }
2695
2696        /**
2697         * Given an array of values, creates a group of radio buttons to be part of a
                form
2698         * helper function for print_delete_version
2699         *
2700         * @param array  $options  An array of value-label pairs for the radio group (
                values as keys).
2701         * @param string $name     Name of the radiogroup (unique in the form).
2702         * @param string $onclick  Function to be executed when the radios are clicked.
2703         * @param string $checked  The value that is already checked.
2704         * @param bool   $return   If true, return the HTML as a string, otherwise
                print it.
2705         *
```

```
2706         * @return mixed If $return is false, returns nothing, otherwise returns a
                 string of HTML.
2707         */
2708        private function choose_from_radio($options, $name, $onclick = '', $checked = '
                ', $return = false) {
2709
2710            static $idcounter = 0;
2711
2712            if (!$name) {
2713                $name = 'unnamed';
2714            }
2715
2716            $output = '<span class="radiogroup ' . $name . "\">\n";
2717
2718            if (!empty($options)) {
2719                $currentradio = 0;
2720                foreach ($options as $value => $label) {
2721                    $htmlid = 'auto-rb' . sprintf('%04d', ++$idcounter);
2722                    $output .= ' <span class="radioelement ' . $name . ' rb' .
                            $currentradio . "\">";
2723                    $output .= '<input name="' . $name . '" id="' . $htmlid . '" type="
                            radio" value="' . $value . '"';
2724                    if ($value == $checked) {
2725                        $output .= ' checked="checked"';
2726                    }
2727                    if ($onclick) {
2728                        $output .= ' onclick="' . $onclick . '"';
2729                    }
2730                    if ($label === '') {
2731                        $output .= ' /> <label for="' . $htmlid . '">' . $value . '</
                                label></span>' . "\n";
2732                    } else {
2733                        $output .= ' /> <label for="' . $htmlid . '">' . $label . '</
                                label></span>' . "\n";
2734                    }
2735                    $currentradio = ($currentradio + 1) % 2;
2736                }
2737            }
2738
2739            $output .= '</span>' . "\n";
2740
2741            if ($return) {
2742                return $output;
2743            } else {
2744                echo $output;
2745            }
2746        }
2747    }
```

3.1.18. version.php

```
<?php

defined('MOODLE_INTERNAL') || die();

$module->version   = 2012022100;       // The current module version (Date: YYYYMMDDXX)
$module->requires  = 2011112900;       // Requires this Moodle version
$module->component = 'mod_wikicode';       // Full name of the plugin (used for diagnostics)
$module->cron      = 0;
```

3.1.19. view.php

```
<?php

require_once('../../config.php');
require_once($CFG->dirroot . '/mod/wikicode/lib.php');
require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
require_once($CFG->dirroot . '/mod/wikicode/pagelib.php');

$id = optional_param('id', 0, PARAM_INT); // Course Module ID

$pageid = optional_param('pageid', 0, PARAM_INT); // Page ID

$wid = optional_param('wid', 0, PARAM_INT); // Wiki ID
$title = optional_param('title', '', PARAM_TEXT); // Page Title
$currentgroup = optional_param('group', 0, PARAM_INT); // Group ID
$userid = optional_param('uid', 0, PARAM_INT); // User ID
$groupanduser = optional_param('groupanduser', 0, PARAM_TEXT);

$edit = optional_param('edit', -1, PARAM_BOOL);

$action = optional_param('action', '', PARAM_ALPHA);
$swid = optional_param('swid', 0, PARAM_INT); // Subwiki ID

/*
 * Case 0:
 *
 * User that comes from a course. First wiki page must be shown
 *
 * URL params: id -> course module id
 *
 */
if ($id) {
    // Cheacking course module instance
    if (!$cm = get_coursemodule_from_id('wikicode', $id)) {
        print_error('invalidcoursemodule');
    }

    // Checking course instance
```

```
38      $course = $DB->get_record('course', array('id' => $cm->course), '*', MUST_EXIST
            );
39
40      // Checking wiki instance
41      if (!$wiki = wikicode_get_wiki($cm->instance)) {
42          print_error('incorrectwikiid', 'wikicode');
43      }
44      $PAGE->set_cm($cm);
45
46      // Getting the subwiki corresponding to that wiki, group and user.
47      //
48      // Also setting the page if it exists or getting the first page title form
49      // that wiki
50
51      // Getting current group id
52      $currentgroup = groups_get_activity_group($cm);
53      $currentgroup = !empty($currentgroup) ? $currentgroup : 0;
54      // Getting current user id
55      if ($wiki->wikimode == 'individual') {
56          $userid = $USER->id;
57      } else {
58          $userid = 0;
59      }
60
61      // Getting subwiki. If it does not exists, redirecting to create page
62      if (!$subwiki = wikicode_get_subwiki_by_group($wiki->id, $currentgroup, $userid
            )) {
63          $params = array('wid' => $wiki->id, 'gid' => $currentgroup, 'uid' =>
                $userid, 'title' => $wiki->firstpagetitle);
64          $url = new moodle_url('/mod/wikicode/create.php', $params);
65          redirect($url);
66      }
67
68      // Getting first page. If it does not exists, redirecting to create page
69      if (!$page = wikicode_get_first_page($subwiki->id, $wiki)) {
70          $params = array('swid'=>$subwiki->id, 'title'=>$wiki->firstpagetitle);
71          $url = new moodle_url('/mod/wikicode/create.php', $params);
72          redirect($url);
73      }
74
75      /*
76       * Case 1:
77       *
78       * A user wants to see a page.
79       *
80       * URL Params: pageid -> page id
81       *
82       */
83  } elseif ($pageid) {
84
85      // Checking page instance
86      if (!$page = wikicode_get_page($pageid)) {
87          print_error('incorrectpageid', 'wikicode');
88      }
```

```
89
90      // Checking subwiki
91      if (!$subwiki = wikicode_get_subwiki($page->subwikiid)) {
92          print_error('incorrectsubwikiid', 'wikicode');
93      }
94
95      // Checking wiki instance of that subwiki
96      if (!$wiki = wikicode_get_wiki($subwiki->wikiid)) {
97          print_error('incorrectwikiid', 'wikicode');
98      }
99
100     // Checking course module instance
101     if (!$cm = get_coursemodule_from_instance("wikicode", $subwiki->wikiid)) {
102         print_error('invalidcoursemodule');
103     }
104
105     // Checking course instance
106     $course = $DB->get_record('course', array('id' => $cm->course), '*', MUST_EXIST
            );
107
108     /*
109      * Case 2:
110      *
111      * Trying to read a page from another group or user
112      *
113      * Page can exists or not.
114      *  * If it exists, page must be shown
115      *  * If it does not exists, system must ask for its creation
116      *
117      * URL params: wid -> subwiki id (required)
118      *             title -> a page title (required)
119      *             group -> group id (optional)
120      *             uid -> user id (optional)
121      *             groupanduser -> (optional)
122      */
123 } elseif ($wid && $title) {
124
125     // Setting wiki instance
126     if (!$wiki = wikicode_get_wiki($wid)) {
127         print_error('incorrectwikiid', 'wikicode');
128     }
129
130     // Checking course module
131     if (!$cm = get_coursemodule_from_instance("wikicode", $wiki->id)) {
132         print_error('invalidcoursemodule');
133     }
134
135     // Checking course instance
136     if (!$course = $DB->get_record("course", array("id" => $cm->course))) {
137         print_error('coursemisconf');
138     }
139
140     $groupmode = groups_get_activity_groupmode($cm);
141     if (empty($currentgroup)) {
```

```
142             $currentgroup = groups_get_activity_group($cm);
143             $currentgroup = !empty($currentgroup) ? $currentgroup : 0;
144         }
145
146         if ($wiki->wikimode == 'individual' && ($groupmode == SEPARATEGROUPS ||
                $groupmode == VISIBLEGROUPS)) {
147             list($gid, $uid) = explode('-', $groupanduser);
148         } else if ($wiki->wikimode == 'individual') {
149             $gid = 0;
150             $uid = $userid;
151         } else if ($groupmode == NOGROUPS) {
152             $gid = 0;
153             $uid = 0;
154         } else {
155             $gid = $currentgroup;
156             $uid = 0;
157         }
158
159         // Getting subwiki instance. If it does not exists, redirect to create page
160         if (!$subwiki = wikicode_get_subwiki_by_group($wiki->id, $gid, $uid)) {
161             $context = get_context_instance(CONTEXT_MODULE, $cm->id);
162
163             $modeanduser = $wiki->wikimode == 'individual' && $uid != $USER->id;
164             $modeandgroupmember = $wiki->wikimode == 'collaborative' && !
                    groups_is_member($gid);
165
166             $manage = has_capability('mod/wikicode:managewiki', $context);
167             $edit = has_capability('mod/wikicode:editpage', $context);
168             $manageandedit = $manage && $edit;
169
170             if ($groupmode == VISIBLEGROUPS and ($modeanduser || $modeandgroupmember)
                    and !$manageandedit) {
171                 print_error('nocontent','wikicode');
172             }
173
174             $params = array('wid' => $wiki->id, 'gid' => $gid, 'uid' => $uid, 'title'
                    => $title);
175             $url = new moodle_url('/mod/wikicode/create.php', $params);
176             redirect($url);
177         }
178
179         // Checking is there is a page with this title. If it does not exists, redirect
                to first page
180         if (!$page = wikicode_get_page_by_title($subwiki->id, $title)) {
181             $params = array('wid' => $wiki->id, 'gid' => $gid, 'uid' => $uid, 'title'
                    => $wiki->firstpagetitle);
182             $url = new moodle_url('/mod/wikicode/view.php', $params);
183             redirect($url);
184         }
185
186         //    /*
187         //     * Case 3:
188         //     *
189         //     * A user switches group when is 'reading' a non-existent page.
```

```
190         //     *
191         //     * URL Params: wid -> wiki id
192         //     *             title -> page title
193         //     *             currentgroup -> group id
194         //     *
195         //     */
196         //} elseif ($wid && $title && $currentgroup) {
197         //
198         //    // Checking wiki instance
199         //    if (!$wiki = wikicode_get_wiki($wid)) {
200         //        print_error('incorrectwikiid', 'wiki');
201         //    }
202         //
203         //    // Checking subwiki instance
204         //    // @TODO: Fix call to wikicode_get_subwiki_by_group
205         //    if (!$currentgroup = groups_get_activity_group($cm)){
206         //        $currentgroup = 0;
207         //    }
208         //    if (!$subwiki = wikicode_get_subwiki_by_group($wid, $currentgroup)) {
209         //        print_error('incorrectsubwikiid', 'wiki');
210         //    }
211         //
212         //    // Checking page instance
213         //    if ($page = wikicode_get_page_by_title($subwiki->id, $title)) {
214         //        unset($title);
215         //    }
216         //
217         //    // Checking course instance
218         //    $course = $DB->get_record('course', array('id'=>$cm->course), '*',
                MUST_EXIST);
219         //
220         //    // Checking course module instance
221         //    if (!$cm = get_coursemodule_from_instance("wiki", $wiki->id, $course->id)
                ) {
222         //        print_error('invalidcoursemodule');
223         //    }
224         //
225         //    $subwiki = null;
226         //    $page = null;
227         //
228         //    /*
229         //     * Case 4:
230         //     *
231         //     * Error. No more options
232         //     */
233     } else {
234         print_error('incorrectparameters');
235     }
236     require_login($course, true, $cm);
237
238     $context = get_context_instance(CONTEXT_MODULE, $cm->id);
239     require_capability('mod/wikicode:viewpage', $context);
240
241     add_to_log($course->id, 'wikicode', 'view', 'view.php?id=' . $cm->id, $wiki->id);
```

```
242
243 // Update 'viewed' state if required by completion system
244 require_once($CFG->libdir . '/completionlib.php');
245 $completion = new completion_info($course);
246 $completion->set_module_viewed($cm);
247
248 if (($edit != - 1) and $PAGE->user_allowed_editing()) {
249     $USER->editing = $edit;
250 }
251
252 $wikipage = new page_wikicode_view($wiki, $subwiki, $cm);
253
254 /*The following piece of code is used in order
255  * to perform set_url correctly. It is necessary in order
256  * to make page_wikicode_view class know that this page
257  * has been called via its id.
258  */
259 if ($id) {
260     $wikipage->set_coursemodule($id);
261 }
262
263 $wikipage->set_gid($currentgroup);
264 $wikipage->set_page($page);
265
266 $wikipage->print_header();
267 $wikipage->print_content();
268
269 $wikipage->print_footer();
```

3.1.20. viewversion.php

```
1  <?php
2
3  require_once('../../config.php');
4
5  require_once($CFG->dirroot . '/mod/wikicode/lib.php');
6  require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
7  require_once($CFG->dirroot . '/mod/wikicode/pagelib.php');
8
9  $pageid = required_param('pageid', PARAM_TEXT);
10 $versionid = required_param('versionid', PARAM_INT);
11
12 if (!$page = wikicode_get_page($pageid)) {
13     print_error('incorrectpageid', 'wikicode');
14 }
15
16 if (!$subwiki = wikicode_get_subwiki($page->subwikiid)) {
17     print_error('incorrectsubwikiid', 'wikicode');
18 }
19
20 if (!$wiki = wikicode_get_wiki($subwiki->wikiid)) {
```

```
21      print_error('incorrectwikiid', 'wikicode');
22  }
23
24  if (!$cm = get_coursemodule_from_instance('wikicode', $wiki->id)) {
25      print_error('invalidcoursemodule');
26  }
27
28  $course = $DB->get_record('course', array('id' => $cm->course), '*', MUST_EXIST);
29
30  require_login($course->id, true, $cm);
31  add_to_log($course->id, "wikicode", "history", "history.php?id=$cm->id", "$wiki->id
        ");
32
33  /// Print the page header
34  $wikipage = new page_wikicode_viewversion($wiki, $subwiki, $cm);
35
36  $wikipage->set_page($page);
37  $wikipage->set_versionid($versionid);
38
39  $wikipage->print_header();
40  $wikipage->print_content();
41
42  $wikipage->print_footer();
```

3.2. Back-End

3.2.1. instancecomments.php

```
1   <?php
2
3   require_once('../../config.php');
4   require_once($CFG->dirroot . "/mod/wikicode/pagelib.php");
5   require_once($CFG->dirroot . "/mod/wikicode/locallib.php");
6   require_once($CFG->dirroot . '/mod/wikicode/comments_form.php');
7
8   $pageid = required_param('pageid', PARAM_TEXT);
9   $action = optional_param('action', '', PARAM_ACTION);
10  $id = optional_param('id', 0, PARAM_INT);
11  $commentid = optional_param('commentid', 0, PARAM_INT);
12  $newcontent = optional_param('newcontent', '', PARAM_CLEANHTML);
13  $confirm = optional_param('confirm', 0, PARAM_BOOL);
14
15  if (!$page = wikicode_get_page($pageid)) {
16      print_error('incorrectpageid', 'wikicode');
17  }
18
19  if (!$subwiki = wikicode_get_subwiki($page->subwikiid)) {
20      print_error('incorrectsubwikiid', 'wikicode');
21  }
```

```
22  if (!$cm = get_coursemodule_from_instance("wikicode", $subwiki->wikiid)) {
23      print_error('invalidcoursemodule');
24  }
25  $course = $DB->get_record('course', array('id' => $cm->course), '*', MUST_EXIST);
26  if (!$wiki = wikicode_get_wiki($subwiki->wikiid)) {
27      print_error('incorrectwikiid', 'wikicode');
28  }
29  require_login($course->id, true, $cm);
30
31  if ($action == 'add' || $action == 'edit') {
32      //just check sesskey
33      if (!confirm_sesskey()) {
34          print_error(get_string('invalidsesskey', 'wikicode'));
35      }
36      $comm = new page_wikicode_handlecomments($wiki, $subwiki, $cm);
37      $comm->set_page($page);
38  } else {
39      if(!$confirm) {
40          $comm = new page_wikicode_deletecomment($wiki, $subwiki, $cm);
41          $comm->set_page($page);
42          $comm->set_url();
43      } else {
44          $comm = new page_wikicode_handlecomments($wiki, $subwiki, $cm);
45          $comm->set_page($page);
46          if (!confirm_sesskey()) {
47              print_error(get_string('invalidsesskey', 'wikicode'));
48          }
49      }
50  }
51
52  if ($action == 'delete') {
53      $comm->set_action($action, $commentid, 0);
54  } else {
55      if (empty($newcontent)) {
56          $form = new mod_wikicode_comments_form();
57          $newcomment = $form->get_data();
58          $content = $newcomment->entrycomment_editor['text'];
59      } else {
60          $content = $newcontent;
61      }
62
63      if ($action == 'edit') {
64          $comm->set_action($action, $id, $content);
65
66      } else {
67          $action = 'add';
68          $comm->set_action($action, 0, $content);
69      }
70  }
71
72  $comm->print_header();
73  $comm->print_content();
74  $comm->print_footer();
```

3.2.2. lib.php

```
<?php

/**
 * Given an object containing all the necessary data,
 * (defined by the form in mod.html) this function
 * will create a new instance and return the id number
 * of the new instance.
 *
 * @param object $instance An object from the form in mod.html
 * @return int The id of the newly inserted wiki record
 **/
function wikicode_add_instance($wiki) {
    global $DB;

    $wiki->timemodified = time();
    # May have to add extra stuff in here #
    if (empty($wiki->forceformat)) {
        $wiki->forceformat = 0;
    }
    return $DB->insert_record('wikicode', $wiki);
}

/**
 * Given an object containing all the necessary data,
 * (defined by the form in mod.html) this function
 * will update an existing instance with new data.
 *
 * @param object $instance An object from the form in mod.html
 * @return boolean Success/Fail
 **/
function wikicode_update_instance($wiki) {
    global $DB;

    $wiki->timemodified = time();
    $wiki->id = $wiki->instance;
    if (empty($wiki->forceformat)) {
        $wiki->forceformat = 0;
    }

    # May have to add extra stuff in here #

    return $DB->update_record('wikicode', $wiki);
}

/**
 * Given an ID of an instance of this module,
 * this function will permanently delete the instance
 * and any data that depends on it.
 *
 * @param int $id Id of the module instance
 * @return boolean Success/Failure
```

```
52      **/
53  function wikicode_delete_instance($id) {
54      global $DB;
55
56      if (!$wiki = $DB->get_record('wikicode', array('id' => $id))) {
57          return false;
58      }
59
60      $result = true;
61
62      # Get subwiki information #
63      $subwikis = $DB->get_records('wikicode_subwikis', array('wikiid' => $wiki->id))
            ;
64
65      foreach ($subwikis as $subwiki) {
66          # Get existing links, and delete them #
67          if (!$DB->delete_records('wikicode_links', array('subwikiid' => $subwiki->
                id), IGNORE_MISSING)) {
68              $result = false;
69          }
70
71          # Get existing pages #
72          if ($pages = $DB->get_records('wikicode_pages', array('subwikiid' =>
                $subwiki->id))) {
73              foreach ($pages as $page) {
74                  # Get locks, and delete them #
75                  if (!$DB->delete_records('wikicode_locks', array('pageid' => $page
                        ->id), IGNORE_MISSING)) {
76                      $result = false;
77                  }
78
79                  # Get versions, and delete them #
80                  if (!$DB->delete_records('wikicode_versions', array('pageid' =>
                        $page->id), IGNORE_MISSING)) {
81                      $result = false;
82                  }
83              }
84
85              # Delete pages #
86              if (!$DB->delete_records('wikicode_pages', array('subwikiid' =>
                    $subwiki->id), IGNORE_MISSING)) {
87                  $result = false;
88              }
89          }
90
91          # Get existing synonyms, and delete them #
92          if (!$DB->delete_records('wikicode_synonyms', array('subwikiid' => $subwiki
                ->id), IGNORE_MISSING)) {
93              $result = false;
94          }
95
96          # Delete any subwikis #
97          if (!$DB->delete_records('wikicode_subwikis', array('id' => $subwiki->id),
                IGNORE_MISSING)) {
```

```
98              $result = false;
99          }
100     }
101
102     # Delete any dependent records here #
103     if (!$DB->delete_records('wikicode', array('id' => $wiki->id))) {
104         $result = false;
105     }
106
107     return $result;
108 }
109
110 function wikicode_reset_userdata($data) {
111     global $CFG,$DB;
112     require_once($CFG->dirroot . '/mod/wikicode/pagelib.php');
113     require_once($CFG->dirroot . '/tag/lib.php');
114
115     $componentstr = get_string('modulenameplural', 'wikicode');
116     $status = array();
117
118     //get the wiki(s) in this course.
119     if (!$wikis = $DB->get_records('wikicode', array('course' => $data->courseid)))
            {
120         return false;
121     }
122     $errors = false;
123     foreach ($wikis as $wiki) {
124
125         // remove all comments
126         if (!empty($data->reset_wikicode_comments)) {
127             if (!$cm = get_coursemodule_from_instance('wikicode', $wiki->id)) {
128                 continue;
129             }
130             $context = get_context_instance(CONTEXT_MODULE, $cm->id);
131             $DB->delete_records_select('comments', "contextid = ? AND commentarea='
                    wikicode_page'", array($context->id));
132             $status[] = array('component'=>$componentstr, 'item'=>get_string('
                    deleteallcomments'), 'error'=>false);
133         }
134
135         if (!empty($data->reset_wikicode_tags)) {
136             # Get subwiki information #
137             $subwikis = $DB->get_records('wikicode_subwikis', array('wikiid' =>
                    $wiki->id));
138
139             foreach ($subwikis as $subwiki) {
140                 if ($pages = $DB->get_records('wikicode_pages', array('subwikiid'
                        => $subwiki->id))) {
141                     foreach ($pages as $page) {
142                         $tags = tag_get_tags_array('wikicode_pages', $page->id);
143                         foreach ($tags as $tagid => $tagname) {
144                             // Delete the related tag_instances related to the wiki
                                page.
```

```
145                             $errors = tag_delete_instance('wikicode_pages', $page->
                                    id, $tagid);
146                             $status[] = array('component' => $componentstr, 'item'
                                    => get_string('tagsdeleted', 'wikicode'), 'error'
                                    => $errors);
147                         }
148                     }
149                 }
150             }
151         }
152     }
153     return $status;
154 }
155
156
157 function wikicode_reset_course_form_definition(&$mform) {
158     $mform->addElement('header', 'wikiheader', get_string('modulenameplural', '
            wikicode'));
159     $mform->addElement('advcheckbox', 'reset_wikicode_tags', get_string('
            removeallwikitags', 'wikicode'));
160     $mform->addElement('advcheckbox', 'reset_wikicode_comments', get_string('
            deleteallcomments'));
161 }
162
163 /**
164  * Return a small object with summary information about what a
165  * user has done with a given particular instance of this module
166  * Used for user activity reports.
167  * $return->time = the time they did it
168  * $return->info = a short text description
169  *
170  * @return null
171  * @todo Finish documenting this function
172  **/
173 function wikicode_user_outline($course, $user, $mod, $wiki) {
174     $return = NULL;
175     return $return;
176 }
177
178 /**
179  * Print a detailed representation of what a user has done with
180  * a given particular instance of this module, for user activity reports.
181  *
182  * @return boolean
183  * @todo Finish documenting this function
184  **/
185 function wikicode_user_complete($course, $user, $mod, $wiki) {
186     return true;
187 }
188
189 /**
190  * Indicates API features that the wiki supports.
191  *
192  * @uses FEATURE_GROUPS
```

```
193     * @uses FEATURE_GROUPINGS
194     * @uses FEATURE_GROUPMEMBERSONLY
195     * @uses FEATURE_MOD_INTRO
196     * @uses FEATURE_COMPLETION_TRACKS_VIEWS
197     * @uses FEATURE_COMPLETION_HAS_RULES
198     * @uses FEATURE_GRADE_HAS_GRADE
199     * @uses FEATURE_GRADE_OUTCOMES
200     * @param string $feature
201     * @return mixed True if yes (some features may use other values)
202     */
203    function wikicode_supports($feature) {
204        switch ($feature) {
205        case FEATURE_GROUPS:
206            return true;
207        case FEATURE_GROUPINGS:
208            return true;
209        case FEATURE_GROUPMEMBERSONLY:
210            return true;
211        case FEATURE_MOD_INTRO:
212            return true;
213        case FEATURE_COMPLETION_TRACKS_VIEWS:
214            return true;
215        case FEATURE_GRADE_HAS_GRADE:
216            return false;
217        case FEATURE_GRADE_OUTCOMES:
218            return false;
219        case FEATURE_RATE:
220            return false;
221        case FEATURE_BACKUP_MOODLE2:
222            return true;
223        case FEATURE_SHOW_DESCRIPTION:
224            return true;
225
226        default:
227            return null;
228        }
229    }
230
231    /**
232     * Given a course and a time, this module should find recent activity
233     * that has occurred in wiki activities and print it out.
234     * Return true if there was output, or false is there was none.
235     *
236     * @global $CFG
237     * @global $DB
238     * @uses CONTEXT_MODULE
239     * @uses VISIBLEGROUPS
240     * @param object $course
241     * @param bool $viewfullnames capability
242     * @param int $timestart
243     * @return boolean
244     **/
245    function wikicode_print_recent_activity($course, $viewfullnames, $timestart) {
246        global $CFG, $DB, $OUTPUT;
```

```
247
248     $sql = "SELECT p.*, w.id as wikiid, sw.groupid
249             FROM {wikicode_pages} p
250                 JOIN {wikicode_subwikis} sw ON sw.id = p.subwikiid
251                 JOIN {wikicode} w ON w.id = sw.wikiid
252             WHERE p.timemodified > ? AND w.course = ?
253             ORDER BY p.timemodified ASC";
254     if (!$pages = $DB->get_records_sql($sql, array($timestart, $course->id))) {
255         return false;
256     }
257     $modinfo =& get_fast_modinfo($course);
258
259     $wikis = array();
260
261     $modinfo = get_fast_modinfo($course);
262
263     foreach ($pages as $page) {
264         if (!isset($modinfo->instances['wikicode'][$page->wikiid])) {
265             // not visible
266             continue;
267         }
268         $cm = $modinfo->instances['wikicode'][$page->wikiid];
269         if (!$cm->uservisible) {
270             continue;
271         }
272         $context = get_context_instance(CONTEXT_MODULE, $cm->id);
273
274         if (!has_capability('mod/wikicode:viewpage', $context)) {
275             continue;
276         }
277
278         $groupmode = groups_get_activity_groupmode($cm, $course);
279
280         if ($groupmode) {
281             if ($groupmode == SEPARATEGROUPS and !has_capability('mod/wikicode:
                    managewiki', $context)) {
282                 // separate mode
283                 if (isguestuser()) {
284                     // shortcut
285                     continue;
286                 }
287
288                 if (is_null($modinfo->groups)) {
289                     $modinfo->groups = groups_get_user_groups($course->id); // load
                            all my groups and cache it in modinfo
290                     }
291
292                 if (!in_array($page->groupid, $modinfo->groups[0])) {
293                     continue;
294                 }
295             }
296         }
297         $wikis[] = $page;
298     }
```

```
299         unset($pages);
300 
301         if (!$wikis) {
302             return false;
303         }
304         echo $OUTPUT->heading(get_string("updatedwikipages", 'wikicode') . ':', 3);
305         foreach ($wikis as $wiki) {
306             $cm = $modinfo->instances['wikicode'][$wiki->wikiid];
307             $link = $CFG->wwwroot . '/mod/wikicode/view.php?pageid=' . $wiki->id;
308             print_recent_activity_note($wiki->timemodified, $wiki, $cm->name, $link,
                    false, $viewfullnames);
309         }
310 
311         return true; //  True if anything was printed, otherwise false
312     }
313     /**
314      * Function to be run periodically according to the moodle cron
315      * This function searches for things that need to be done, such
316      * as sending out mail, toggling flags etc ...
317      *
318      * @uses $CFG
319      * @return boolean
320      * @todo Finish documenting this function
321      **/
322     function wikicode_cron() {
323         global $CFG;
324 
325         return true;
326     }
327 
328     /**
329      * Must return an array of grades for a given instance of this module,
330      * indexed by user.  It also returns a maximum allowed grade.
331      *
332      * Example:
333      *    $return->grades = array of grades;
334      *    $return->maxgrade = maximum allowed grade;
335      *
336      *    return $return;
337      *
338      * @param int $wikiid ID of an instance of this module
339      * @return mixed Null or object with an array of grades and with the maximum grade
340      **/
341     function wikicode_grades($wikiid) {
342         return null;
343     }
344 
345     /**
346      * Must return an array of user records (all data) who are participants
347      * for a given instance of wiki. Must include every user involved
348      * in the instance, independient of his role (student, teacher, admin...)
349      * See other modules as example.
350      *
351      * @todo: deprecated - to be deleted in 2.2
```

```
352     *
353     * @param int $wikiid ID of an instance of this module
354     * @return mixed boolean/array of students
355     **/
356    function wikicode_get_participants($wikiid) {
357        return false;
358    }
359
360    /**
361     * This function returns if a scale is being used by one wiki
362     * it it has support for grading and scales. Commented code should be
363     * modified if necessary. See forum, glossary or journal modules
364     * as reference.
365     *
366     * @param int $wikiid ID of an instance of this module
367     * @return mixed
368     * @todo Finish documenting this function
369     **/
370    function wikicode_scale_used($wikiid, $scaleid) {
371        $return = false;
372
373        //$rec = get_record("wiki","id","$wikiid","scale","-$scaleid");
374        //
375        //if (!empty($rec)  && !empty($scaleid)) {
376        //    $return = true;
377        //}
378
379        return $return;
380    }
381
382    /**
383     * Checks if scale is being used by any instance of wiki.
384     * This function was added in 1.9
385     *
386     * This is used to find out if scale used anywhere
387     * @param $scaleid int
388     * @return boolean True if the scale is used by any wiki
389     */
390    function wikicode_scale_used_anywhere($scaleid) {
391        global $DB;
392
393        //if ($scaleid and $DB->record_exists('wikicode', array('grade' => -$scaleid)))
                {
394        //    return true;
395        //} else {
396        //    return false;
397        //}
398
399        return false;
400    }
401
402    /**
403     * Pluginfile hook
404     *
```

```
405     * @author Josep Arus
406     */
407    function wikicode_pluginfile($course, $cm, $context, $filearea, $args,
           $forcedownload) {
408        global $CFG;
409
410        if ($context->contextlevel != CONTEXT_MODULE) {
411            return false;
412        }
413
414        require_login($course, true, $cm);
415
416        require_once($CFG->dirroot . "/mod/wikicode/locallib.php");
417
418        if ($filearea == 'attachments') {
419            $swid = (int) array_shift($args);
420
421            if (!$subwiki = wikicode_get_subwiki($swid)) {
422                return false;
423            }
424
425            require_capability('mod/wikicode:viewpage', $context);
426
427            $relativepath = implode('/', $args);
428
429            $fullpath = "/$context->id/mod_wikicode/attachments/$swid/$relativepath";
430
431            $fs = get_file_storage();
432            if (!$file = $fs->get_file_by_hash(sha1($fullpath)) or $file->is_directory
                   ()) {
433                return false;
434            }
435
436            $lifetime = isset($CFG->filelifetime) ? $CFG->filelifetime : 86400;
437
438            send_stored_file($file, $lifetime, 0);
439        }
440    }
441
442    function wikicode_search_form($cm, $search = '') {
443        global $CFG, $OUTPUT;
444
445        $output = '<div class="wikisearch">';
446        $output .= '<form method="post" action="' . $CFG->wwwroot . '/mod/wikicode/
               search.php" style="display:inline">';
447        $output .= '<fieldset class="invisiblefieldset">';
448        $output .= '<input name="searchstring" type="text" size="18" value="' . s(
               $search, true) . '" alt="search" />';
449        $output .= '<input name="courseid" type="hidden" value="' . $cm->course . '" />
               ';
450        $output .= '<input name="cmid" type="hidden" value="' . $cm->id . '" />';
451        $output .= '<input name="searchwikicontent" type="hidden" value="1" />';
452        $output .= ' <input value="' . get_string('searchwikis', 'wikicode') . '" type
               ="submit" />';
```

```
453         $output .= '</fieldset>';
454         $output .= '</form>';
455         $output .= '</div>';
456 
457         return $output;
458     }
459     function wikicode_extend_navigation(navigation_node $navref, $course, $module, $cm)
         {
460         global $CFG, $PAGE, $USER;
461 
462         require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
463 
464         $context = get_context_instance(CONTEXT_MODULE, $cm->id);
465         $url = $PAGE->url;
466         $userid = 0;
467         if ($module->wikimode == 'individual') {
468             $userid = $USER->id;
469         }
470 
471         if (!$wiki = wikicode_get_wiki($cm->instance)) {
472             return false;
473         }
474 
475         if (!$gid = groups_get_activity_group($cm)) {
476             $gid = 0;
477         }
478         if (!$subwiki = wikicode_get_subwiki_by_group($cm->instance, $gid, $userid)) {
479             return null;
480         } else {
481             $swid = $subwiki->id;
482         }
483 
484         $pageid = $url->param('pageid');
485         $cmid = $url->param('id');
486         if (empty($pageid) && !empty($cmid)) {
487             // wiki main page
488             $page = wikicode_get_page_by_title($swid, $wiki->firstpagetitle);
489             $pageid = $page->id;
490         }
491 
492         if (has_capability('mod/wikicode:createpage', $context)) {
493            //$link = new moodle_url('/mod/wikicode/create.php', array('action' => 'new
                ', 'swid' => $swid));
494            //$node = $navref->add(get_string('newpage', 'wikicode'), $link,
                navigation_node::TYPE_SETTING);
495         }
496 
497         if (is_numeric($pageid)) {
498 
499             if (has_capability('mod/wikicode:viewpage', $context)) {
500                 $link = new moodle_url('/mod/wikicode/view.php', array('pageid' =>
                    $pageid));
501                 $node = $navref->add(get_string('view', 'wikicode'), $link,
                    navigation_node::TYPE_SETTING);
```

```
502             }
503
504             if (has_capability('mod/wikicode:editpage', $context)) {
505                 $link = new moodle_url('/mod/wikicode/edit.php', array('pageid' =>
                        $pageid));
506                 $node = $navref->add(get_string('edit', 'wikicode'), $link,
                        navigation_node::TYPE_SETTING);
507             }
508
509             if (has_capability('mod/wikicode:viewcomment', $context)) {
510                 //$link = new moodle_url('/mod/wikicode/comments.php', array('pageid'
                        => $pageid));
511                 //$node = $navref->add(get_string('comments', 'wikicode'), $link,
                        navigation_node::TYPE_SETTING);
512             }
513
514             if (has_capability('mod/wikicode:viewpage', $context)) {
515                 $link = new moodle_url('/mod/wikicode/history.php', array('pageid' =>
                        $pageid));
516                 $node = $navref->add(get_string('history', 'wikicode'), $link,
                        navigation_node::TYPE_SETTING);
517             }
518
519             if (has_capability('mod/wikicode:viewpage', $context)) {
520                 //$link = new moodle_url('/mod/wikicode/map.php', array('pageid' =>
                        $pageid));
521                 //$node = $navref->add(get_string('map', 'wikicode'), $link,
                        navigation_node::TYPE_SETTING);
522             }
523
524             if (has_capability('mod/wikicode:viewpage', $context)) {
525                 //$link = new moodle_url('/mod/wikicode/files.php', array('pageid' =>
                        $pageid));
526                 //$node = $navref->add(get_string('files', 'wikicode'), $link,
                        navigation_node::TYPE_SETTING);
527             }
528
529                 if (has_capability('mod/wikicode:viewpage', $context)) {
530                 $link = new moodle_url('/mod/wikicode/log.php', array('pageid' =>
                        $pageid));
531                 $node = $navref->add(get_string('log', 'wikicode'), $link,
                        navigation_node::TYPE_SETTING);
532             }
533
534             if (has_capability('mod/wikicode:managewiki', $context)) {
535                 $link = new moodle_url('/mod/wikicode/admin.php', array('pageid' =>
                        $pageid));
536                 $node = $navref->add(get_string('admin', 'wikicode'), $link,
                        navigation_node::TYPE_SETTING);
537             }
538         }
539     }
540     /**
541      * Returns all other caps used in wiki module
```

```
542     *
543     * @return array
544     */
545    function wikicode_get_extra_capabilities() {
546        return array('moodle/comment:view', 'moodle/comment:post', 'moodle/comment:
               delete');
547    }
548
549    /**
550     * Running addtional permission check on plugin, for example, plugins
551     * may have switch to turn on/off comments option, this callback will
552     * affect UI display, not like pluginname_comment_validate only throw
553     * exceptions.
554     * Capability check has been done in comment->check_permissions(), we
555     * don't need to do it again here.
556     *
557     * @param stdClass $comment_param {
558     *              context  => context the context object
559     *              courseid => int course id
560     *              cm       => stdClass course module object
561     *              commentarea => string comment area
562     *              itemid      => int itemid
563     * }
564     * @return array
565     */
566    function wikicode_comment_permissions($comment_param) {
567        return array('post'=>true, 'view'=>true);
568    }
569
570    /**
571     * Validate comment parameter before perform other comments actions
572     *
573     * @param stdClass $comment_param {
574     *              context  => context the context object
575     *              courseid => int course id
576     *              cm       => stdClass course module object
577     *              commentarea => string comment area
578     *              itemid      => int itemid
579     * }
580     * @return boolean
581     */
582    function wikicode_comment_validate($comment_param) {
583        global $DB, $CFG;
584        require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
585        // validate comment area
586        if ($comment_param->commentarea != 'wikicode_page') {
587            throw new comment_exception('invalidcommentarea');
588        }
589        // validate itemid
590        if (!$record = $DB->get_record('wikicode_pages', array('id'=>$comment_param->
               itemid'))) {
591            throw new comment_exception('invalidcommentitemid');
592        }
593        if (!$subwiki = wikicode_get_subwiki($record->subwikiid)) {
```

```
594             throw new comment_exception('invalidsubwikiid');
595         }
596         if (!$wiki = wikicode_get_wikicode_from_pageid($comment_param->itemid)) {
597             throw new comment_exception('invalidid', 'data');
598         }
599         if (!$course = $DB->get_record('course', array('id'=>$wiki->course))) {
600             throw new comment_exception('coursemisconf');
601         }
602         if (!$cm = get_coursemodule_from_instance('wikicode', $wiki->id, $course->id))
                {
603             throw new comment_exception('invalidcoursemodule');
604         }
605         $context = get_context_instance(CONTEXT_MODULE, $cm->id);
606         // group access
607         if ($subwiki->groupid) {
608             $groupmode = groups_get_activity_groupmode($cm, $course);
609             if ($groupmode == SEPARATEGROUPS and !has_capability('moodle/site:
                    accessallgroups', $context)) {
610                 if (!groups_is_member($subwiki->groupid)) {
611                     throw new comment_exception('notmemberofgroup');
612                 }
613             }
614         }
615         // validate context id
616         if ($context->id != $comment_param->context->id) {
617             throw new comment_exception('invalidcontext');
618         }
619         // validation for comment deletion
620         if (!empty($comment_param->commentid)) {
621             if ($comment = $DB->get_record('comments', array('id'=>$comment_param->
                    commentid))) {
622                 if ($comment->commentarea != 'wikicode_page') {
623                     throw new comment_exception('invalidcommentarea');
624                 }
625                 if ($comment->contextid != $context->id) {
626                     throw new comment_exception('invalidcontext');
627                 }
628                 if ($comment->itemid != $comment_param->itemid) {
629                     throw new comment_exception('invalidcommentitemid');
630                 }
631             } else {
632                 throw new comment_exception('invalidcommentid');
633             }
634         }
635         return true;
636     }
637
638     /**
639      * Return a list of page types
640      * @param string $pagetype current page type
641      * @param stdClass $parentcontext Block's parent context
642      * @param stdClass $currentcontext Current context of block
643      */
644     function wikicode_page_type_list($pagetype, $parentcontext, $currentcontext) {
```

```
645     $module_pagetype = array(
646         'mod-wiki-*'=>get_string('page-mod-wiki-x', 'wikicode'),
647         'mod-wiki-view'=>get_string('page-mod-wiki-view', 'wikicode'),
648         'mod-wiki-comments'=>get_string('page-mod-wiki-comments', 'wikicode'),
649         'mod-wiki-history'=>get_string('page-mod-wiki-history', 'wikicode'),
650         'mod-wiki-map'=>get_string('page-mod-wiki-map', 'wikicode')
651     );
652     return $module_pagetype;
653 }
```

3.2.3. localib.php

```
1  <?php
2
3  require_once($CFG->dirroot . '/mod/wikicode/lib.php');
4  require_once($CFG->dirroot . '/mod/wikicode/parser/parser.php');
5  require_once($CFG->libdir . '/filelib.php');
6
7  define('WIKI_REFRESH_CACHE_TIME', 30); // @TODO: To be deleted.
8  define('FORMAT_CREOLE', '37');
9  define('FORMAT_NWIKI', '38');
10 define('NO_VALID_RATE', '-999');
11 define('IMPROVEMENT', '+');
12 define('EQUAL', '=');
13 define('WORST', '-');
14
15 define('LOCK_TIMEOUT', 30);
16
17 /**
18  * Get a wiki instance
19  * @param int $wikiid the instance id of wiki
20  */
21 function wikicode_get_wiki($wikiid) {
22     global $DB;
23
24     return $DB->get_record('wikicode', array('id' => $wikiid));
25 }
26
27 /**
28  * Get sub wiki instances with same wiki id
29  * @param int $wikiid
30  */
31 function wikicode_get_subwikis($wikiid) {
32     global $DB;
33     return $DB->get_records('wikicode_subwikis', array('wikiid' => $wikiid));
34 }
35
36 /**
37  * Get a sub wiki instance by wiki id and group id
38  * @param int $wikiid
39  * @param int $groupid
```

```
40      * @return object
41      */
42     function wikicode_get_subwiki_by_group($wikiid, $groupid, $userid = 0) {
43         global $DB;
44         return $DB->get_record('wikicode_subwikis', array('wikiid' => $wikiid, 'groupid
               ' => $groupid, 'userid' => $userid));
45     }
46
47     /**
48      * Get a sub wiki instace by instance id
49      * @param int $subwikiid
50      * @return object
51      */
52     function wikicode_get_subwiki($subwikiid) {
53         global $DB;
54         return $DB->get_record('wikicode_subwikis', array('id' => $subwikiid));
55
56     }
57
58     /**
59      * Add a new sub wiki instance
60      * @param int $wikiid
61      * @param int $groupid
62      * @return int $insertid
63      */
64     function wikicode_add_subwiki($wikiid, $groupid, $userid = 0) {
65         global $DB;
66
67         $record = new StdClass();
68         $record->wikiid = $wikiid;
69         $record->groupid = $groupid;
70         $record->userid = $userid;
71
72         $insertid = $DB->insert_record('wikicode_subwikis', $record);
73         return $insertid;
74     }
75
76     /**
77      * Get a wiki instance by pageid
78      * @param int $pageid
79      * @return object
80      */
81     function wikicode_get_wikicode_from_pageid($pageid) {
82         global $DB;
83
84         $sql = "SELECT w.*
85                 FROM {wikicode} w, {wikicode_subwikis} s, {wikicode_pages} p
86                 WHERE p.id = ? AND
87                 p.subwikiid = s.id AND
88                 s.wikiid = w.id";
89
90         return $DB->get_record_sql($sql, array($pageid));
91     }
92
```

```
93  /**
94   * Get gcc path most appearances
95   * @return object
96   */
97  function wikicode_get_gccpath() {
98      global $DB;
99
100     $sql = "SELECT w.gccpath
101             FROM {wikicode} w
102             GROUP BY w.gccpath ORDER BY w.gccpath DESC
103             LIMIT 1";
104
105     return $DB->get_record_sql($sql);
106 }
107
108 /**
109  * Get mingw path most appearances
110  * @return object
111  */
112 function wikicode_get_mingwpath() {
113     global $DB;
114
115     $sql = "SELECT w.mingwpath
116             FROM {wikicode} w
117             GROUP BY w.mingwpath ORDER BY w.mingwpath DESC
118             LIMIT 1";
119
120     return $DB->get_record_sql($sql);
121 }
122
123 /**
124  * Get a wiki page by pageid
125  * @param int $pageid
126  * @return object
127  */
128 function wikicode_get_page($pageid) {
129     global $DB;
130     return $DB->get_record('wikicode_pages', array('id' => $pageid));
131 }
132
133 /**
134  * Get latest version of wiki page
135  * @param int $pageid
136  * @return object
137  */
138 function wikicode_get_current_version($pageid) {
139     global $DB;
140
141     // @TODO: Fix this query
142     $sql = "SELECT *
143             FROM {wikicode_versions}
144             WHERE pageid = ?
145             ORDER BY version DESC";
146     return array_pop($DB->get_records_sql($sql, array($pageid), 0, 1));
```

```
147
148 }
149
150 /**
151  * Alias of wikicode_get_current_version
152  * @TODO, does the exactly same thing as wikicode_get_current_version, should be
        removed
153  * @param int $pageid
154  * @return object
155  */
156 function wikicode_get_last_version($pageid) {
157     return wikicode_get_current_version($pageid);
158 }
159
160 /**
161  * Get page section
162  * @param int $pageid
163  * @param string $section
164  */
165 function wikicode_get_section_page($page, $section) {
166
167     $version = wikicode_get_current_version($page->id);
168     return wikicode_parser_proxy::get_section($version->content, $version->
          contentformat, $section);
169 }
170
171 /**
172  * Get a wiki page by page title
173  * @param int $swid, sub wiki id
174  * @param string $title
175  * @return object
176  */
177 function wikicode_get_page_by_title($swid, $title) {
178     global $DB;
179     return $DB->get_record('wikicode_pages', array('subwikiid' => $swid, 'title' =>
           $title));
180 }
181
182 /**
183  * Get a version record by record id
184  * @param int $versionid, the version id
185  * @return object
186  */
187 function wikicode_get_version($versionid) {
188     global $DB;
189     return $DB->get_record('wikicode_versions', array('id' => $versionid));
190 }
191
192 /**
193  * Get first page of wiki instace
194  * @param int $subwikiid
195  * @param int $module, wiki instance object
196  */
197 function wikicode_get_first_page($subwikid, $module = null) {
```

```
198     global $DB, $USER;
199
200     $sql = "SELECT p.*
201             FROM {wikicode} w, {wikicode_subwikis} s, {wikicode_pages} p
202             WHERE s.id = ? AND
203             s.wikiid = w.id AND
204             w.firstpagetitle = p.title AND
205             p.subwikiid = s.id";
206     return $DB->get_record_sql($sql, array($subwikid));
207 }
208
209 function wikicode_save_section($wikipage, $sectiontitle, $sectioncontent, $userid)
        {
210
211     $wiki = wikicode_get_wikicode_from_pageid($wikipage->id);
212     $cm = get_coursemodule_from_instance('wikicode', $wiki->id);
213     $context = get_context_instance(CONTEXT_MODULE, $cm->id);
214
215     if (has_capability('mod/wikicode:editpage', $context)) {
216         $version = wikicode_get_current_version($wikipage->id);
217         $content = wikicode_parser_proxy::get_section($version->content, $version->
                contentformat, $sectiontitle, true);
218
219         $newcontent = $content[0] . $sectioncontent . $content[2];
220
221         return wikicode_save_page($wikipage, $newcontent, $userid);
222     } else {
223         return false;
224     }
225 }
226
227 /**
228  * Save page content
229  * @param object $wikipage
230  * @param string $newcontent
231  * @param int $userid
232  */
233 function wikicode_save_page($wikipage, $newcontent, $userid) {
234     global $DB;
235
236     $wiki = wikicode_get_wikicode_from_pageid($wikipage->id);
237     $cm = get_coursemodule_from_instance('wikicode', $wiki->id);
238     $context = get_context_instance(CONTEXT_MODULE, $cm->id);
239
240     if (has_capability('mod/wikicode:editpage', $context)) {
241         $version = wikicode_get_current_version($wikipage->id);
242
243         $version->content = $newcontent;
244         $version->userid = $userid;
245         $version->version++;
246         $version->timecreated = time();
247         $versionid = $DB->insert_record('wikicode_versions', $version);
248
249         $wikipage->timemodified = $version->timecreated;
```

```
250             $wikipage->userid = $userid;
251             $return = wikicode_refresh_cachedcontent($wikipage, $newcontent);
252 
253             return $return;
254         } else {
255             return false;
256         }
257     }
258 
259     /**
260      * Compile page content
261      * @param object $wikipage
262      * @param string $newcontent
263      * @param int $userid
264      */
265     function wikicode_compile_page($wikipage, $newcontent, $userid, $download) {
266         global $DB;
267 
268         $wiki = wikicode_get_wikicode_from_pageid($wikipage->id);
269         $cm = get_coursemodule_from_instance('wikicode', $wiki->id);
270         $context = get_context_instance(CONTEXT_MODULE, $cm->id);
271 
272         if (has_capability('mod/wikicode:editpage', $context)) {
273 
274                     //Creamos el fichero
275                     $fileC = fopen("code.c","w");
276                     $codigo = wikicode_remove_tags($newcontent);
277                     fputs($fileC,$codigo);
278                     fclose($fileC);
279 
280                     //Detectamos el OS
281                     $userAgent = $_SERVER[HTTP_USER_AGENT];
282                     $userAgent = strtolower ($userAgent);
283 
284                     //Llamamos al compilador correspondiente
285                     if(strpos($userAgent, "windows") !== false) {
286                     $output=system($wiki->mingwpath . " code.c -o moodle.exe -w 2>
                            output.txt");
287                             $ejecutable='moodle.exe';
288                     }
289                     else {
290                             $output=system($wiki->gccpath . " code.c -o moodle.out -w
                                    2> output.txt");
291                             $ejecutable='moodle.out';
292                     }
293 
294                     $file_output = fopen ("output.txt", "r");
295                     $output = fread($file_output, filesize("output.txt"));
296                     $output = utf8_encode($output);
297 
298                     if ( $output == '0' || $output == '') {
299                             $output = 'Compilacion correcta';
300                     } else {
301                             $wikipage->errorcompile++;
```

```
302                 }
303
304         //Guardamos los datos
305         $wikipage->cachedcompile = $newcontent;
306                 $wikipage->cachedgcc = $output;
307                 $DB->update_record('wikicode_pages', $wikipage);
308
309                 //Descargamos el ejecutable
310                 if ($output == 'Compilacion correcta' and $download == TRUE) {
311                         echo "<script language='JavaScript'>window.open('".
                            $ejecutable."')</script>";
312                 }
313
314         return array('page' => $wikipage);
315     } else {
316         return false;
317     }
318 }
319
320 function wikicode_remove_tags($newcontent) {
321
322         $codigo = str_replace("</!>", "", $newcontent);
323
324         $desde = strpos($codigo,"<!");
325
326         while (is_numeric($desde)) {
327                 $hasta = strpos($codigo, ">", $desde);
328                 $codigo = substr_replace($codigo, '', $desde, $hasta - $desde + 1);
329
330                 $desde = strpos($codigo,"<!");
331         }
332
333         return $codigo;
334 }
335
336 function wikicode_remove_tags_owner($codigo) {
337
338         global $USER;
339
340         $desde = strpos($codigo,"<!".$USER->username.">");
341
342         while (is_numeric($desde)) {
343                 $codigoaux = substr($codigo, $desde);
344                 $desdeaux = strpos($codigoaux, "</!>");
345
346                 $codigo = substr_replace($codigo, '', $desde + $desdeaux, 4);
347                 $codigo = substr_replace($codigo, '', $desde, strlen("<!".$USER->
                    username.">"));
348
349                 $desde = strpos($codigo,"<!".$USER->username.">");
350         }
351
352         return $codigo;
353 }
```

```
354
355 function wikicode_refresh_cachedcontent($page, $newcontent = null) {
356     global $DB;
357
358     $version = wikicode_get_current_version($page->id);
359     if (empty($version)) {
360         return null;
361     }
362     if (!isset($newcontent)) {
363         $newcontent = $version->content;
364     }
365
366     $options = array('swid' => $page->subwikiid, 'pageid' => $page->id);
367     $parseroutput = wikicode_parse_content($version->contentformat, $newcontent,
        $options);
368         if ($veresion->contentformat = 'nwiki') {
369                 $page->cachedcontent = $version->content;
370         }
371         else {
372                 $page->cachedcontent = $parseroutput['toc'] . $parseroutput['
                    parsed_text'];
373         }
374
375     $page->timerendered = time();
376     $DB->update_record('wikicode_pages', $page);
377
378     wikicode_refresh_page_links($page, $parseroutput['link_count']);
379
380     return array('page' => $page, 'sections' => $parseroutput['repeated_sections'],
         'version' => $version->version);
381 }
382 /**
383  * Restore a page
384  */
385 function wikicode_restore_page($wikipage, $newcontent, $userid) {
386     $return = wikicode_save_page($wikipage, $newcontent, $userid);
387     return $return['page'];
388 }
389
390 function wikicode_refresh_page_links($page, $links) {
391     global $DB;
392
393     $DB->delete_records('wikicode_links', array('frompageid' => $page->id));
394     foreach ($links as $linkname => $linkinfo) {
395
396         $newlink = new stdClass();
397         $newlink->subwikiid = $page->subwikiid;
398         $newlink->frompageid = $page->id;
399
400         if ($linkinfo['new']) {
401             $newlink->tomissingpage = $linkname;
402         } else {
403             $newlink->topageid = $linkinfo['pageid'];
404         }
```

```
405
406         try {
407             $DB->insert_record('wikicode_links', $newlink);
408         } catch (dml_exception $e) {
409             debugging($e->getMessage());
410         }
411
412     }
413 }
414
415 /**
416  * Create a new wiki page, if the page exists, return existing pageid
417  * @param int $swid
418  * @param string $title
419  * @param string $format
420  * @param int $userid
421  */
422 function wikicode_create_page($swid, $title, $format, $userid) {
423     global $DB, $PAGE;
424     $subwiki = wikicode_get_subwiki($swid);
425     $cm = get_coursemodule_from_instance('wikicode', $subwiki->wikiid);
426     $context = get_context_instance(CONTEXT_MODULE, $cm->id);
427     require_capability('mod/wikicode:editpage', $context);
428     // if page exists
429     if ($page = wikicode_get_page_by_title($swid, $title)) {
430         return $page->id;
431     }
432
433     // Creating a new empty version
434     $version = new stdClass();
435     $version->content = '';
436     $version->contentformat = $format;
437     $version->version = 0;
438     $version->timecreated = time();
439     $version->userid = $userid;
440
441     $versionid = null;
442     $versionid = $DB->insert_record('wikicode_versions', $version);
443
444     // Createing a new empty page
445     $page = new stdClass();
446     $page->subwikiid = $swid;
447     $page->title = $title;
448     $page->cachedcontent = '';
449     $page->timecreated = $version->timecreated;
450     $page->timemodified = $version->timecreated;
451     $page->timerendered = $version->timecreated;
452     $page->userid = $userid;
453     $page->pageviews = 0;
454     $page->readonly = 0;
455         $page->cachedcompile = '';
456         $page->cachedgcc = '';
457         $page->errorcompile = 0;
458         $page->timestartedit = 0;
```

```
459             $page->timeendedit = 0;
460
461         $pageid = $DB->insert_record('wikicode_pages', $page);
462
463         // Setting the pageid
464         $version->id = $versionid;
465         $version->pageid = $pageid;
466         $DB->update_record('wikicode_versions', $version);
467
468         wikicode_make_cache_expire($page->title);
469         return $pageid;
470     }
471
472     function wikicode_make_cache_expire($pagename) {
473         global $DB;
474
475         $sql = "UPDATE {wikicode_pages}
476                 SET timerendered = 0
477                 WHERE id IN ( SELECT l.frompageid
478                     FROM {wikicode_links} l
479                     WHERE l.tomissingpage = ?
480                 )";
481         $DB->execute ($sql, array($pagename));
482     }
483
484     /**
485      * Get a specific version of page
486      * @param int $pageid
487      * @param int $version
488      */
489     function wikicode_get_wikicode_page_version($pageid, $version) {
490         global $DB;
491         return $DB->get_record('wikicode_versions', array('pageid' => $pageid, 'version
                ' => $version));
492     }
493
494     /**
495      * Get version list
496      * @param int $pageid
497      * @param int $limitfrom
498      * @param int $limitnum
499      */
500     function wikicode_get_wikicode_page_versions($pageid, $limitfrom, $limitnum) {
501         global $DB;
502         return $DB->get_records('wikicode_versions', array('pageid' => $pageid), '
                version DESC', '*', $limitfrom, $limitnum);
503     }
504
505     /**
506      * Count the number of page version
507      * @param int $pageid
508      */
509     function wikicode_count_wikicode_page_versions($pageid) {
510         global $DB;
```

```
511         return $DB->count_records('wikicode_versions', array('pageid' => $pageid));
512     }
513
514     /**
515      * Get linked from page
516      * @param int $pageid
517      */
518     function wikicode_get_linked_to_pages($pageid) {
519         global $DB;
520         return $DB->get_records('wikicode_links', array('frompageid' => $pageid));
521     }
522
523     /**
524      * Get linked from page
525      * @param int $pageid
526      */
527     function wikicode_get_linked_from_pages($pageid) {
528         global $DB;
529         return $DB->get_records('wikicode_links', array('topageid' => $pageid));
530     }
531
532     /**
533      * Get pages which user have been edited
534      * @param int $swid
535      * @param int $userid
536      */
537     function wikicode_get_contributions($swid, $userid) {
538         global $DB;
539
540         $sql = "SELECT v.*
541                 FROM {wikicode_versions} v, {wikicode_pages} p
542                 WHERE p.subwikiid = ? AND
543                 v.pageid = p.id AND
544                 v.userid = ?";
545
546         return $DB->get_records_sql($sql, array($swid, $userid));
547     }
548
549     /**
550      * Get missing or empty pages in wiki
551      * @param int $swid sub wiki id
552      */
553     function wikicode_get_missing_or_empty_pages($swid) {
554         global $DB;
555
556         $sql = "SELECT DISTINCT p.title, p.id, p.subwikiid
557                 FROM {wikicode} w, {wikicode_subwikis} s, {wikicode_pages} p
558                 WHERE s.wikiid = w.id and
559                 s.id = ? and
560                 w.firstpagetitle != p.title and
561                 p.subwikiid = ? and
562                 1 =  (SELECT count(*)
563                     FROM {wikicode_versions} v
564                     WHERE v.pageid = p.id)
```

```
565             UNION
566             SELECT DISTINCT l.tomissingpage as title, 0 as id, l.subwikiid
567             FROM {wikicode_links} l
568             WHERE l.subwikiid = ? and
569             l.topageid = 0";
570
571     return $DB->get_records_sql($sql, array($swid, $swid, $swid));
572 }
573
574 /**
575  * Get pages list in wiki
576  * @param int $swid sub wiki id
577  */
578 function wikicode_get_page_list($swid) {
579     global $DB;
580     $records = $DB->get_records('wikicode_pages', array('subwikiid' => $swid), '
        title ASC');
581     return $records;
582 }
583
584 /**
585  * Return a list of orphaned wikis for one specific subwiki
586  * @global object
587  * @param int $swid sub wiki id
588  */
589 function wikicode_get_orphaned_pages($swid) {
590     global $DB;
591
592     $sql = "SELECT p.id, p.title
593             FROM {wikicode_pages} p, {wikicode} w , {wikicode_subwikis} s
594             WHERE p.subwikiid = ?
595             AND s.id = ?
596             AND w.id = s.wikiid
597             AND p.title != w.firstpagetitle
598             AND p.id NOT IN (SELECT topageid FROM {wikicode_links} WHERE subwikiid
                = ?);";
599
600     return $DB->get_records_sql($sql, array($swid, $swid, $swid));
601 }
602
603 /**
604  * Search wiki title
605  * @param int $swid sub wiki id
606  * @param string $search
607  */
608 function wikicode_search_title($swid, $search) {
609     global $DB;
610
611     return $DB->get_records_select('wikicode_pages', "subwikiid = ? AND title LIKE
        ?", array($swid, '%'.$search.'%'));
612 }
613
614 /**
615  * Search wiki content
```

```
616     * @param int $swid sub wiki id
617     * @param string $search
618     */
619    function wikicode_search_content($swid, $search) {
620        global $DB;
621
622        return $DB->get_records_select('wikicode_pages', "subwikiid = ? AND
               cachedcontent LIKE ?", array($swid, '%'.$search.'%'));
623    }
624
625    /**
626     * Search wiki title and content
627     * @param int $swid sub wiki id
628     * @param string $search
629     */
630    function wikicode_search_all($swid, $search) {
631        global $DB;
632
633        return $DB->get_records_select('wikicode_pages', "subwikiid = ? AND (
               cachedcontent LIKE ? OR title LIKE ?)", array($swid, '%'.$search.'%', '%'.
               $search.'%'));
634    }
635
636    /**
637     * Get user data
638     */
639    function wikicode_get_user_info($userid) {
640        global $DB;
641        return $DB->get_record('user', array('id' => $userid));
642    }
643
644    /**
645     * Increase page view nubmer
646     * @param int $page, database record
647     */
648    function wikicode_increment_pageviews($page) {
649        global $DB;
650
651        $page->pageviews++;
652        $DB->update_record('wikicode_pages', $page);
653    }
654
655    //-------------------------------------------------------------
656    //-------------------------------------------------------------
657
658    /**
659     * Text format supported by wiki module
660     */
661    function wikicode_get_formats() {
662        return array('wcode');
663    }
664
665    /**
666     * Parses a string with the wiki markup language in $markup.
```

```
667     *
668     * @return Array or false when something wrong has happened.
669     *
670     * Returned array contains the following fields:
671     *      'parsed_text' => String. Contains the parsed wiki content.
672     *      'unparsed_text' => String. Constains the original wiki content.
673     *      'link_count' => Array of array('destination' => ..., 'new' => "is new?").
            Contains the internal wiki links found in the wiki content.
674     *       'deleted_sections' => the list of deleted sections.
675     *              '' =>
676     *
677     * @author Josep Arus Pous
678     **/
679    function wikicode_parse_content($markup, $pagecontent, $options = array()) {
680        global $PAGE;
681
682        $subwiki = wikicode_get_subwiki($options['swid']);
683        $cm = get_coursemodule_from_instance("wikicode", $subwiki->wikiid);
684        $context = get_context_instance(CONTEXT_MODULE, $cm->id);
685
686        $parser_options = array(
687            'link_callback' => '/mod/wikicode/locallib.php:wiki_parser_link',
688            'link_callback_args' => array('swid' => $options['swid']),
689            'table_callback' => '/mod/wikicode/locallib.php:wiki_parser_table',
690            'real_path_callback' => '/mod/wikicode/locallib.php:wiki_parser_real_path',
691            'real_path_callback_args' => array(
692                'context' => $context,
693                'component' => 'mod_wikicode',
694                'filearea' => 'attachments',
695                'subwikiid'=> $subwiki->id,
696                'pageid' => $options['pageid']
697            ),
698            'pageid' => $options['pageid'],
699            'pretty_print' => (isset($options['pretty_print']) && $options['
                pretty_print']),
700            'printable' => (isset($options['printable']) && $options['printable'])
701        );
702
703        return wikicode_parser_proxy::parse($pagecontent, $markup, $parser_options);
704    }
705
706    /**
707     * This function is the parser callback to parse wiki links.
708     *
709     * It returns the necesary information to print a link.
710     *
711     * NOTE: Empty pages and non-existent pages must be print in red color.
712     *
713     * @param link name of a page
714     * @param $options
715     *
716     * @return
717     *
718     * @TODO Doc return and options
```

```
719     */
720    function wikicode_parser_link($link, $options = null) {
721        global $CFG;
722
723        if (is_object($link)) {
724            $parsedlink = array('content' => $link->title, 'url' => $CFG->wwwroot . '/
                   mod/wikicode/view.php?pageid=' . $link->id, 'new' => false, 'link_info'
                    => array('link' => $link->title, 'pageid' => $link->id, 'new' => false
                   ));
725
726            $version = wikicode_get_current_version($link->id);
727            if ($version->version == 0) {
728                $parsedlink['new'] = true;
729            }
730            return $parsedlink;
731        } else {
732            $swid = $options['swid'];
733
734            if ($page = wikicode_get_page_by_title($swid, $link)) {
735                $parsedlink = array('content' => $link, 'url' => $CFG->wwwroot . '/mod/
                       wikicode/view.php?pageid=' . $page->id, 'new' => false, 'link_info'
                        => array('link' => $link, 'pageid' => $page->id, 'new' => false));
736
737                $version = wikicode_get_current_version($page->id);
738                if ($version->version == 0) {
739                    $parsedlink['new'] = true;
740                }
741
742                return $parsedlink;
743
744            } else {
745                return array('content' => $link, 'url' => $CFG->wwwroot . '/mod/
                       wikicode/create.php?swid=' . $swid . '&amp;title=' . urlencode(
                       $link) . '&amp;action=new', 'new' => true, 'link_info' => array('
                       link' => $link, 'new' => true, 'pageid' => 0));
746            }
747        }
748    }
749
750    /**
751     * Returns the table fully parsed (HTML)
752     *
753     * @return HTML for the table $table
754     * @author Josep Arus Pous
755     *
756     **/
757    function wikicode_parser_table($table) {
758        global $OUTPUT;
759
760        $htmltable = new html_table();
761
762        $headers = $table[0];
763        $htmltable->head = array();
764        foreach ($headers as $h) {
```

```
765             $htmltable->head[] = $h[1];
766         }
767 
768         array_shift($table);
769         $htmltable->data = array();
770         foreach ($table as $row) {
771             $row_data = array();
772             foreach ($row as $r) {
773                 $row_data[] = $r[1];
774             }
775             $htmltable->data[] = $row_data;
776         }
777 
778         return html_writer::table($htmltable);
779     }
780 
781     /**
782      * Returns an absolute path link, unless there is no such link.
783      *
784      * @param string $url Link's URL or filename
785      * @param stdClass $context filearea params
786      * @param string $component The component the file is associated with
787      * @param string $filearea The filearea the file is stored in
788      * @param int $swid Sub wiki id
789      *
790      * @return string URL for files full path
791      */
792 
793     function wikicode_parser_real_path($url, $context, $component, $filearea, $swid) {
794         global $CFG;
795 
796         if (preg_match("/^(?:http|ftp)s?\:\/\//", $url)) {
797             return $url;
798         } else {
799 
800             $file = 'pluginfile.php';
801             if (!$CFG->slasharguments) {
802                 $file = $file . '?file=';
803             }
804             $baseurl = "$CFG->wwwroot/$file/{$context->id}/$component/$filearea/$swid/"
                    ;
805             // it is a file in current file area
806             return $baseurl . $url;
807         }
808     }
809 
810     /**
811      * Returns the token used by a wiki language to represent a given tag or "object" (
            bold -> **)
812      *
813      * @return A string when it has only one token at the beginning (f. ex. lists). An
            array composed by 2 strings when it has 2 tokens, one at the beginning and one
             at the end (f. ex. italics). Returns false otherwise.
814      * @author Josep Arus Pous
```

```
815  **/
816 function wikicode_parser_get_token($markup, $name) {
817
818     return wikicode_parser_proxy::get_token($name, $markup);
819 }
820
821 /**
822  * Checks if current user can view a subwiki
823  *
824  * @param $subwiki
825  */
826 function wikicode_user_can_view($subwiki) {
827     global $USER;
828
829     $wiki = wikicode_get_wiki($subwiki->wikiid);
830     $cm = get_coursemodule_from_instance('wikicode', $wiki->id);
831     $context = get_context_instance(CONTEXT_MODULE, $cm->id);
832
833     // Working depending on activity groupmode
834     switch (groups_get_activity_groupmode($cm)) {
835     case NOGROUPS:
836
837         if ($wiki->wikimode == 'collaborative') {
838             // Collaborative Mode:
839             // There is one wiki for all the class.
840             //
841             // Only view capbility needed
842             return has_capability('mod/wikicode:viewpage', $context);
843         } else if ($wiki->wikimode == 'individual') {
844             // Individual Mode:
845             // Each person owns a wiki.
846             if ($subwiki->userid == $USER->id) {
847                 // Only the owner of the wiki can view it
848                 return has_capability('mod/wikicode:viewpage', $context);
849             } else { // User has special capabilities
850                 // User must have:
851                 //      mod/wiki:viewpage capability
852                 // and
853                 //      mod/wiki:managewiki capability
854                 $view = has_capability('mod/wikicode:viewpage', $context);
855                 $manage = has_capability('mod/wikicode:managewiki', $context);
856
857                 return $view && $manage;
858             }
859         } else {
860             //Error
861             return false;
862         }
863     case SEPARATEGROUPS:
864         // Collaborative and Individual Mode
865         //
866         // Collaborative Mode:
867         //      There is one wiki per group.
868         // Individual Mode:
```

```
869             //      Each person owns a wiki.
870             if ($wiki->wikimode == 'collaborative' || $wiki->wikimode == 'individual')
                    {
871                 // Only members of subwiki group could view that wiki
872                 if ($subwiki->groupid == groups_get_activity_group($cm)) {
873                     // Only view capability needed
874                     return has_capability('mod/wikicode:viewpage', $context);
875 
876                 } else { // User is not part of that group
877                     // User must have:
878                     //      mod/wiki:managewiki capability
879                     // or
880                     //      moodle/site:accessallgroups capability
881                     // and
882                     //      mod/wiki:viewpage capability
883                     $view = has_capability('mod/wikicode:viewpage', $context);
884                     $manage = has_capability('mod/wikicode:managewiki', $context);
885                     $access = has_capability('moodle/site:accessallgroups', $context);
886                     return ($manage || $access) && $view;
887                 }
888             } else {
889                 //Error
890                 return false;
891             }
892         case VISIBLEGROUPS:
893             // Collaborative and Individual Mode
894             //
895             // Collaborative Mode:
896             //      There is one wiki per group.
897             // Individual Mode:
898             //      Each person owns a wiki.
899             if ($wiki->wikimode == 'collaborative' || $wiki->wikimode == 'individual')
                    {
900                 // Everybody can read all wikis
901                 //
902                 // Only view capability needed
903                 return has_capability('mod/wikicode:viewpage', $context);
904             } else {
905                 //Error
906                 return false;
907             }
908         default: // Error
909             return false;
910         }
911     }
912 
913     /**
914      * Checks if current user can edit a subwiki
915      *
916      * @param $subwiki
917      */
918     function wikicode_user_can_edit($subwiki) {
919         global $USER;
920 
```

```
921     $wiki = wikicode_get_wiki($subwiki->wikiid);
922     $cm = get_coursemodule_from_instance('wikicode', $wiki->id);
923     $context = get_context_instance(CONTEXT_MODULE, $cm->id);
924 
925     // Working depending on activity groupmode
926     switch (groups_get_activity_groupmode($cm)) {
927     case NOGROUPS:
928 
929         if ($wiki->wikimode == 'collaborative') {
930             // Collaborative Mode:
931             // There is a wiki for all the class.
932             //
933             // Only edit capbility needed
934             return has_capability('mod/wikicode:editpage', $context);
935         } else if ($wiki->wikimode == 'individual') {
936             // Individual Mode
937             // There is a wiki per user
938 
939             // Only the owner of that wiki can edit it
940             if ($subwiki->userid == $USER->id) {
941                 return has_capability('mod/wikicode:editpage', $context);
942             } else { // Current user is not the owner of that wiki.
943 
944                 // User must have:
945                 //      mod/wiki:editpage capability
946                 // and
947                 //      mod/wiki:managewiki capability
948                 $edit = has_capability('mod/wikicode:editpage', $context);
949                 $manage = has_capability('mod/wikicode:managewiki', $context);
950 
951                 return $edit && $manage;
952             }
953         } else {
954             //Error
955             return false;
956         }
957     case SEPARATEGROUPS:
958         if ($wiki->wikimode == 'collaborative') {
959             // Collaborative Mode:
960             // There is one wiki per group.
961             //
962             // Only members of subwiki group could edit that wiki
963             if ($subwiki->groupid == groups_get_activity_group($cm)) {
964                 // Only edit capability needed
965                 return has_capability('mod/wikicode:editpage', $context);
966             } else { // User is not part of that group
967                 // User must have:
968                 //      mod/wiki:managewiki capability
969                 // and
970                 //      moodle/site:accessallgroups capability
971                 // and
972                 //      mod/wiki:editpage capability
973                 $manage = has_capability('mod/wikicode:managewiki', $context);
974                 $access = has_capability('moodle/site:accessallgroups', $context);
```

```
975                 $edit = has_capability('mod/wikicode:editpage', $context);
976                 return $manage && $access && $edit;
977             }
978         } else if ($wiki->wikimode == 'individual') {
979             // Individual Mode:
980             // Each person owns a wiki.
981             //
982             // Only the owner of that wiki can edit it
983             if ($subwiki->userid == $USER->id) {
984                 return has_capability('mod/wikicode:editpage', $context);
985             } else { // Current user is not the owner of that wiki.
986                 // User must have:
987                 //      mod/wiki:managewiki capability
988                 // and
989                 //      moodle/site:accessallgroups capability
990                 // and
991                 //      mod/wiki:editpage capability
992                 $manage = has_capability('mod/wikicode:managewiki', $context);
993                 $access = has_capability('moodle/site:accessallgroups', $context);
994                 $edit = has_capability('mod/wikicode:editpage', $context);
995                 return $manage && $access && $edit;
996             }
997         } else {
998             //Error
999             return false;
1000        }
1001    case VISIBLEGROUPS:
1002        if ($wiki->wikimode == 'collaborative') {
1003            // Collaborative Mode:
1004            // There is one wiki per group.
1005            //
1006            // Only members of subwiki group could edit that wiki
1007            if (groups_is_member($subwiki->groupid)) {
1008                // Only edit capability needed
1009                return has_capability('mod/wikicode:editpage', $context);
1010            } else { // User is not part of that group
1011                // User must have:
1012                //      mod/wiki:managewiki capability
1013                // and
1014                //      mod/wiki:editpage capability
1015                $manage = has_capability('mod/wikicode:managewiki', $context);
1016                $edit = has_capability('mod/wikicode:editpage', $context);
1017                return $manage && $edit;
1018            }
1019        } else if ($wiki->wikimode == 'individual') {
1020            // Individual Mode:
1021            // Each person owns a wiki.
1022            //
1023            // Only the owner of that wiki can edit it
1024            if ($subwiki->userid == $USER->id) {
1025                return has_capability('mod/wikicode:editpage', $context);
1026            } else { // Current user is not the owner of that wiki.
1027                // User must have:
1028                //      mod/wiki:managewiki capability
```

```
1029                    // and
1030                    //      mod/wiki:editpage capability
1031                    $manage = has_capability('mod/wikicode:managewiki', $context);
1032                    $edit = has_capability('mod/wikicode:editpage', $context);
1033                    return $manage && $edit;
1034                }
1035            } else {
1036                //Error
1037                return false;
1038            }
1039        default: // Error
1040            return false;
1041        }
1042    }
1043
1044    //----------------
1045    // Locks
1046    //----------------
1047
1048    /**
1049     * Checks if a page-section is locked.
1050     *
1051     * @return true if the combination of section and page is locked, FALSE otherwise.
1052     */
1053    function wikicode_is_page_section_locked($pageid, $userid, $section = null) {
1054        global $DB;
1055
1056        $sql = "pageid = ? AND lockedat > ? AND userid != ?";
1057        $params = array($pageid, time(), $userid);
1058
1059        if (!empty($section)) {
1060            $sql .= " AND (sectionname = ? OR sectionname IS null)";
1061            $params[] = $section;
1062        }
1063
1064        return $DB->record_exists_select('wikicode_locks', $sql, $params);
1065    }
1066
1067    /**
1068     * Inserts or updates a wikicode_locks record.
1069     */
1070    function wikicode_set_lock($pageid, $userid, $section = null, $insert = false) {
1071        global $DB;
1072
1073        if (wikicode_is_page_section_locked($pageid, $userid, $section)) {
1074            return false;
1075        }
1076
1077        $params = array('pageid' => $pageid, 'userid' => $userid, 'sectionname' =>
            $section);
1078
1079        $lock = $DB->get_record('wikicode_locks', $params);
1080
1081        if (!empty($lock)) {
```

```
1082            $DB->update_record('wikicode_locks', array('id' => $lock->id, 'lockedat' =>
                     time() + LOCK_TIMEOUT));
1083        } else if ($insert) {
1084            $DB->insert_record('wikicode_locks', array('pageid' => $pageid, '
                    sectionname' => $section, 'userid' => $userid, 'lockedat' => time() +
                    30));
1085        }
1086
1087        return true;
1088    }
1089
1090    /**
1091     * Deletes wikicode_locks that are not in use. (F.Ex. after submitting the changes)
              . If no userid is present, it deletes ALL the wikicode_locks of a specific
             page.
1092     */
1093    function wikicode_delete_locks($pageid, $userid = null, $section = null,
            $delete_from_db = true, $delete_section_and_page = false) {
1094        global $DB;
1095
1096        $params = array('pageid' => $pageid);
1097
1098        if (!empty($userid)) {
1099            $params['userid'] = $userid;
1100        }
1101
1102        if (!empty($section)) {
1103            $params['sectionname'] = $section;
1104        }
1105
1106        if ($delete_from_db) {
1107            $DB->delete_records('wikicode_locks', $params);
1108            if ($delete_section_and_page && !empty($section)) {
1109                $params['sectionname'] = null;
1110                $DB->delete_records('wikicode_locks', $params);
1111            }
1112        } else {
1113            $DB->set_field('wikicode_locks', 'lockedat', time(), $params);
1114        }
1115    }
1116
1117    /**
1118     * Deletes wikicode_locks that expired 1 hour ago.
1119     */
1120    function wikicode_delete_old_locks() {
1121        global $DB;
1122
1123        $DB->delete_records_select('wikicode_locks', "lockedat < ?", array(time() -
            3600));
1124    }
1125
1126    /**
1127     * Deletes wikicode_links. It can be sepecific link or links attached in subwiki
1128     *
```

```
1129     * @global mixed $DB database object
1130     * @param int $linkid id of the link to be deleted
1131     * @param int $topageid links to the specific page
1132     * @param int $frompageid links from specific page
1133     * @param int $subwikiid links to subwiki
1134     */
1135    function wikicode_delete_links($linkid = null, $topageid = null, $frompageid = null
         , $subwikiid = null) {
1136        global $DB;
1137        $params = array();
1138
1139        // if link id is givien then don't check for anything else
1140        if (!empty($linkid)) {
1141            $params['id'] = $linkid;
1142        } else {
1143            if (!empty($topageid)) {
1144                $params['topageid'] = $topageid;
1145            }
1146            if (!empty($frompageid)) {
1147                $params['frompageid'] = $frompageid;
1148            }
1149            if (!empty($subwikiid)) {
1150                $params['subwikiid'] = $subwikiid;
1151            }
1152        }
1153
1154        //Delete links if any params are passed, else nothing to delete.
1155        if (!empty($params)) {
1156            $DB->delete_records('wikicode_links', $params);
1157        }
1158    }
1159
1160    /**
1161     * Delete wiki synonyms related to subwikiid or page
1162     *
1163     * @param int $subwikiid id of sunbwiki
1164     * @param int $pageid id of page
1165     */
1166    function wikicode_delete_synonym($subwikiid, $pageid = null) {
1167        global $DB;
1168
1169        $params = array('subwikiid' => $subwikiid);
1170        if (!is_null($pageid)) {
1171            $params['pageid'] = $pageid;
1172        }
1173        $DB->delete_records('wikicode_synonyms', $params, IGNORE_MISSING);
1174    }
1175
1176    /**
1177     * Delete pages and all related data
1178     *
1179     * @param mixed $context context in which page needs to be deleted.
1180     * @param mixed $pageids id's of pages to be deleted
1181     * @param int $subwikiid id of the subwiki for which all pages should be deleted
```

```
1182  */
1183 function wikicode_delete_pages($context, $pageids = null, $subwikiid = null) {
1184     global $DB;
1185 
1186     if (!empty($pageids) && is_int($pageids)) {
1187        $pageids = array($pageids);
1188     } else if (!empty($subwikiid)) {
1189         $pageids = wikicode_get_page_list($subwikiid);
1190     }
1191 
1192     //If there is no pageid then return as we can't delete anything.
1193     if (empty($pageids)) {
1194         return;
1195     }
1196 
1197     /// Delete page and all it's relevent data
1198     foreach ($pageids as $pageid) {
1199         if (is_object($pageid)) {
1200             $pageid = $pageid->id;
1201         }
1202 
1203         //Delete page comments
1204         $comments = wikicode_get_comments($context->id, $pageid);
1205         foreach ($comments as $commentid => $commentvalue) {
1206             wikicode_delete_comment($commentid, $context, $pageid);
1207         }
1208 
1209         //Delete page tags
1210         $tags = tag_get_tags_array('wikicode_pages', $pageid);
1211         foreach ($tags as $tagid => $tagvalue) {
1212             tag_delete_instance('wikicode_pages', $pageid, $tagid);
1213         }
1214 
1215         //Delete Synonym
1216         wikicode_delete_synonym($subwikiid, $pageid);
1217 
1218         //Delete all page versions
1219         wikicode_delete_page_versions(array($pageid=>array(0)));
1220 
1221         //Delete all page locks
1222         wikicode_delete_locks($pageid);
1223 
1224         //Delete all page links
1225         wikicode_delete_links(null, $pageid);
1226 
1227         //Delete page
1228         $params = array('id' => $pageid);
1229         $DB->delete_records('wikicode_pages', $params);
1230     }
1231 }
1232 
1233 /**
1234  * Delete specificed versions of a page or versions created by users
1235  * if version is 0 then it will remove all versions of the page
```

```
1236     *
1237     * @param array $deleteversions delete versions for a page
1238     */
1239    function wikicode_delete_page_versions($deleteversions) {
1240        global $DB;
1241
1242        /// delete page-versions
1243        foreach ($deleteversions as $id => $versions) {
1244            foreach ($versions as $version) {
1245                $params = array('pageid' => $id);
1246                //If version = 0, then remove all versions of this page, else remove
1247                //specified version
1248                if ($version != 0) {
1249                    $params['version'] = $version;
1250                }
1251                $DB->delete_records('wikicode_versions', $params, IGNORE_MISSING);
1252            }
1253        }
1254    }
1255
1256    function wikicode_get_comment($commentid){
1257        global $DB;
1258        return $DB->get_record('comments', array('id' => $commentid));
1259    }
1260
1261    /**
1262     * Returns all comments by context and pageid
1263     *
1264     * @param $context. Current context
1265     * @param $pageid. Current pageid
1266     **/
1267    function wikicode_get_comments($contextid, $pageid) {
1268        global $DB;
1269
1270        return $DB->get_records('comments', array('contextid' => $contextid, 'itemid'
            => $pageid, 'commentarea' => 'wikicode_page'));
1271    }
1272
1273    /**
1274     * Add comments ro database
1275     *
1276     * @param object $context. Current context
1277     * @param int $pageid. Current pageid
1278     * @param string $content. Content of the comment
1279     * @param string editor. Version of editor we are using.
1280     **/
1281    function wikicode_add_comment($context, $pageid, $content, $editor) {
1282        global $CFG;
1283        require_once($CFG->dirroot . '/comment/lib.php');
1284
1285        list($context, $course, $cm) = get_context_info_array($context->id);
1286        $cmt = new stdclass();
1287        $cmt->context = $context;
1288        $cmt->itemid = $pageid;
```

```
1289        $cmt->area = 'wikicode_page';
1290        $cmt->course = $course;
1291        $cmt->component = 'mod_wikicode';
1292
1293        $manager = new comment($cmt);
1294
1295        if ($editor == 'creole') {
1296            $manager->add($content, FORMAT_CREOLE);
1297        } else if ($editor == 'html') {
1298            $manager->add($content, FORMAT_HTML);
1299        } else if ($editor == 'nwiki') {
1300            $manager->add($content, FORMAT_NWIKI);
1301        }
1302
1303    }
1304
1305    /**
1306     * Delete comments from database
1307     *
1308     * @param $idcomment. Id of comment which will be deleted
1309     * @param $context. Current context
1310     * @param $pageid. Current pageid
1311     **/
1312    function wikicode_delete_comment($idcomment, $context, $pageid) {
1313        global $CFG;
1314        require_once($CFG->dirroot . '/comment/lib.php');
1315
1316        list($context, $course, $cm) = get_context_info_array($context->id);
1317        $cmt = new stdClass();
1318        $cmt->context = $context;
1319        $cmt->itemid = $pageid;
1320        $cmt->area = 'wikicode_page';
1321        $cmt->course = $course;
1322        $cmt->component = 'mod_wikicode';
1323
1324        $manager = new comment($cmt);
1325        $manager->delete($idcomment);
1326
1327    }
1328
1329    /**
1330     * Delete al comments from wiki
1331     *
1332     **/
1333    function wikicode_delete_comments_wiki() {
1334        global $PAGE, $DB;
1335
1336        $cm = $PAGE->cm;
1337        $context = get_context_instance(CONTEXT_MODULE, $cm->id);
1338
1339        $table = 'comments';
1340        $select = 'contextid = ?';
1341
1342        $DB->delete_records_select($table, $select, array($context->id));
```

```
1343
1344  }
1345
1346  function wikicode_add_progress($pageid, $oldversionid, $versionid, $progress) {
1347      global $DB;
1348      for ($v = $oldversionid + 1; $v <= $versionid; $v++) {
1349          $user = wikicode_get_wikicode_page_id($pageid, $v);
1350
1351          $DB->insert_record('wikicode_progress', array('userid' => $user->userid, '
                pageid' => $pageid, 'versionid' => $v, 'progress' => $progress));
1352      }
1353  }
1354
1355  function wikicode_get_wikicode_page_id($pageid, $id) {
1356      global $DB;
1357      return $DB->get_record('wikicode_versions', array('pageid' => $pageid, 'id' =>
            $id));
1358  }
1359
1360  function wikicode_print_page_content($page, $context, $subwikiid) {
1361      global $OUTPUT, $CFG;
1362
1363      if ($page->timerendered + WIKI_REFRESH_CACHE_TIME < time()) {
1364          $content = wikicode_refresh_cachedcontent($page);
1365          $page = $content['page'];
1366      }
1367
1368      if (isset($content)) {
1369          $box = '';
1370          foreach ($content['sections'] as $s) {
1371              $box .= '<p>' . get_string('repeatedsection', 'wikicode', $s) . '</p>';
1372          }
1373
1374          if (!empty($box)) {
1375              echo $OUTPUT->box($box);
1376          }
1377      }
1378
1379      $html = file_rewrite_pluginfile_urls(wikicode_remove_tags($page->cachedcontent)
            , 'pluginfile.php', $context->id, 'mod_wikicode', 'attachments', $subwikiid
            );
1380      $html = format_text($html, FORMAT_PLAIN, array('overflowdiv'=>true));
1381
1382      echo $OUTPUT->box($html);
1383
1384      if (!empty($CFG->usetags)) {
1385          $tags = tag_get_tags_array('wikicode_pages', $page->id);
1386          echo $OUTPUT->container_start('wiki-tags');
1387          echo '<span class="wiki-tags-title">'.get_string('tags').': </span>';
1388          $links = array();
1389          foreach ($tags as $tagid=>$tag) {
1390              $url = new moodle_url('/tag/index.php', array('tag'=>$tag));
1391              $links[] = html_writer::link($url, $tag, array('title'=>get_string('
                    tagtitle', 'wiki', $tag)));
```

```
1392            }
1393            echo join($links, ", ");
1394            echo $OUTPUT->container_end();
1395        }
1396
1397        wikicode_increment_pageviews($page);
1398    }
1399
1400    /**
1401     * This function trims any given text and returns it with some dots at the end
1402     *
1403     * @param string $text
1404     * @param string $limit
1405     *
1406     * @return string
1407     */
1408    function wikicode_trim_string($text, $limit = 25) {
1409
1410        if (textlib::strlen($text) > $limit) {
1411            $text = textlib::substr($text, 0, $limit) . '...';
1412        }
1413
1414        return $text;
1415    }
1416
1417    /**
1418     * Prints default edit form fields and buttons
1419     *
1420     * @param string $format Edit form format (html, creole...)
1421     * @param integer $version Version number. A negative number means no versioning.
1422     */
1423
1424    function wikicode_print_edit_form_default_fields($format, $pageid, $version = -1,
        $upload = false, $deleteuploads = array()) {
1425        global $CFG, $PAGE, $OUTPUT;
1426
1427        echo '<input type="hidden" name="sesskey" value="' . sesskey() . '" />';
1428
1429        if ($version >= 0) {
1430            echo '<input type="hidden" name="version" value="' . $version . '" />';
1431        }
1432
1433        echo '<input type="hidden" name="format" value="' . $format . '"/>';
1434
1435        //attachments
1436        require_once($CFG->dirroot . '/lib/form/filemanager.php');
1437
1438        $filemanager = new MoodleQuickForm_filemanager('attachments', get_string('
            wikiattachments', 'wikicode'), array('id' => 'attachments'), array('subdirs
            ' => false, 'maxfiles' => 99, 'maxbytes' => $CFG->maxbytes));
1439
1440        $value = file_get_submitted_draft_itemid('attachments');
1441        if (!empty($value) && !$upload) {
1442            $filemanager->setValue($value);
```

```
1443        }
1444
1445        echo "<fieldset class=\"wiki-upload-section clearfix\"><legend class=\"ftoggler
                \">" . get_string("uploadtitle", 'wikicode') . "</legend>";
1446
1447        echo $OUTPUT->container_start('mdl-align wiki-form-center aaaaa');
1448        print $filemanager->toHtml();
1449        echo $OUTPUT->container_end();
1450
1451        $cm = $PAGE->cm;
1452        $context = get_context_instance(CONTEXT_MODULE, $cm->id);
1453
1454        echo $OUTPUT->container_start('mdl-align wiki-form-center wiki-upload-table');
1455        wikicode_print_upload_table($context, 'wikicode_upload', $pageid,
                $deleteuploads);
1456        echo $OUTPUT->container_end();
1457
1458        echo "</fieldset>";
1459
1460        echo '<input class="wiki_button" type="submit" name="editoption" value="' .
                get_string('save', 'wikicode') . '"/>';
1461        echo '<input class="wiki_button" type="submit" name="editoption" value="' .
                get_string('upload', 'wikicode') . '"/>';
1462        echo '<input class="wiki_button" type="submit" name="editoption" value="' .
                get_string('preview') . '"/>';
1463        echo '<input class="wiki_button" type="submit" name="editoption" value="' .
                get_string('cancel') . '" />';
1464    }
1465
1466    /**
1467     * Prints a table with the files attached to a wiki page
1468     * @param object $context
1469     * @param string $filearea
1470     * @param int $fileitemid
1471     * @param array deleteuploads
1472     */
1473    function wikicode_print_upload_table($context, $filearea, $fileitemid,
            $deleteuploads = array()) {
1474        global $CFG, $OUTPUT;
1475
1476        $htmltable = new html_table();
1477
1478        $htmltable->head = array(get_string('deleteupload', 'wikicode'), get_string('
                uploadname', 'wikicode'), get_string('uploadactions', 'wiki'));
1479
1480        $fs = get_file_storage();
1481        $files = $fs->get_area_files($context->id, 'mod_wikicode', $filearea,
                $fileitemid); //TODO: this is weird (skodak)
1482
1483        foreach ($files as $file) {
1484            if (!$file->is_directory()) {
1485                $checkbox = '<input type="checkbox" name="deleteupload[]", value="' .
                        $file->get_pathnamehash() . '"';
1486
```

```
1487                if (in_array($file->get_pathnamehash(), $deleteuploads)) {
1488                    $checkbox .= ' checked="checked"';
1489                }
1490
1491                $checkbox .= " />";
1492
1493                $htmltable->data[] = array($checkbox, '<a href="' . file_encode_url(
                        $CFG->wwwroot . '/pluginfile.php', '/' . $context->id . '/
                        wikicode_upload/' . $fileitemid . '/' . $file->get_filename()) . '
                        ">' . $file->get_filename() . '</a>', "");
1494            }
1495        }
1496
1497        print '<h3 class="upload-table-title">' . get_string('uploadfiletitle', '
                wikicode') . "</h3>";
1498        print html_writer::table($htmltable);
1499    }
1500
1501    /**
1502     * Generate wiki's page tree
1503     *
1504     * @param $page. A wiki page object
1505     * @param $node. Starting navigation_node
1506     * @param $keys. An array to store keys
1507     * @return an array with all tree nodes
1508     */
1509    function wikicode_build_tree($page, $node, &$keys) {
1510        $content = array();
1511        static $icon;
1512        $icon = new pix_icon('f/odt', '');
1513        $pages = wikicode_get_linked_pages($page->id);
1514        foreach ($pages as $p) {
1515            $key = $page->id . ':' . $p->id;
1516            if (in_array($key, $keys)) {
1517                break;
1518            }
1519            array_push($keys, $key);
1520            $l = wikicode_parser_link($p);
1521            $link = new moodle_url('/mod/wikicode/view.php', array('pageid' => $p->id))
                    ;
1522            $nodeaux = $node->add($p->title, $link, null, null, null, $icon);
1523            if ($l['new']) {
1524                $nodeaux->add_class('wikicode_newentry');
1525            }
1526            wikicode_build_tree($p, $nodeaux, $keys);
1527        }
1528        $content[] = $node;
1529        return $content;
1530    }
1531
1532    /**
1533     * Get linked pages from page
1534     * @param int $pageid
1535     */
```

```
1536 function wikicode_get_linked_pages($pageid) {
1537     global $DB;
1538 
1539     $sql = "SELECT p.id, p.title
1540             FROM {wikicode_pages} p
1541             JOIN {wikicode_links} l ON l.topageid = p.id
1542             WHERE l.frompageid = ?
1543             ORDER BY p.title ASC";
1544     return $DB->get_records_sql($sql, array($pageid));
1545 }
1546 
1547 /**
1548  * Get updated pages from wiki
1549  * @param int $pageid
1550  */
1551 function wikicode_get_updated_pages_by_subwiki($swid) {
1552     global $DB, $USER;
1553 
1554     $sql = "SELECT *
1555             FROM {wikicode_pages}
1556             WHERE subwikiid = ? AND timemodified > ?
1557             ORDER BY timemodified DESC";
1558     return $DB->get_records_sql($sql, array($swid, $USER->lastlogin));
1559 }
```

3.2.4. map.php

```
1  <?php
2  
3  require_once('../../config.php');
4  require_once($CFG->dirroot . '/mod/wikicode/lib.php');
5  require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
6  require_once($CFG->dirroot . '/mod/wikicode/pagelib.php');
7  
8  $pageid = required_param('pageid', PARAM_INT); // Page ID
9  $option = optional_param('option', 0, PARAM_INT); // Option ID
10 
11 if (!$page = wikicode_get_page($pageid)) {
12     print_error('incorrectpageid', 'wikicode');
13 }
14 if (!$subwiki = wikicode_get_subwiki($page->subwikiid)) {
15     print_error('incorrectsubwikiid', 'wikicode');
16 }
17 if (!$cm = get_coursemodule_from_instance("wikicode", $subwiki->wikiid)) {
18     print_error('invalidcoursemodule');
19 }
20 $course = $DB->get_record('course', array('id' => $cm->course), '*', MUST_EXIST);
21 if (!$wiki = wikicode_get_wiki($subwiki->wikiid)) {
22     print_error('incorrectwikiid', 'wikicode');
23 }
24 
```

```
25  require_login($course->id, true, $cm);
26  $context = get_context_instance(CONTEXT_MODULE, $cm->id);
27  require_capability('mod/wikicode:viewpage', $context);
28  add_to_log($course->id, "wiki", "map", "map.php?id=$cm->id", "$wiki->id");
29
30  /// Print page header
31
32  /// Finish the page
33  $wikipage = new page_wikicode_map($wiki, $subwiki, $cm);
34
35  $wikipage->set_view($option);
36  $wikipage->set_page($page);
37  $wikipage->print_header();
38  $wikipage->print_content();
39
40  $wikipage->print_footer();
```

3.2.5. overridelocks.php

```
1   <?php
2
3   require_once('../../config.php');
4
5   require_once($CFG->dirroot . '/mod/wikicode/lib.php');
6   require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
7   require_once($CFG->dirroot . '/mod/wikicode/pagelib.php');
8
9   $pageid = required_param('pageid', PARAM_INT);
10  $section = optional_param('section', '', PARAM_TEXT);
11
12  if (!$page = wikicode_get_page($pageid)) {
13      print_error('incorrectpageid', 'wikicode');
14  }
15
16  if (!$subwiki = wikicode_get_subwiki($page->subwikiid)) {
17      print_error('incorrectsubwikiid', 'wikicode');
18  }
19
20  if (!$wiki = wikicode_get_wiki($subwiki->wikiid)) {
21      print_error('incorrectwikiid', 'wikicode');
22  }
23
24  if (!$cm = get_coursemodule_from_instance('wikicode', $wiki->id)) {
25      print_error('invalidcoursemodule');
26  }
27
28  $course = $DB->get_record('course', array('id' => $cm->course), '*', MUST_EXIST);
29
30  if (!empty($section) && !$sectioncontent = wikicode_get_section_page($page,
        $section)) {
31      print_error('invalidsection', 'wikicode');
```

```
}

require_login($course->id, true, $cm);

$context = get_context_instance(CONTEXT_MODULE, $cm->id);
require_capability('mod/wikicode:overridelock', $context);

add_to_log($course->id, "wikicode", "overridelocks", "overridelocks.php?id=$cm->id"
    , "$wiki->id");

if (!confirm_sesskey()) {
    print_error(get_string('invalidsesskey', 'wikicode'));
}

$wikipage = new page_wikicode_overridelocks($wiki, $subwiki, $cm);
$wikipage->set_page($page);

if (!empty($section)) {
    $wikipage->set_section($sectioncontent, $section);
}

$wikipage->print_header();

$wikipage->print_content();

$wikipage->print_footer();
```

3.2.6. prettyview.php

```
<?php

require_once('../../config.php');
require_once($CFG->dirroot . '/mod/wikicode/lib.php');
require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
require_once($CFG->dirroot . '/mod/wikicode/pagelib.php');

$pageid = required_param('pageid', PARAM_INT); // Page ID

if (!$page = wikicode_get_page($pageid)) {
    print_error('incorrectpageid', 'wikicode');
}
if (!$subwiki = wikicode_get_subwiki($page->subwikiid)) {
    print_error('incorrectsubwikiid', 'wikicode');
}
if (!$cm = get_coursemodule_from_instance("wikicode", $subwiki->wikiid)) {
    print_error('invalidcoursemodule');
}
$course = $DB->get_record('course', array('id' => $cm->course), '*', MUST_EXIST);
if (!$wiki = wikicode_get_wiki($subwiki->wikiid)) {
    print_error('incorrectwikiid', 'wikicode');
}
```

```
23
24  require_login($course->id, true, $cm);
25
26  $context = get_context_instance(CONTEXT_MODULE, $cm->id);
27  require_capability('mod/wikicode:viewpage', $context);
28
29  add_to_log($course->id, "wikicode", "view", "prettyview.php?pageid=$pageid", "$wiki
        ->id");
30
31  $wikipage = new page_wikicode_prettyview($wiki, $subwiki, $cm);
32
33  $wikipage->set_page($page);
34
35  $wikipage->print_header();
36  $wikipage->print_content();
37  $wikipage->print_footer();
```

3.2.7. renderer.php

```
1   <?php
2
3   /**
4    * Moodle Wiki 2.0 Renderer
5    *
6    * @package   mod-wikicode
7    * @copyright 2010 Dongsheng Cai <dongsheng@moodle.com>
8    * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
9    */
10
11  defined('MOODLE_INTERNAL') || die();
12
13  class mod_wikicode_renderer extends plugin_renderer_base {
14      public function page_index() {
15          global $CFG;
16          $output = '';
17          // Checking wiki instance
18          if (!$wiki = wikicode_get_wiki($this->page->cm->instance)) {
19              return false;
20          }
21
22          // @TODO: Fix call to wikicode_get_subwiki_by_group
23          $gid = groups_get_activity_group($this->page->cm);
24          $gid = !empty($gid) ? $gid : 0;
25          if (!$subwiki = wikicode_get_subwiki_by_group($this->page->cm->instance,
                $gid)) {
26              return false;
27          }
28          $swid = $subwiki->id;
29          $pages = wikicode_get_page_list($swid);
30          $selectoptions = array();
31          foreach ($pages as $page) {
```

```
32              $selectoptions[$page->id] = $page->title;
33          }
34          $label = get_string('pageindex', 'wikicode') . ': ';
35          $select = new single_select(new moodle_url('/mod/wikicode/view.php'), '
                pageid', $selectoptions);
36          $select->label = $label;
37          return $this->output->container($this->output->render($select), '
                wikicode_index');
38      }
39
40      public function search_result($records, $subwiki) {
41          global $CFG, $PAGE;
42          $table = new html_table();
43          $context = get_context_instance(CONTEXT_MODULE, $PAGE->cm->id);
44          $strsearchresults = get_string('searchresult', 'wikicode');
45          $totalcount = count($records);
46          $html = $this->output->heading("$strsearchresults $totalcount");
47          foreach ($records as $page) {
48              $table->head = array('title' => format_string($page->title) . ' (' .
                    html_writer::link($CFG->wwwroot . '/mod/wikicode/view.php?pageid='
                    . $page->id, get_string('view', 'wikicode')) . ')');
49              $table->align = array('title' => 'left');
50              $table->width = '100%';
51              $table->data = array(array(file_rewrite_pluginfile_urls(format_text(
                    $page->cachedcontent, FORMAT_HTML), 'pluginfile.php', $context->id,
                     'mod_wikicode', 'attachments', $subwiki->id)));
52              $table->colclasses = array('wikisearchresults');
53              $html .= html_writer::table($table);
54          }
55          $html = html_writer::tag('div', $html, array('class'=>'no-overflow'));
56          return $this->output->container($html);
57      }
58
59      public function diff($pageid, $old, $new, $options = array()) {
60          global $CFG;
61          if (!empty($options['total'])) {
62              $total = $options['total'];
63          } else {
64              $total = 0;
65          }
66          $diff1 = format_text($old->diff, FORMAT_HTML, array('overflowdiv'=>true));
67          $diff2 = format_text($new->diff, FORMAT_HTML, array('overflowdiv'=>true));
68          $strdatetime = get_string('strftimedatetime', 'langconfig');
69
70          $olduser = $old->user;
71          $versionlink = new moodle_url('/mod/wikicode/viewversion.php', array('
                pageid' => $pageid, 'versionid' => $old->id));
72          $restorelink = new moodle_url('/mod/wikicode/restoreversion.php', array('
                pageid' => $pageid, 'versionid' => $old->id));
73          $userlink = new moodle_url('/user/view.php', array('id' => $olduser->id));
74          // view version link
75          $oldversionview = ' ';
76          $oldversionview .= html_writer::link($versionlink->out(false), get_string('
                view', 'wikicode'), array('class' => 'wikicode_diffview'));
```

```
77          $oldversionview .= ' ';
78          // restore version link
79          $oldversionview .= html_writer::link($restorelink->out(false), get_string('
                restore', 'wikicode'), array('class' => 'wikicode_diffview'));
80
81          // userinfo container
82          $oldheading = $this->output->container_start('wikicode_diffuserleft');
83          // username
84          $oldheading .= html_writer::link($CFG->wwwroot . '/user/view.php?id=' .
                $olduser->id, fullname($olduser)) . ' ';
85          // user picture
86          $oldheading .= html_writer::link($userlink->out(false), $this->output->
                user_picture($olduser, array('popup' => true)), array('class' => '
                notunderlined'));
87          $oldheading .= $this->output->container_end();
88
89          // version number container
90          $oldheading .= $this->output->container_start('wikicode_diffversion');
91          $oldheading .= get_string('version') . ' ' . $old->version .
                $oldversionview;
92          $oldheading .= $this->output->container_end();
93          // userdate container
94          $oldheading .= $this->output->container_start('wikicode_difftime');
95          $oldheading .= userdate($old->timecreated, $strdatetime);
96          $oldheading .= $this->output->container_end();
97
98          $newuser = $new->user;
99          $versionlink = new moodle_url('/mod/wikicode/viewversion.php', array('
                pageid' => $pageid, 'versionid' => $new->id));
100         $restorelink = new moodle_url('/mod/wikicode/restoreversion.php', array('
                pageid' => $pageid, 'versionid' => $new->id));
101         $userlink = new moodle_url('/user/view.php', array('id' => $newuser->id));
102
103         $newversionview = ' ';
104         $newversionview .= html_writer::link($versionlink->out(false), get_string('
                view', 'wikicode'), array('class' => 'wikicode_diffview'));
105         // new user info
106         $newheading = $this->output->container_start('wikicode_diffuserright');
107         $newheading .= $this->output->user_picture($newuser, array('popup' => true)
                );
108
109         $newheading .= html_writer::link($userlink->out(false), fullname($newuser),
                 array('class' => 'notunderlined'));
110         $newheading .= $this->output->container_end();
111
112         // version
113         $newheading .= $this->output->container_start('wikicode_diffversion');
114         $newheading .= get_string('version') . ' ' . $new->version .
                $newversionview;
115         $newheading .= $this->output->container_end();
116         // userdate
117         $newheading .= $this->output->container_start('wikicode_difftime');
118         $newheading .= userdate($new->timecreated, $strdatetime);
119         $newheading .= $this->output->container_end();
```

```
120
121         $oldheading = html_writer::tag('div', $oldheading, array('class'=>'wiki-
                diff-heading header clearfix'));
122         $newheading = html_writer::tag('div', $newheading, array('class'=>'wiki-
                diff-heading header clearfix'));
123
124         $output  = '';
125         $output .= html_writer::start_tag('div', array('class'=>'wiki-diff-
                container clearfix'));
126         $output .= html_writer::tag('div', $oldheading.$diff1, array('class'=>'wiki
                -diff-leftside'));
127         $output .= html_writer::tag('div', $newheading.$diff2, array('class'=>'wiki
                -diff-rightside'));
128         $output .= html_writer::end_tag('div');
129
130         if (!empty($total)) {
131             $output .= '<div class="wikicode_diff_paging">';
132             $output .= $this->output->container($this->diff_paging_bar(1, $new->
                    version - 1, $old->version, $CFG->wwwroot . '/mod/wikicode/diff.php
                    ?pageid=' . $pageid . '&amp;comparewith=' . $new->version . '&amp;'
                    , 'compare', false, true), 'wikicode_diff_oldpaging');
133             $output .= $this->output->container($this->diff_paging_bar($old->
                    version + 1, $total, $new->version, $CFG->wwwroot . '/mod/wikicode/
                    diff.php?pageid=' . $pageid . '&amp;compare=' . $old->version . '&
                    amp;', 'comparewith', false, true), 'wikicode_diff_newpaging');
134             $output .= '</div>';
135         }
136
137         return $output;
138     }
139
140     /**
141      * Prints a single paging bar to provide access to other versions
142      *
143      * @param int $minpage First page to be displayed in the bar
144      * @param int $maxpage Last page to be displayed in the bar
145      * @param int $page The page you are currently viewing
146      * @param mixed $baseurl If this  is a string then it is the url which will be
             appended with $pagevar, an equals sign and the page number.
147      *                          If this is a moodle_url object then the pagevar
             param will be replaced by the page no, for each page.
148      * @param string $pagevar This is the variable name that you use for the page
             number in your code (ie. 'tablepage', 'blogpage', etc)
149      * @param bool $nocurr do not display the current page as a link
150      * @param bool $return whether to return an output string or echo now
151      * @return bool or string
152      */
153     public function diff_paging_bar($minpage, $maxpage, $page, $baseurl, $pagevar =
             'page', $nocurr = false) {
154         $totalcount = $maxpage - $minpage;
155         $maxdisplay = 2;
156         $output = '';
157
158         if ($totalcount > 0) {
```

```
159                 $output .= '<div class="paging">';
160                 $output .= get_string('version', 'wikicode') . ':';
161                 if ($page - $minpage > 0) {
162                     $pagenum = $page - 1;
163                     if (!is_a($baseurl, 'moodle_url')) {
164                         $output .= ' (<a class="previous" href="' . $baseurl .
                                $pagevar . '=' . $pagenum . '">' . get_string('previous') .
                                 '</a>) ';
165                     } else {
166                         $output .= ' (<a class="previous" href="' . $baseurl->out(
                                false, array($pagevar => $pagenum)) . '">' . get_string('
                                previous') . '</a>) ';
167                     }
168                 }
169
170                 if ($page - $minpage > 4) {
171                     $startpage = $page - 3;
172                     if (!is_a($baseurl, 'moodle_url')) {
173                         $output .= ' <a href="' . $baseurl . $pagevar . '=' .
                                $minpage . '">' . $minpage . '</a> ...';
174                     } else {
175                         $output .= ' <a href="' . $baseurl->out(false, array(
                                $pagevar => $minpage)) . '">' . $minpage . '</a> ...';
176                     }
177                 } else {
178                     $startpage = $minpage;
179                 }
180                 $currpage = $startpage;
181                 $displaycount = 0;
182                 while ($displaycount < $maxdisplay and $currpage <= $maxpage) {
183                     if ($page == $currpage && empty($nocurr)) {
184                         $output .= '  ' . $currpage;
185                     } else {
186                         if (!is_a($baseurl, 'moodle_url')) {
187                             $output .= '  <a href="' . $baseurl . $pagevar .
                                    '=' . $currpage . '">' . $currpage . '</a>';
188                         } else {
189                             $output .= '  <a href="' . $baseurl->out(false,
                                    array($pagevar => $currpage)) . '">' . $currpage . '</a
                                    >';
190                         }
191
192                     }
193                     $displaycount++;
194                     $currpage++;
195                 }
196                 if ($currpage < $maxpage) {
197                     if (!is_a($baseurl, 'moodle_url')) {
198                         $output .= ' ...<a href="' . $baseurl . $pagevar . '=' .
                                $maxpage . '">' . $maxpage . '</a> ';
199                     } else {
200                         $output .= ' ...<a href="' . $baseurl->out(false, array(
                                $pagevar => $maxpage)) . '">' . $maxpage . '</a> ';
201                     }
```

```
202             } else if ($currpage == $maxpage) {
203                 if (!is_a($baseurl, 'moodle_url')) {
204                     $output .= '  <a href="' . $baseurl . $pagevar . '='
                            . $currpage . '">' . $currpage . '</a>';
205                 } else {
206                     $output .= '  <a href="' . $baseurl->out(false, array
                            ($pagevar => $currpage)) . '">' . $currpage . '</a>';
207                 }
208             }
209             $pagenum = $page + 1;
210             if ($page != $maxpage) {
211                 if (!is_a($baseurl, 'moodle_url')) {
212                     $output .= '  (<a class="next" href="' . $baseurl .
                            $pagevar . '=' . $pagenum . '">' . get_string('next') . '</
                            a>)';
213                 } else {
214                     $output .= '  (<a class="next" href="' . $baseurl->
                            out(false, array($pagevar => $pagenum)) . '">' . get_string
                            ('next') . '</a>)';
215                 }
216             }
217             $output .= '</div>';
218         }
219
220         return $output;
221     }
222     public function wikicode_info() {
223         global $PAGE;
224         return $this->output->box(format_module_intro('wikicode', $this->page->
                activityrecord, $PAGE->cm->id), 'generalbox', 'intro');
225     }
226
227     public function tabs($page, $tabitems, $options) {
228         global $CFG;
229         $tabs = array();
230         $baseurl = $CFG->wwwroot . '/mod/wikicode/';
231         $context = get_context_instance(CONTEXT_MODULE, $this->page->cm->id);
232
233         $pageid = null;
234         if (!empty($page)) {
235             $pageid = $page->id;
236         }
237
238         $selected = $options['activetab'];
239
240         // make specific tab linked even it is active
241         if (!empty($options['linkedwhenactive'])) {
242             $linked = $options['linkedwhenactive'];
243         } else {
244             $linked = '';
245         }
246
247         if (!empty($options['inactivetabs'])) {
248             $inactive = $options['inactivetabs'];
```

```
249         } else {
250             $inactive = array();
251         }
252 
253         foreach ($tabitems as $tab) {
254             if ($tab == 'edit' && !has_capability('mod/wikicode:editpage', $context
                ) {
255                 continue;
256             }
257             if ($tab == 'comments' && !has_capability('mod/wikicode:viewcomment',
                $context)) {
258                 continue;
259             }
260             if ($tab == 'files' && !has_capability('mod/wikicode:viewpage',
                $context)) {
261                 continue;
262             }
263             if (($tab == 'view' || $tab == 'map' || $tab == 'history') && !
                has_capability('mod/wikicode:viewpage', $context)) {
264                 continue;
265             }
266             if ($tab == 'admin' && !has_capability('mod/wikicode:managewiki',
                $context)) {
267                 continue;
268             }
269             $link = $baseurl . $tab . '.php?pageid=' . $pageid;
270             if ($linked == $tab) {
271                 $tabs[] = new tabobject($tab, $link, get_string($tab, 'wikicode'),
                    '', true);
272             } else {
273                 $tabs[] = new tabobject($tab, $link, get_string($tab, 'wikicode'));
274             }
275         }
276 
277         return print_tabs(array($tabs), $selected, $inactive, null, true);
278     }
279 
280     public function prettyview_link($page) {
281         $html = '';
282         $link = new moodle_url('/mod/wikicode/prettyview.php', array('pageid' =>
            $page->id));
283         $html .= $this->output->container_start('wikicode_right');
284         $html .= $this->output->action_link($link, get_string('prettyprint', '
            wikicode'), new popup_action('click', $link));
285         $html .= $this->output->container_end();
286         return $html;
287     }
288 
289     public function wikicode_print_subwiki_selector($wiki, $subwiki, $page,
        $pagetype = 'view') {
290         global $CFG, $USER;
291         require_once($CFG->dirroot . '/user/lib.php');
292         switch ($pagetype) {
293         case 'files':
```

```
294             $baseurl = new moodle_url('/mod/wikicode/files.php');
295             break;
296         case 'view':
297         default:
298             $baseurl = new moodle_url('/mod/wikicode/view.php');
299             break;
300         }
301
302         $cm = get_coursemodule_from_instance('wikicode', $wiki->id);
303         $context = get_context_instance(CONTEXT_MODULE, $cm->id);
304         // @TODO: A plenty of duplicated code below this lines.
305         // Create private functions.
306         switch (groups_get_activity_groupmode($cm)) {
307         case NOGROUPS:
308             if ($wiki->wikimode == 'collaborative') {
309                 // No need to print anything
310                 return;
311             } else if ($wiki->wikimode == 'individual') {
312                 // We have private wikis here
313
314                 $view = has_capability('mod/wikicode:viewpage', $context);
315                 $manage = has_capability('mod/wikicode:managewiki', $context);
316
317                 // Only people with these capabilities can view all wikis
318                 if ($view && $manage) {
319                     // @TODO: Print here a combo that contains all users.
320                     $users = get_enrolled_users($context);
321                     $options = array();
322                     foreach ($users as $user) {
323                         $options[$user->id] = fullname($user);
324                     }
325
326                     echo $this->output->container_start('wikicode_right');
327                     $params = array('wid' => $wiki->id, 'title' => $page->title);
328                     if ($pagetype == 'files') {
329                         $params['pageid'] = $page->id;
330                     }
331                     $baseurl->params($params);
332                     $name = 'uid';
333                     $selected = $subwiki->userid;
334                     echo $this->output->single_select($baseurl, $name, $options,
                            $selected);
335                     echo $this->output->container_end();
336                 }
337                 return;
338             } else {
339                 // error
340                 return;
341             }
342         case SEPARATEGROUPS:
343             if ($wiki->wikimode == 'collaborative') {
344                 // We need to print a select to choose a course group
345
346                 $params = array('wid'=>$wiki->id, 'title'=>$page->title);
```

```
347                     if ($pagetype == 'files') {
348                         $params['pageid'] = $page->id;
349                     }
350                     $baseurl->params($params);
351 
352                     echo $this->output->container_start('wikicode_right');
353                     groups_print_activity_menu($cm, $baseurl->out());
354                     echo $this->output->container_end();
355                     return;
356                 } else if ($wiki->wikimode == 'individual') {
357                     //  @TODO: Print here a combo that contains all users of that
                            subwiki.
358                     $view = has_capability('mod/wikicode:viewpage', $context);
359                     $manage = has_capability('mod/wikicode:managewiki', $context);
360 
361                     // Only people with these capabilities can view all wikis
362                     if ($view && $manage) {
363                         $users = get_enrolled_users($context);
364                         $options = array();
365                         foreach ($users as $user) {
366                             $groups = groups_get_all_groups($cm->course, $user->id);
367                             if (!empty($groups)) {
368                                 foreach ($groups as $group) {
369                                     $options[$group->id][$group->name][$group->id . '-'
                                            . $user->id] = fullname($user);
370                                 }
371                             } else {
372                                 $name = get_string('notingroup', 'wikicode');
373                                 $options[0][$name]['0' . '-' . $user->id] = fullname(
                                        $user);
374                             }
375                         }
376                     } else {
377                         $group = groups_get_group($subwiki->groupid);
378                         $users = groups_get_members($subwiki->groupid);
379                         foreach ($users as $user) {
380                             $options[$group->id][$group->name][$group->id . '-' . $user
                                    ->id] = fullname($user);
381                         }
382                     }
383                     echo $this->output->container_start('wikicode_right');
384                     $params = array('wid' => $wiki->id, 'title' => $page->title);
385                     if ($pagetype == 'files') {
386                         $params['pageid'] = $page->id;
387                     }
388                     $baseurl->params($params);
389                     $name = 'groupanduser';
390                     $selected = $subwiki->groupid . '-' . $subwiki->userid;
391                     echo $this->output->single_select($baseurl, $name, $options,
                            $selected);
392                     echo $this->output->container_end();
393 
394                     return;
395 
```

```
396             } else {
397                 // error
398                 return;
399             }
400         CASE VISIBLEGROUPS:
401             if ($wiki->wikimode == 'collaborative') {
402                 // We need to print a select to choose a course group
403                 $params = array('wid'=>$wiki->id, 'title'=>urlencode($page->title))
                        ;
404                 if ($pagetype == 'files') {
405                     $params['pageid'] = $page->id;
406                 }
407                 $baseurl->params($params);
408 
409                 echo $this->output->container_start('wikicode_right');
410                 groups_print_activity_menu($cm, $baseurl->out());
411                 echo $this->output->container_end();
412                 return;
413 
414             } else if ($wiki->wikimode == 'individual') {
415                 $users = get_enrolled_users($context);
416                 $options = array();
417                 foreach ($users as $user) {
418                     $groups = groups_get_all_groups($cm->course, $user->id);
419                     if (!empty($groups)) {
420                         foreach ($groups as $group) {
421                             $options[$group->id][$group->name][$group->id . '-' .
                                    $user->id] = fullname($user);
422                         }
423                     } else {
424                         $name = get_string('notingroup', 'wikicode');
425                         $options[0][$name]['0' . '-' . $user->id] = fullname($user)
                                ;
426                     }
427                 }
428 
429                 echo $this->output->container_start('wikicode_right');
430                 $params = array('wid' => $wiki->id, 'title' => $page->title);
431                 if ($pagetype == 'files') {
432                     $params['pageid'] = $page->id;
433                 }
434                 $baseurl->params($params);
435                 $name = 'groupanduser';
436                 $selected = $subwiki->groupid . '-' . $subwiki->userid;
437                 echo $this->output->single_select($baseurl, $name, $options,
                        $selected);
438                 echo $this->output->container_end();
439 
440                 return;
441 
442             } else {
443                 // error
444                 return;
445             }
```

```
446             default:
447                 // error
448                 return;
449 
450             }
451 
452         }
453 
454         function menu_map($pageid, $currentselect) {
455             $options = array('contributions', 'links', 'orphaned', 'pageindex', '
                    pagelist', 'updatedpages');
456             $items = array();
457             foreach ($options as $opt) {
458                 $items[] = get_string($opt, 'wikicode');
459             }
460             $selectoptions = array();
461             foreach ($items as $key => $item) {
462                 $selectoptions[$key + 1] = $item;
463             }
464             $select = new single_select(new moodle_url('/mod/wikicode/map.php', array('
                    pageid' => $pageid)), 'option', $selectoptions, $currentselect);
465             $select->label = get_string('mapmenu', 'wikicode') . ': ';
466             return $this->output->container($this->output->render($select), 'midpad');
467         }
468         public function wikicode_files_tree($context, $subwiki) {
469             return $this->render(new wikicode_files_tree($context, $subwiki));
470         }
471         public function render_wikicode_files_tree(wikicode_files_tree $tree) {
472             if (empty($tree->dir['subdirs']) && empty($tree->dir['files'])) {
473                 $html = $this->output->box(get_string('nofilesavailable', 'repository')
                        );
474             } else {
475                 $htmlid = 'wikicode_files_tree_'.uniqid();
476                 $module = array('name'=>'mod_wikicode', 'fullpath'=>'/mod/wikicode/
                        module.js');
477                 $this->page->requires->js_init_call('M.mod_wikicode.init_tree', array(
                        false, $htmlid), false, $module);
478                 $html = '<div id="'.$htmlid.'">';
479                 $html .= $this->htmllize_tree($tree, $tree->dir);
480                 $html .= '</div>';
481             }
482             return $html;
483         }
484 
485         function menu_admin($pageid, $currentselect) {
486             $options = array('removepages', 'deleteversions');
487             $items = array();
488             foreach ($options as $opt) {
489                 $items[] = get_string($opt, 'wikicode');
490             }
491             $selectoptions = array();
492             foreach ($items as $key => $item) {
493                 $selectoptions[$key + 1] = $item;
494             }
```

```
495             $select = new single_select(new moodle_url('/mod/wikicode/admin.php', array
                    ('pageid' => $pageid)), 'option', $selectoptions, $currentselect);
496             $select->label = get_string('adminmenu', 'wikicode') . ': ';
497             return $this->output->container($this->output->render($select), 'midpad');
498         }
499 
500         /**
501          * Internal function - creates htmls structure suitable for YUI tree.
502          */
503         protected function htmllize_tree($tree, $dir) {
504             global $CFG;
505             $yuiconfig = array();
506             $yuiconfig['type'] = 'html';
507 
508             if (empty($dir['subdirs']) and empty($dir['files'])) {
509                 return '';
510             }
511             $result = '<ul>';
512             foreach ($dir['subdirs'] as $subdir) {
513                 $image = $this->output->pix_icon("f/folder", $subdir['dirname'], '
                        moodle', array('class'=>'icon'));
514                 $result .= '<li yuiConfig=\''.json_encode($yuiconfig).'\'><div>'.$image
                        .' '.s($subdir['dirname']).'</div> '.$this->htmllize_tree($tree,
                        $subdir).'</li>';
515             }
516             foreach ($dir['files'] as $file) {
517                 $url = file_encode_url("$CFG->wwwroot/pluginfile.php", '/'.$tree->
                        context->id.'/mod_wiki/attachments/' . $tree->subwiki->id . '/'.
                        $file->get_filepath() . $file->get_filename(), true);
518                 $filename = $file->get_filename();
519                 $icon = mimeinfo("icon", $filename);
520                 $image = $this->output->pix_icon("f/$icon", $filename, 'moodle', array(
                        'class'=>'icon'));
521                 $result .= '<li yuiConfig=\''.json_encode($yuiconfig).'\'><div>'.$image
                        .' '.html_writer::link($url, $filename).'</div></li>';
522             }
523             $result .= '</ul>';
524 
525             return $result;
526         }
527     }
528 
529     class wikicode_files_tree implements renderable {
530         public $context;
531         public $dir;
532         public $subwiki;
533         public function __construct($context, $subwiki) {
534             $fs = get_file_storage();
535             $this->context = $context;
536             $this->subwiki = $subwiki;
537             $this->dir = $fs->get_area_tree($context->id, 'mod_wikicode', 'attachments'
                    , $subwiki->id);
538         }
539     }
```

3.2.8. restoreversion.php

```
<?php

require_once('../../config.php');
require_once($CFG->dirroot . '/mod/wikicode/lib.php');
require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
require_once($CFG->dirroot . '/mod/wikicode/pagelib.php');

$pageid = required_param('pageid', PARAM_INT);
$versionid = required_param('versionid', PARAM_INT);
$confirm = optional_param('confirm', 0, PARAM_BOOL);

if (!$page = wikicode_get_page($pageid)) {
    print_error('incorrectpageid', 'wikicode');
}

if (!$subwiki = wikicode_get_subwiki($page->subwikiid)) {
    print_error('incorrectsubwikiid', 'wikicode');
}

if (!$wiki = wikicode_get_wiki($subwiki->wikiid)) {
    print_error('incorrectwikiid', 'wikicode');
}

if (!$cm = get_coursemodule_from_instance('wikicode', $wiki->id)) {
    print_error('invalidcoursemodule');
}

$course = $DB->get_record('course', array('id' => $cm->course), '*', MUST_EXIST);

require_login($course->id, true, $cm);

add_to_log($course->id, "restore", "restore", "view.php?id=$cm->id", "$wiki->id");

if ($confirm) {
    if (!confirm_sesskey()) {
        print_error(get_string('invalidsesskey', 'wikicode'));
    }
    $wikipage = new page_wikicode_confirmrestore($wiki, $subwiki, $cm);
    $wikipage->set_page($page);
    $wikipage->set_versionid($versionid);

} else {

    $wikipage = new page_wikicode_restoreversion($wiki, $subwiki, $cm);
    $wikipage->set_page($page);
    $wikipage->set_versionid($versionid);

```

```
}

$wikipage->print_header();
$wikipage->print_content();

$wikipage->print_footer();
```

3.2.9. search.php

```
<?php

require_once('../../config.php');
require_once($CFG->dirroot . '/mod/wikicode/lib.php');
require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
require_once($CFG->dirroot . '/mod/wikicode/pagelib.php');

$search = optional_param('searchstring', null, PARAM_ALPHANUMEXT);
$courseid = optional_param('courseid', 0, PARAM_INT);
$searchcontent = optional_param('searchwikicontent', 0, PARAM_INT);
$cmid = optional_param('cmid', 0, PARAM_INT);

if (!$course = $DB->get_record('course', array('id' => $courseid))) {
    print_error('invalidcourseid');
}
if (!$cm = get_coursemodule_from_id('wikicode', $cmid)) {
    print_error('invalidcoursemodule');
}

require_login($course, true, $cm);

// @TODO: Fix call to wikicode_get_subwiki_by_group
if (!$gid = groups_get_activity_group($cm)) {
    $gid = 0;
}
if (!$subwiki = wikicode_get_subwiki_by_group($cm->instance, $gid)) {
    return false;
}
if (!$wiki = wikicode_get_wiki($subwiki->wikiid)) {
    print_error('incorrectwikiid', 'wikicode');
}

$wikipage = new page_wikicode_search($wiki, $subwiki, $cm);

$wikipage->set_search_string($search, $searchcontent);

$wikipage->set_title(get_string('search'));

$wikipage->print_header();

$wikipage->print_content();

```

```
43 $wikipage->print_footer();
```

Capítulo 4

Módulo para Moodle 1.x

4.1. Front-End

4.1.1. wikichat.php

```
<?php

require_once($CFG->dirroot.'/lib/formslib.php');
require_once($CFG->dirroot.'/lib/form/text.php');

class MoodleQuickForm_Wikichat extends MoodleQuickForm_text {

    private $itemid;

    function MoodleQuickForm_Wikichat($elementName = null, $elementLabel = null,
        $attributes = null) {
                $this->itemid = $attributes['itemid'];
    }

    function setWikiFormat($wikiformat) {
        $this->wikiformat = $wikiformat;
    }

    function toHtml() {
        $chat = $this->getChat();

        return $chat;
    }

    private function getChat() {
        global $PAGE, $OUTPUT, $CFG, $USER;

            $itemid = $this->itemid;

```

```
29              $html = "<link type=\"text/css\" rel=\"stylesheet\" href=\"chat/style.
                    css\" />";
30
31              $html .= "<div id=\"wrapper\">";
32              $html .= "<div id=\"menu\">";
33              $html .= "  <div style=\"clear:both\"></div>";
34              $html .= "</div>";
35              $html .= "<div id=\"chatbox\">";
36
37              if(file_exists("chat/logs/log_".$itemid.".html") && filesize("chat/logs
                    /log_".$itemid.".html") > 0){
38
39                      $handle = fopen("chat/logs/log_".$itemid.".html", "r");
40                      $contents = fread($handle, filesize("chat/logs/log_".
                            $itemid.".html"));
41                      fclose($handle);
42
43                      $html .= $contents;
44              }
45
46                  $html .= "</div>
47                  ";
48
49                  $html .= "      <input name=\"usermsg\" type=\"text\" id=\"usermsg
                        \" />
50                  ";
51                  $html .= "      <input name=\"submitmsg\" type=\"submit\"  id=\"
                        submitmsg\" value=\"Send\" />
52                  ";
53
54                  $html .= "</div>
55                  ";
56
57                  /*$html .= "<script type=\"text/javascript\" src=\"http://ajax.
                        googleapis.com/ajax/libs/jquery/1.3/jquery.min.js\"></script>
58                  ";*/
59                  $html .= "<script type=\"text/javascript\">
60                  ";
61                  // jQuery Document
62                  $html .= "$(document).ready(function(){
63                          ";
64
65                  //If user submits the form
66                  $html .=        "$(\"#submitmsg\").click(function(){
67                                  ";
68                  $html .=                "var clientmsg = $(\"#usermsg\").val();
69                  ";
70                  $html .=                "$.post(\"chat/post.php\", {text: clientmsg
                        , itemid: \"".$itemid."\", user: \"".$USER->username."\"});
71                  ";
72                  $html .=                "$(\"#usermsg\").attr(\"value\", \"\");
73                  ";
74                  $html .=                "return false;
75                  ";
```

```
76                  $html .=        "});
77                  ";
78
79                  //Load the file containing the chat log
80                  $html .=        "function loadLog(){
81                  ";
82                  $html .=                "var oldscrollHeight = $(\"#chatbox\").attr
                        (\"scrollHeight\") - 20;
83                  ";
84                  $html .=                "$.ajax({
85                          ";
86                  $html .=                        "url: \"chat/logs/log_".$itemid.".
                        html\",
87                  ";
88                  $html .=                        "cache: false,
89                  ";
90                  $html .=                        "success: function(html){
91                              ";
92                  $html .=                                "$(\"#chatbox\").html(html)
                        ;
93                  ";              //Insert chat log into the #chatbox div
94                  $html .=                                "var newscrollHeight = $
                        (\"#chatbox\").attr(\"scrollHeight\") - 20;
95                  ";
96                  $html .=                                "if(newscrollHeight >
                        oldscrollHeight){
97                          ";
98                  $html .=                                        "$(\"#chatbox\").
                        animate({ scrollTop: newscrollHeight }, 'normal');
99                  "; //Autoscroll to bottom of div
100                 $html .=                                "}
101                 ";
102                 $html .=                        "},
103                 ";
104                 $html .=                "});
105                 ";
106                 $html .=        "}
107                 ";
108                 $html .=        "setInterval (loadLog, 2500);
109                 ";      //Reload file every 2.5 seconds
110                 $html .= "});
111                 ";
112                 $html .= "</script>
113                 ";
114
115                 return $html;
116     }
117
118 }
119
120 //register wikieditor
121 MoodleQuickForm::registerElementType('wikicodechat', $CFG->dirroot."/mod/wikicode/
        editors/wikichat.php", 'MoodleQuickForm_Wikichat');
```

4.1.2. wikieditor.php

```
<?php

require_once($CFG->dirroot.'/lib/formslib.php');
require_once($CFG->dirroot.'/lib/form/textarea.php');

class MoodleQuickForm_Wikieditor extends MoodleQuickForm_textarea {

    private $files;

    function MoodleQuickForm_Wikieditor($elementName = null, $elementLabel = null,
        $attributes = null) {
        if (isset($attributes['Wiki_format'])) {
            $this->wikiformat = $attributes['Wiki_format'];
            unset($attributes['Wiki_format']);
        }
        if (isset($attributes['files'])) {
            $this->files = $attributes['files'];
            unset($attributes['files']);
        }

        parent::MoodleQuickForm_textarea($elementName, $elementLabel, $attributes);
    }

    function setWikiFormat($wikiformat) {
        $this->wikiformat = $wikiformat;
    }

    function toHtml() {
        $textarea = parent::toHtml();

        return $this->{
            $this->wikiformat."Editor"}
            ($textarea);
    }

    function creoleEditor($textarea) {
        return $this->printWikiEditor($textarea);
    }

    function nwikiEditor($textarea) {
        return $this->printWikiEditor($textarea);
    }

        function wcodeEditor($textarea) {
                global $OUTPUT;

        $textarea = $OUTPUT->container_start().$textarea.$OUTPUT->container_end();
                $editor = $this->getEditor();

        return $textarea.$editor;

```

```
51          }
52
53      private function printWikiEditor($textarea) {
54          global $OUTPUT;
55
56          $textarea = $OUTPUT->container_start().$textarea.$OUTPUT->container_end();
57
58          $buttons = $this->getButtons();
59
60          return $buttons.$textarea;
61      }
62
63          private function getEditor() {
64                  global $PAGE, $OUTPUT, $CFG;
65
66                  $editor = $this->wikiformat;
67
68                  $PAGE->requires->js('/mod/wikicode/js/php.js');
69                  $PAGE->requires->js('/mod/wikicode/js/codemirror.js');
70
71
72                  $html .= "<script src=\"js/php.js\" type=\"text/javascript\"></
                        script>";
73                  $html .= "<script src=\"js/jquery.js\" type=\"text/javascript\"></
                        script>";
74                  $html .= "<script src=\"js/unlock.js\" type=\"text/javascript\"></
                        script>";
75
76                          $html .= "<link rel=\"stylesheet\" type=\"text/css\"/>";
77                          $html .= "<style type=\"text/css\">";
78                          $html .= " .CodeMirror-line-numbers {";
79                  $html .= "   width: 2.2em;";
80                  $html .= "  color: #aaa;";
81                  $html .= "  background-color: #eee;";
82                          $html .= "  text-align: right;";
83                          $html .= "  padding-right: .3em;";
84                          $html .= "  font-size: 10pt;";
85                          $html .= "  font-family: monospace;";
86                          $html .= "  padding-top: .5em;";
87                          $html .= "}";
88                          $html .= "</style>";
89
90                          $html .= "<script type=\"text/javascript\">";
91                          $html .=   "var editor = CodeMirror.fromTextArea('
                                id_newcontent', {";
92                          $html .=   "parserfile: \"parseC.js\",";
93                          $html .=   "stylesheet: \"css/Ccolors.css\",";
94                          $html .=   "path: \"js/\",";
95                          $html .=   "continuousScanning: 500,";
96                  $html .=   "height: \"450px\",";
97                          $html .=   "lineNumbers: true";
98                          $html .= "});";
99                          $html .="</script>";
100
```

```
101                         return $html;
102
103         }
104
105     private function getButtons() {
106         global $PAGE, $OUTPUT, $CFG;
107
108         $editor = $this->wikiformat;
109
110         $tag = $this->getTokens($editor, 'bold');
111         $Wiki_editor['bold'] = array('ed_bold.gif', get_string('wikiboldtext', '
                Wikicode'), $tag[0], $tag[1], get_string('wikiboldtext', 'Wikicode'));
112
113         $Wiki_editor['italic'] = array('ed_italic.gif', get_string('wikiitalictext'
                , 'Wikicode'), $tag[0], $tag[1], get_string('wikiitalictext', 'Wikicode
                '));
114
115         $Wiki_editor['image'] = array('ed_img.gif', get_string('wikiimage', '
                Wikicode'), $imagetag[0], $imagetag[1], get_string('wikiimage', '
                Wikicode'));
116
117         $Wiki_editor['internal'] = array('ed_internal.gif', get_string('
                wikiinternalurl', 'Wikicode'), $tag[0], $tag[1], get_string('
                wikiinternalurl', 'Wikicode'));
118
119         $Wiki_editor['external'] = array('ed_external.gif', get_string('
                wikiexternalurl', 'Wikicode'), $tag, "", get_string('wikiexternalurl',
                'Wikicode'));
120
121         $Wiki_editor['u_list'] = array('ed_ul.gif', get_string('wikiunorderedlist',
                 'Wikicode'), '\\n'.$tag[0], '', '');
122         $Wiki_editor['o_list'] = array('ed_ol.gif', get_string('wikiorderedlist', '
                Wikicode'), '\\n'.$tag[1], '', '');
123
124         $Wiki_editor['h1'] = array('ed_h1.gif', get_string('wikiheader', 'Wikicode'
                , 1), '\\n'.$tag.' ', ' '.$tag.'\\n', get_string('wikiheader', '
                Wikicode', 1));
125         $Wiki_editor['h2'] = array('ed_h2.gif', get_string('wikiheader', 'Wikicode'
                , 2), '\\n'.$tag.$tag.' ', ' '.$tag.$tag.'\\n', get_string('wikiheader'
                , 'Wikicode', 2));
126         $Wiki_editor['h3'] = array('ed_h3.gif', get_string('wikiheader', 'Wikicode'
                , 3), '\\n'.$tag.$tag.$tag.' ', ' '.$tag.$tag.$tag.'\\n', get_string('
                wikiheader', 'Wikicode', 3));
127
128         $Wiki_editor['hr'] = array('ed_hr.gif', get_string('wikihr', 'Wikicode'), '
                \\n'.$tag.'\\n', '', '');
129
130         $Wiki_editor['nowiki'] = array('ed_nowiki.gif', get_string('wikinowikitext'
                , 'Wikicode'), $tag[0], $tag[1], get_string('wikinowikitext', 'Wikicode
                '));
131
132         $PAGE->requires->js('/mod/wikicode/editors/wikicode/buttons.js');
133
134         $html = '<div class="wikieditor-toolbar">';
```

```
135             foreach ($Wiki_editor as $button) {
136                 $html .= "<a href=\"javascript:insertTags";
137                 $html .= "('".$button[2]."','".$button[3]."','".$button[4]."');\">";
138                 $html .= html_writer::empty_tag('img', array('alt' => $button[1], 'src'
                        => $CFG->wwwroot . '/mod/wikicode/editors/wikicode/images/' .
                        $button[0]));
139                 $html .= "</a>";
140             }
141             $html .= "<select onchange=\"insertTags('{$imagetag[0]}', '{$imagetag[1]}',
                     this.value)\">";
142             $html .= "<option value='" . s(get_string('wikiimage', 'Wikicode')) . "'>"
                    . get_string('insertimage', 'Wikicode') . '</option>';
143             foreach ($this->files as $filename) {
144                 $html .= "<option value='".s($filename)."'>";
145                 $html .= $filename;
146                 $html .= '</option>';
147             }
148             $html .= '</select>';
149             $html .= $OUTPUT->help_icon('insertimage', 'Wikicode');
150             $html .= '</div>';
151 
152             return $html;
153         }
154 
155         private function getTokens($format, $token) {
156             $tokens = Wikicode_parser_get_token($format, $token);
157 
158             if (is_array($tokens)) {
159                 foreach ($tokens as & $t) {
160                     $this->escapeToken($t);
161                 }
162             } else {
163                 $this->escapeToken($tokens);
164             }
165 
166             return $tokens;
167         }
168 
169         private function escapeToken(&$token) {
170             $token = urlencode(str_replace("'", "\'", $token));
171         }
172     }
173 
174     //register wikieditor
175     MoodleQuickForm::registerElementType('wikicodeeditor', $CFG->dirroot."/mod/wikicode
        /editors/wikieditor.php", 'MoodleQuickForm_Wikieditor');
```

4.2. Back-End PHP

4.2.1. chat/post.php

```
<?php
        $text = $_POST['text'];

        if (file_exists("logs/log_".$_POST['itemid'].".html")) {
                $fp = fopen("logs/log_".$_POST['itemid'].".html", 'a');
        }
        else {
                $fp = fopen("logs/log_".$_POST['itemid'].".html", 'w');
        }
        fwrite($fp, "<div class='msgln'>(".date("g:i A").") <b>".$_POST['user']."</
            b>: ".stripslashes(htmlspecialchars($text))."<br></div>");
        fclose($fp);
?>
```

4.2.2. editorlib.php

```
<?php

/**
 * @author Antonio J. Gonzalez
 * @package mod-wikicode
 * @license http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
 */
require_once('../../config.php');
require_once($CFG->dirroot . '/mod/wikicode/editorlibfn.php');

$editorCode = $_GET["codigo"];
$position = $_GET["position"];
$pageid = $_GET["pageid"];
$mode = $_GET["mode"];
$del = $_GET["del"];
$char_del = $_GET["char_del"];

if ($char_del == '}' || $char_del == '{') {
        $mode = 2;
}

switch ($mode) {
        case 0:
                $return["code"] = "test";
                break;
    case 1: //Procedure when user push a key into the editor
        $return = keyPress($editorCode, $position, $pageid, $del);
        break;
        case 2: //Procedure when user remove a need-part
```

```
30              $return = removePairKey($editorCode, $position, $pageid, $char_del);
31              break;
32          case 3: //Unlock function
33              $return = unlockFunctionDB($editorCode, $pageid);
34              break;
35  }
36
37  echo json_encode($return);
38
39  ?>
```

4.2.3. editorlibfn.php

```
1   <?php
2
3   /**
4    * @author Antonio J. Gonzalez
5    * @package mod-wikicode
6    * @license http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
7    */
8
9   require_once('../../config.php');
10  require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
11
12  function salto_linea() {echo "<br>";}
13  function mostrar_variable($variable, $titulo) {echo $titulo.": ".$variable;
        salto_linea(); }
14  function escribir_fichero($cadena) {$file = fopen("fichero_prueba.txt","a"); fputs(
        $file, $cadena); fclose($file); }
15
16  /**
17   * @param string $codigo
18   *
19   * @return string New code
20   */
21  function deleteEnterRepeat($codigo) {
22
23          $codigo = str_replace(chr(13) . chr(10), chr(10), $codigo);
24
25          $return = "";
26
27          for($i=0; $i<strlen($codigo); $i++) {
28                  if (substr($codigo, $i - 1, 1) != chr(10) || substr($codigo, $i, 1)
                         != chr(10) || substr($codigo, $i+1, 1) != chr(10) || substr(
                        $codigo, $i+2, 1) != chr(10)) {
29                      $return = $return . substr($codigo, $i, 1);
30                  }
31          }
32
33          $return = str_replace(chr(10), chr(13) . chr(10), $return);
34
```

```
35          return $return;
36  }
37
38  /**
39   * Return function name in a determinate cursor position
40   *
41   * @param string $codigo Full source code
42   * @param int $posicion Character Position
43   */
44  function getFunction($codigo, $posicion) {
45
46          //First. Level of the character.
47          $level = getLevel($codigo, $posicion);
48          $help_cursor = $posicion;
49
50          //When level=0, cursor is out any function. Get next function.
51          if ($level == 0) {
52                  while ($level == 0 && $help_cursor <= strlen($codigo)) {
53                          $help_cursor++;
54                          $level = getLevel($codigo, $help_cursor);
55                  }
56          }
57
58          if ($level == 0) {return FALSE;} //There isnt any function in the code
59
60          //Function Start
61          while (getLevel($codigo, $help_cursor) > 0) {
62
63              $cursor_lu = strrpos(substr($codigo, 0, $help_cursor), "{");
64                  $cursor_ld = strrpos(substr($codigo, 0, $help_cursor), "}");
65
66                  if (!$cursor_lu) {$cursor_lu=0;}
67                  if (!$cursor_ld) {$cursor_ld=0;}
68
69                  if ($cursor_lu > $cursor_ld) {
70                          $help_cursor = $cursor_lu;
71                  }
72                  else {
73                          $help_cursor = $cursor_ld;
74                  }
75
76          }
77
78          //Get function name
79          $end_cursor = strrpos(substr($codigo, 0, $help_cursor),"(");
80          while (substr($codigo,$end_cursor-1,1) == " ") {
81                  $end_cursor = $end_cursor - 1;
82          }
83
84          $init_cursor = strrpos(substr($codigo, 0, $end_cursor)," ");
85          $init_cursor_aux = strrpos(substr($codigo, 0, $end_cursor),chr(10));
86
87          if ($init_cursor_aux > $init_cursor) {$init_cursor = $init_cursor_aux;} //
                When main declaration have not object type
```

```
88
89          $function_name = substr($codigo, $init_cursor, $end_cursor-$init_cursor);
90
91          return trim($function_name);
92  }
93
94  /**
95   * Return level indexation in a determinate cursor position
96   *
97   * @param string $codigo Full source code
98   * @param int $posicion Character Position
99   */
100 function getLevel($codigo, $posicion) {
101
102         $subcodigo = substr($codigo, 0, $posicion);
103
104         $levelup = substr_count($subcodigo,"{");
105         $leveldown = substr_count($subcodigo,"}");
106
107         $level = $levelup - $leveldown;
108
109         return $level;
110 }
111
112 /**
113  * Return True is a determinated function is locked.
114  *
115  * @param string $codigo Full source code
116  * @param string $function Function name
117  */
118 function isLocked($codigo, $function) {
119
120         //Starting code in main function, ignoring declarations
121         $pos_main = strpos(strtolower($codigo),"main");
122         $subcodigo = substr($codigo, $pos_main);
123
124         //Special case: Main function
125         if (strtolower($function) == "main") {
126
127                 $lock = strpos(substr($codigo, 0, $pos_main), "<!");
128
129                 if (is_numeric($lock)) {
130                         return TRUE;
131                 }
132                 return FALSE;
133         }
134
135         //Get position of function start
136         $end_cursor = strpos($subcodigo, $function);
137         $level = getLevel($codigo, $pos_main + $end_cursor);
138
139         while ($level > 0) {
140                 $end_cursor = strpos($subcodigo, $function, $end_cursor + 1);
141                 $level = getLevel($codigo, $pos_main + $end_cursor);
```

```
142         }
143
144         $init_cursor = strrpos(substr($subcodigo,0,$end_cursor),"}");
145         $subcadena = substr($subcodigo, $init_cursor, $end_cursor - $init_cursor);
146
147         //Search the locking-tag
148         $lock = strpos($subcadena, "<!");
149
150         if (is_numeric($lock)) {
151                 return TRUE;
152         }
153
154         return FALSE;
155 }
156
157 /**
158  * Return True is a determinated function is locked by the user.
159  *
160  * @param string $codigo Full source code
161  * @param string $function Function name
162  */
163 function isLockedByUser($codigo, $function) {
164
165         global $USER;
166
167         //Starting code in main function, ignoring declarations
168         $pos_main = strpos(strtolower($codigo),"main");
169         $subcodigo = substr($codigo, $pos_main);
170
171         //Special case: Main function
172         if (strtolower($function) == "main") {
173
174                 $lock = strpos(substr($codigo, 0, $pos_main), "<!".$USER->username)
                        ;
175
176                 if (is_numeric($lock)) {
177                         return TRUE;
178                 }
179                 return FALSE;
180         }
181
182         //Get position of function start
183         $end_cursor = strpos($subcodigo, $function);
184         $level = getLevel($codigo, $pos_main + $end_cursor);
185
186         while ($level > 0) {
187                 $end_cursor = strpos($subcodigo, $function, $end_cursor + 1);
188                 $level = getLevel($codigo, $pos_main + $end_cursor);
189         }
190
191         $init_cursor = strrpos(substr($subcodigo,0,$end_cursor),"}");
192         $subcadena = substr($subcodigo, $init_cursor, $end_cursor - $init_cursor);
193
194         //Search the locking-tag
```

```
195         $lock = strpos($subcadena, "<!".$USER->username);
196
197         if (is_numeric($lock)) {
198                 return TRUE;
199         }
200
201         return FALSE;
202 }
203
204 /**
205  * Lock a determinate function into a source-code
206  *
207  * @param string $codigo Full source-code
208  * @param string $function Function name
209  * @param string $user User who is blocking function
210  * @param int $pos_end Optional Param. Force cursor position end_lock.
211  *
212  * @return Source-code with new lock tag. False if is not possible.
213  */
214 function lockFunction($codigo, $function, $user, $pos_end="") {
215
216         $obviar=array(chr(10), chr(13), chr(32));
217
218         if (isLocked($codigo, $function)) {
219                 return FALSE;
220         }
221
222         if (strtolower($function) == 'main') {
223
224                 $init_cursor = 0;
225                 $pos_main = 0;
226
227                 while (in_array(substr($codigo,$init_cursor, 1),$obviar) ) {
228                         $init_cursor++;
229                 }
230
231                 $end_cursor = strpos($codigo, "}", $init_cursor);
232                 if ($end_cursor==FALSE) {$end_cursor=strlen($codigo);}
233
234                 $level = getLevel($codigo, $end_cursor);
235
236                 while ($level > 0) {
237                         $end_cursor = strpos($codigo, "}", $end_cursor) + 1;
238                     $level = getLevel($codigo, $end_cursor);
239                 }
240
241         }
242         else
243         {
244                 //Starting code in main function, ignoring declarations
245                 $pos_main = strpos(strtolower($codigo),"main");
246                 $subcodigo = substr($codigo, $pos_main);
247
248             //Get position of function start
```

```
249                 $end_cursor = strpos($subcodigo, $function);
250                 $level = getLevel($codigo, $pos_main + $end_cursor);
251
252                 while ($level > 0) {
253                         $end_cursor = strpos($subcodigo, $function, $end_cursor +
                                1);
254                         $level = getLevel($codigo, $pos_main + $end_cursor);
255                 }
256
257                 $init_cursor = strrpos(substr($subcodigo,0,$end_cursor),"}");
258
259                 if (substr($codigo,$pos_main + $init_cursor + 1, 4) == "</!>") {
                        $init_cursor = $init_cursor + 4;}
260
261                 while (in_array(substr($codigo,$pos_main + $init_cursor + 1, 1),
                        $obviar) ) {
262                         $init_cursor++;
263                 }
264
265                 $end_cursor = $pos_main + $end_cursor;
266                 $end_cursor = strpos($codigo, "}", $end_cursor);
267                 if ($end_cursor==FALSE) {
268                         $end_cursor=strlen($codigo);
269                 } else {
270                         $end_cursor = $end_cursor + 1;
271                 }
272
273                 $level = getLevel($codigo, $end_cursor);
274
275                 while ($level > 0) {
276                         $end_cursor = strpos($codigo, "}", $end_cursor) + 1;
277                     $level = getLevel($codigo, $end_cursor);
278                 }
279
280                 $init_cursor++;
281
282         }
283
284     if ($pos_end != "") {$end_cursor = $pos_end;}
285
286         $return  = substr($codigo, 0, $init_cursor + $pos_main);
287         $return .= "<!";
288         $return .= $user;
289         $return .= ">";
290         $return .= substr($codigo, $init_cursor + $pos_main, $end_cursor -
                $init_cursor - $pos_main);
291         $return .= "</!>";
292         $return .= substr($codigo, $end_cursor);
293
294         return $return;
295 }
296
297 /**
298  * Unlock a determinate function into a source-code
```

```
299     *
300     * @param string $codigo Full source-code
301     * @param string $function Function name
302     *
303     * @return string Source-code without lock tag. False if is not locked.
304     */
305    function unlockFunction($codigo, $function) {
306
307            if (!isLocked($codigo, $function)) {
308                    return FALSE;
309            }
310
311            $pos_main = strpos(strtolower($codigo), "main");
312
313            if (strtolower($function) == "main") {
314                    $init_start = strrpos(substr($codigo, 0, $pos_main),"<!");
315                    $end_start = strpos($codigo,">",$init_start);
316                    $init_ending = strpos($codigo,"</!>",$pos_main);
317            }
318            else {
319                    $subcodigo = substr($codigo, $pos_main);
320
321                    $init_start = strpos($subcodigo, $function);
322                    $level = getLevel($codigo, $pos_main + $init_start);
323
324                    while ($level > 0) {
325                            $init_start = strpos($subcodigo, $function, $init_start +
                                    1);
326                            $level = getLevel($codigo, $pos_main + $init_start);
327                    }
328
329                    $init_start = strrpos(substr($codigo, 0, $pos_main + $init_start),
                            "<!");
330                    $end_start = strpos($codigo,">",$init_start);
331                    $init_ending = strpos($codigo,"</!>", $end_start);
332            }
333
334            $return  = substr($codigo, 0, $init_start);
335            $return .= substr($codigo, $end_start + 1, $init_ending - $end_start - 1);
336            $return .= substr($codigo, $init_ending + 4);
337
338            return $return;
339    }
340
341    /**
342     * Returned all functions that are being locked by a determined user
343     *
344     * @param string $codigo Full source-code
345     * @param string $user Username
346     *
347     * @return array Function names
348     */
349    function getLocksByUser($codigo, $user) {
350            $functions = array();
```

```
351
352         $lock = strpos($codigo, "<!".$user.">");
353
354         if (is_bool($lock)) {return false;}
355
356         while (is_numeric($lock)) {
357                 array_push($functions, getFunction($codigo, $lock));
358                 $lock = strpos($codigo, "<!".$user.">", $lock + 1);
359         }
360
361         return $functions;
362 }
363
364 /**
365  * Return init char-position of a function locked
366  *
367  * @param string $codigo Full source-code
368  * @param string $function Function name
369  *
370  * @return int Character position
371  */
372 function getInitFunctionLock($codigo, $function) {
373
374         if (!isLocked($codigo, $function)) {return false;}
375
376         if (strtolower($function) == 'main') {
377                 $pos = strpos($codigo, "<!");
378         }
379         else {
380                 $pos_main = strpos(strtolower($codigo), "main");
381                 $subcode = substr($codigo, $pos_main);
382
383                 $pos = strpos($subcode, $function);
384                 $level = getLevel($codigo, $pos + $pos_main);
385
386                 while ($level > 0) {
387                         $pos = strpos($subcode, $function, $pos + 1);
388                         $level = getLevel($codigo, $pos + $pos_main);
389                 }
390                 $pos = strrpos(substr($codigo, 0,  $pos_main + $pos), "<!");
391
392         }
393
394         return $pos;
395 }
396
397 /**
398  * Return ending char-position of a function locked
399  *
400  * @param string $codigo Full source-code
401  * @param string $function Function name
402  *
403  * @return int Character position
404  */
```

```
405 function getEndFunctionLock($codigo, $function) {
406
407         if (!isLocked($codigo, $function)) {return false;}
408
409         if (strtolower($function) == 'main') {
410                 $pos = strpos($codigo, "</!>");
411         }
412         else {
413                 $pos_main = strpos(strtolower($codigo), "main");
414
415                 $subcode = substr($codigo, $pos_main);
416
417                 $pos = strpos($subcode, $function);
418                 $level = getLevel($codigo, $pos + $pos_main);
419
420                 while ($level > 0) {
421                         $pos = strpos($subcode, $function, $pos + 1);
422                         $level = getLevel($codigo, $pos + $pos_main);
423                 }
424
425                 $subcodigo = substr($codigo,$pos_main + $pos);
426                 $pos = strpos($subcodigo, "</!>") + $pos_main + $pos;
427         }
428
429         return $pos + 4;
430 }
431
432 /**
433  * Return locked function with the tag-lock delimiters
434  *
435  * @param string $codigo Full source-code
436  * @param string $function Function name
437  *
438  * @return string Function source-code
439  */
440 function getFunctionLock($codigo, $function) {
441
442         if (!isLocked($codigo, $function)) {return false;}
443
444         return substr($codigo, getInitFunctionLock($codigo, $function),
             getEndFunctionLock($codigo, $function) - getInitFunctionLock($codigo,
             $function));
445 }
446
447 /**
448  * Return locked function with the tag-lock delimiters
449  *
450  * @param string $codigo Full source-code
451  * @param string $function Function name
452  * @param string $newfunction New source-code of the function
453  *
454  * @return string Function source-code
455  */
456 function changeFunctionLock($codigo, $function, $newfunction) {
```

```
457
458         if (!isLocked($codigo, $function)) {return false;}
459
460         $init_function = getInitFunctionLock($codigo, $function);
461         $end_function = getEndFunctionLock($codigo, $function);
462
463         $return  = substr($codigo, 0, $init_function);
464         $return .= $newfunction;
465         $return .= substr($codigo, $end_function);
466
467         return $return;
468
469 }
470
471 /**********************
472  * DATABASE FUNCTIONS *
473  **********************/
474
475 /**
476  * Return original source-code from DB
477  *
478  * @param int $pageid
479  *
480  * @return string Original source-code
481  */
482 function getCodigofromDB($pageid) {
483
484         $version = wikicode_get_current_version($pageid);
485
486         $resultado = $version->content;
487
488         return $resultado;
489 }
490
491 /**
492  * Save new source-code into DB
493  *
494  * @param int $pageid
495  * @param string $newcode
496  */
497 function saveCodigoToDB($pageid, $newcode) {
498
499         global $DB;
500
501         $page = wikicode_get_page($pageid);
502         $page->cachedcontent = $newcode;
503
504         $page->timerendered = time();
505         if ($page->timestartedit == 0) {
506                 $page->timestartedit = time();
507         }
508         $page->timeendedit = time();
509         $DB->update_record('wikicode_pages', $page);
510
```

```
511         $version = wikicode_get_current_version($pageid);
512         $version->content = $newcode;
513         $DB->update_record('wikicode_versions', $version);
514 
515         return TRUE;
516 }
517 
518 /******************
519  *** PROCEDURES ***
520  ******************/
521 
522 /**
523  * Procedure to exec when push a key into edior
524  *
525  * @param string $newcode
526  * @param int $position
527  * @param int $pageid
528  * @param string $del
529  *
530  * @return object
531  */
532 function keyPress($newcode, $position, $pageid, $del) {
533         global $USER;
534 
535         $return = array();
536 
537         $code = $newcode;
538         $position++;
539 
540         if ($del==1) {$position = $position - 2;}
541 
542         //In first place, change the double quotes for simple quotes.
543         $code = str_replace("\"", "'", $code);
544     $code = str_replace("\\", "", $code);
545 
546         //Get the original code
547         $originalcode = getCodigofromDB($pageid);
548         $dbcode = $originalcode;
549         $originalcode = str_replace("\"", "'", $originalcode);
550     $originalcode = str_replace("\\", "", $originalcode);
551 
552         //Get the function name which are modifying
553         $function = getFunction($code, $position);
554         escribir_fichero($function);
555 
556         //There is not any function in the code
557         if (is_bool($function)) {
558                 $newcodelock = $code;
559                 $endcode = $code;
560 
561                 $last_function = strrpos($code, "}");
562 
563                 if (!is_bool($last_function)) {
564                         if ($del==0) {
```

```
565                                 //$endcode = $originalcode . substr($code,
                                        $position - 1, 1);
566                         }
567                         else {
568                                 $endcode = substr($originalcode, 0, strlen(
                                        $originalcode) - 1);
569                         }
570                 }
571
572                 escribir_fichero($endcode);
573
574                 $return["position"] = $position;
575         }
576         elseif (isLocked($originalcode, $function) && !isLockedByUser($originalcode
                , $function)) //Function is blocked by other user
577         {
578                 $endcode = $originalcode;
579         }
580         else {
581                 //Locking function into the original source-code if this is not
                        locked
582                 if (!isLocked($originalcode, $function)){
583                         $originalcode = lockFunction($originalcode, $function,
                                $USER->username);
584                 }
585
586                 //Insert lock tags into the new source-code
587                 $newcodeLock = lockFunction($code, $function, $USER->username);
588
589                 //Change original function by the new function
590                 $newfunction = getFunctionLock($newcodeLock, $function);
591                 $endcode = changeFunctionLock($originalcode, $function,
                        $newfunction);
592         }
593
594         //Saving into DB
595         $returncode = str_replace("'","\"", $endcode);
596
597         if (trim($returncode) == "" && strlen($dbcode) > 1) {$returncode = $dbcode
                ;}
598         $returncode = deleteEnterRepeat($returncode);
599
600         if (saveCodigoToDB($pageid, $returncode)) {
601                 $return["code"] = wikicode_remove_tags_owner($returncode);
602         }
603
604         //Get new cursor position
605         if (!is_bool($function)) {
606                 $lenght_user = strlen($USER->username) + 3;
607                 $pos_inside = $position - $lenght_user - getInitFunctionLock(
                        $newcodeLock, $function);
608                 $str_search = deleteEnterRepeat(substr($newcodeLock,
                        getInitFunctionLock($newcodeLock, $function) + $lenght_user,
                        $pos_inside + $lenght_user));
```

```
609                 $newposition = strpos($return["code"], $str_search) + strlen(
                        $str_search);
610                 $return["position"] = $newposition;
611         } else {
612                 $return["position"] = $position;
613         }
614
615         return $return;
616 }
617
618 /**
619  * Procedure to exec when user removed a need-part
620  *
621  * @param string $newcode
622  * @param int $position
623  * @param int $pageid
624  * @param string $char_del
625  *
626  * @return object
627  */
628 function removePairKey($newcode, $position, $pageid, $char_del) {
629         global $USER;
630
631         $code = $newcode;
632         $position = $position - 1;
633
634         //In first place, change the double quotes for simple quotes.
635         $code = str_replace("\"", "'", $code);
636         $code = str_replace("\n", chr(10), $code);
637     $code = str_replace("\\", "", $code);
638
639         //Get the original code
640         $dbcode = getCodigofromDB($pageid);
641         $dbcode = str_replace("\"", "'", $dbcode);
642     $dbcode = str_replace("\\", "", $dbcode);
643
644         //Code before remove any character
645         $originalcode = substr($code, 0, $position) . $char_del . substr($code,
                $position);
646
647         if ($char_del == "{") {
648                 $positionlevel = $position;
649         } else {
650                 $positionlevel = $position + 1;
651         }
652
653         $level = getLevel($originalcode, $positionlevel);
654         $function = getFunction($originalcode, $position);
655
656         if ($level == 0) {
657                 if ($char_del == "{" && substr($originalcode, $position + 1, 1) ==
                        "}") {
658                         $newfunction = "";
```

```
659                 } else if ($char_del == "}" && substr($originalcode, $position -1,
                        1) == "{") {
660                     $newfunction = "";
661                 } else {
662                     //Insert lock tags into the new source-code
663                     $newcodeLock = lockFunction($originalcode, $function, $USER
                            ->username);
664                     $newfunction = getFunctionLock($newcodeLock, $function);
665                 }
666         } else {
667
668                 $found = 1;
669                 $pos_pair = 0;
670
671                 if ($char_del == "{") {
672                     for ($i=$position; $found != 0; $i++) {
673                         if (substr($originalcode, $i + 1, 1) == "{") {
674                             $found++;
675                         } else if (substr($originalcode, $i + 1, 1) == "}")
                                {
676                             $found--;
677                         }
678                         $pos_pair = $i + 1;
679                     }
680                 } else if ($char_del == "}") {
681                     for ($i=1; $found != 0; $i++) {
682                         if (substr($originalcode, $position - $i, 1) == "}"
                                ) {
683                             $found++;
684                         } else if (substr($originalcode, $position - $i, 1)
                                == "{") {
685                             $found--;
686                         }
687                         $pos_pair = $position - $i;
688                     }
689                 }
690                 if ($position < $pos_pair) {
691                     $newcode = substr($originalcode, 0, $position) . substr(
                            $originalcode, $position + 1, $pos_pair - $position -
                            1) . substr($originalcode, $pos_pair + 1);
692                 } else {
693                     $newcode = substr($originalcode, 0, $pos_pair) . substr(
                            $originalcode, $pos_pair + 1, $position - $pos_pair -
                            1) . substr($originalcode, $position + 1);
694                 }
695
696                 $newcodeLock = lockFunction($newcode, $function, $USER->username);
697                 $newfunction = getFunctionLock($newcodeLock, $function);
698
699         }
700
701         //Locking function into the original source-code if this is not locked
702         if (!isLocked($dbcode, $function)){
703                 $dbcode = lockFunction($dbcode, $function, $USER->username);
```

```
704         }
705
706         //Change original function by the new function
707         $endcode = changeFunctionLock($dbcode, $function, $newfunction);
708
709         //Saving into DB
710         $returncode = str_replace("'","\"", $endcode);
711         $returncode = deleteEnterRepeat($returncode);
712
713         if (saveCodigoToDB($pageid, $returncode)) {
714                 $return["code"] = wikicode_remove_tags_owner($returncode);
715         }
716
717         return $return;
718
719 }
720
721 /**
722  * Procedure to exec when user removed a need-part
723  *
724  * @param string $function
725  * @param int $pageid
726  *
727  * @return object
728  */
729 function unlockFunctionDB($function, $pageid) {
730
731         //Get the original code
732         $dbcode = getCodigofromDB($pageid);
733
734         //Unlock the function
735         $savecode = unlockFunction($dbcode, $function);
736
737         //Save newcode
738         if (saveCodigoToDB($pageid, $savecode)) {
739                 $return["code"] = wikicode_remove_tags_owner($savecode);
740         }
741
742         return $return;
743
744 }
745
746 ?>
```

4.2.4. unlock.php

```
1 <?php
2
3 /**
4  * @author Antonio J. Gonzalez
5  * @package mod-wikicode
```

```
6    * @license http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
7    */
8
9   global $USER, $DB;
10
11  $pageid = $_GET["pageid"];
12
13  require_once('../../config.php');
14  require_once($CFG->dirroot . '/mod/wikicode/locallib.php');
15  require_once($CFG->dirroot . '/mod/wikicode/editorlibfn.php');
16
17  $version = wikicode_get_current_version($pageid);
18  $codigo = $version->content;
19
20  $funciones = getLocksByUser($codigo, $USER->username);
21
22  echo '<link rel="stylesheet" type="text/css"/>
23        <style type="text/css">
24         div#checks {
25          text-align:left;
26                  margin:0 auto;
27                  padding:10px;
28                  background:#fff;
29                  height:auto;
30                  width:430px;
31                  border:1px solid #ACD8F0;
32                  overflow:auto;
33                  font-family:arial;
34                  font-size:9pt;
35        }
36        div#button {
37          text-align:left;
38                  margin:0 auto;
39                  padding:10px;
40                  background:#ACD8F0;
41                  height:auto;
42                  width:430px;
43                  border:1px solid #ACD8F0;
44                  overflow:auto;
45                  font-family:arial;
46                  font-size:9pt;
47        }
48        div#titulo {
49          text-align:left;
50                  margin:0 auto;
51                  padding:10px;
52                  background:#ACD8F0;
53                  height:auto;
54                  width:430px;
55                  border:1px solid #ACD8F0;
56                  overflow:auto;
57                  font-family:arial;
58                  font-size:10pt;
59        }
```

```
60        </style>';
61
62
63  echo '<div id="titulo"><h4>Funciones</h4></div>';
64  echo '<div id="checks">';
65
66  foreach ($funciones as $variable => $valor){
67   echo '
68  <input type="checkbox" name="'.$valor.'" value="'.$valor.'" />'.$valor.'</br>';
69  }
70
71  echo '</div>';
72  echo '<div id="button"><button type="button" class="unlock">Desbloquear</button>
        <button type="cancel" class="cancelUnlock">Cancelar</button></div>';
73
74  echo '<script src="js/jquery.js"></script>';
75  echo '<script src="js/php.js"></script>';
76  echo '<script src="js/unlock.js"></script>';
77
78
79  ?>
```

4.3. Back-End JS

4.3.1. codemirror.js

```
1   /* CodeMirror main module
2    *
3    * Implements the CodeMirror constructor and prototype, which take care
4    * of initializing the editor frame, and providing the outside interface.
5    */
6
7   // The CodeMirrorConfig object is used to specify a default
8   // configuration. If you specify such an object before loading this
9   // file, the values you put into it will override the defaults given
10  // below. You can also assign to it after loading.
11  var CodeMirrorConfig = window.CodeMirrorConfig || {};
12
13  var CodeMirror = (function(){
14    function setDefaults(object, defaults) {
15      for (var option in defaults) {
16        if (!object.hasOwnProperty(option))
17          object[option] = defaults[option];
18      }
19    }
20    function forEach(array, action) {
21      for (var i = 0; i < array.length; i++)
22        action(array[i]);
23    }
```

```
24
25    // These default options can be overridden by passing a set of
26    // options to a specific CodeMirror constructor. See manual.html for
27    // their meaning.
28    setDefaults(CodeMirrorConfig, {
29      stylesheet: "",
30      path: "",
31      parserfile: [],
32      basefiles: ["util.js", "stringstream.js", "select.js", "undo.js", "editor.js",
            "tokenize.js", "jquery.js", "lock.js", "php.js"],
33      iframeClass: null,
34      passDelay: 200,
35      passTime: 50,
36      lineNumberDelay: 200,
37      lineNumberTime: 50,
38      continuousScanning: false,
39      saveFunction: null,
40      onChange: null,
41      undoDepth: 50,
42      undoDelay: 800,
43      disableSpellcheck: true,
44      textWrapping: true,
45      readOnly: false,
46      width: "",
47      height: "",
48      autoMatchParens: false,
49      parserConfig: null,
50      tabMode: "indent", // or "spaces", "default", "shift"
51      reindentOnLoad: false,
52      activeTokens: null,
53      cursorActivity: null,
54      lineNumbers: false,
55      indentUnit: 2,
56      domain: null
57    });
58
59    function addLineNumberDiv(container) {
60      var nums = document.createElement("DIV"),
61          scroller = document.createElement("DIV");
62      nums.style.position = "absolute";
63      nums.style.height = "100%";
64      if (nums.style.setExpression) {
65        try {nums.style.setExpression("height", "this.previousSibling.offsetHeight +
              'px'");}
66        catch(e) {} // Seems to throw 'Not Implemented' on some IE8 versions
67      }
68      nums.style.top = "0px";
69      nums.style.overflow = "hidden";
70      container.appendChild(nums);
71      scroller.className = "CodeMirror-line-numbers";
72      nums.appendChild(scroller);
73      return nums;
74    }
75
```

```
76    function frameHTML(options) {
77      if (typeof options.parserfile == "string")
78        options.parserfile = [options.parserfile];
79      if (typeof options.stylesheet == "string")
80        options.stylesheet = [options.stylesheet];
81
82      var html = ["<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 Transitional//EN\" \"
            http://www.w3.org/TR/html4/loose.dtd\"><html><head>"];
83      // Hack to work around a bunch of IE8-specific problems.
84      html.push("<meta http-equiv=\"X-UA-Compatible\" content=\"IE=EmulateIE7\"/>");
85      forEach(options.stylesheet, function(file) {
86        html.push("<link rel=\"stylesheet\" type=\"text/css\" href=\"" + file +
              "\"/>");
87      });
88      forEach(options.basefiles.concat(options.parserfile), function(file) {
89        html.push("<script type=\"text/javascript\" src=\"" + options.path + file +
              "\"><" + "/script>");
90      });
91      html.push("</head><body style=\"border-width: 0;\" class=\"editbox\" spellcheck
            =\"" +
92                (options.disableSpellcheck ? "false" : "true") + "\"></body></html>")
                    ;
93      return html.join("");
94    }
95
96    var internetExplorer = document.selection && window.ActiveXObject && /MSIE/.test(
          navigator.userAgent);
97
98    function CodeMirror(place, options) {
99      // Backward compatibility for deprecated options.
100     if (options.dumbTabs) options.tabMode = "spaces";
101     else if (options.normalTab) options.tabMode = "default";
102
103     // Use passed options, if any, to override defaults.
104     this.options = options = options || {};
105     setDefaults(options, CodeMirrorConfig);
106
107     var frame = this.frame = document.createElement("IFRAME");
108     if (options.iframeClass) frame.className = options.iframeClass;
109     frame.frameBorder = 0;
110     frame.style.border = "0";
111     frame.style.width = '100%';
112     frame.style.height = '100%';
113     // display: block occasionally suppresses some Firefox bugs, so we
114     // always add it, redundant as it sounds.
115     frame.style.display = "block";
116     frame.id = "iframeid";
117
118     var div = this.wrapping = document.createElement("DIV");
119     div.style.position = "relative";
120     div.className = "CodeMirror-wrapping";
121     div.style.width = options.width;
122     div.style.height = options.height;
123     // This is used by Editor.reroutePasteEvent
```

```
124       var teHack = this.textareaHack = document.createElement("TEXTAREA");
125       div.appendChild(teHack);
126       teHack.style.position = "absolute";
127       teHack.style.left = "-10000px";
128       teHack.style.width = "10px";
129
130       //Hidden
131       var inHack = this.textareaInsert = document.createElement("TEXTAREA");
132       div.appendChild(inHack);
133       inHack.style.position = "absolute";
134       inHack.style.left = "-10000px";
135       inHack.style.width = "10px";
136       inHack.value = "1";
137
138       // Link back to this object, so that the editor can fetch options
139       // and add a reference to itself.
140       frame.CodeMirror = this;
141       if (options.domain && internetExplorer) {
142         this.html = frameHTML(options);
143         frame.src = "javascript:(function(){document.open();" +
144           (options.domain ? "document.domain=\"" + options.domain + "\";" : "") +
145           "document.write(window.frameElement.CodeMirror.html);document.close();})()
                ";
146       }
147       else {
148         frame.src = "javascript:false";
149       }
150
151       if (place.appendChild) place.appendChild(div);
152       else place(div);
153       div.appendChild(frame);
154       if (options.lineNumbers) this.lineNumbers = addLineNumberDiv(div);
155
156       this.win = frame.contentWindow;
157       if (!options.domain || !internetExplorer) {
158         this.win.document.open();
159         this.win.document.write(frameHTML(options));
160         this.win.document.close();
161       }
162     }
163
164     CodeMirror.prototype = {
165       init: function() {
166         if (this.options.initCallback) this.options.initCallback(this);
167         if (this.options.lineNumbers) this.activateLineNumbers();
168         if (this.options.reindentOnLoad) this.reindent();
169       },
170
171       getCode: function() {
172           return this.editor.getCode();
173       },
174       setCode: function(code) {this.editor.importCode(code);},
175       selection: function() {this.focusIfIE(); return this.editor.selectedText();},
176       reindent: function() {this.editor.reindent();},
```

```
177       reindentSelection: function() {this.focusIfIE(); this.editor.reindentSelection(
              null);},
178
179       focusIfIE: function() {
180         // in IE, a lot of selection-related functionality only works when the frame
                is focused
181         if (this.win.select.ie_selection) this.focus();
182       },
183       focus: function() {
184         this.win.focus();
185         if (this.editor.selectionSnapshot) // IE hack
186           this.win.select.setBookmark(this.win.document.body, this.editor.
                selectionSnapshot);
187       },
188       replaceSelection: function(text) {
189         this.focus();
190         this.editor.replaceSelection(text);
191         return true;
192       },
193       replaceChars: function(text, start, end) {
194         this.editor.replaceChars(text, start, end);
195       },
196       getSearchCursor: function(string, fromCursor, caseFold) {
197         return this.editor.getSearchCursor(string, fromCursor, caseFold);
198       },
199
200       undo: function() {this.editor.history.undo();},
201       redo: function() {this.editor.history.redo();},
202       historySize: function() {return this.editor.history.historySize();},
203       clearHistory: function() {this.editor.history.clear();},
204
205       grabKeys: function(callback, filter) {this.editor.grabKeys(callback, filter);},
206       ungrabKeys: function() {this.editor.ungrabKeys();},
207
208       setParser: function(name) {this.editor.setParser(name);},
209       setSpellcheck: function(on) {this.win.document.body.spellcheck = on;},
210       setStylesheet: function(names) {
211         if (typeof names === "string") names = [names];
212         var activeStylesheets = {};
213         var matchedNames = {};
214         var links = this.win.document.getElementsByTagName("link");
215         // Create hashes of active stylesheets and matched names.
216         // This is O(n^2) but n is expected to be very small.
217         for (var x = 0, link; link = links[x]; x++) {
218           if (link.rel.indexOf("stylesheet") !== -1) {
219             for (var y = 0; y < names.length; y++) {
220               var name = names[y];
221               if (link.href.substring(link.href.length - name.length) === name) {
222                 activeStylesheets[link.href] = true;
223                 matchedNames[name] = true;
224               }
225             }
226           }
227         }
```

```
228         // Activate the selected stylesheets and disable the rest.
229         for (var x = 0, link; link = links[x]; x++) {
230           if (link.rel.indexOf("stylesheet") !== -1) {
231             link.disabled = !(link.href in activeStylesheets);
232           }
233         }
234         // Create any new stylesheets.
235         for (var y = 0; y < names.length; y++) {
236           var name = names[y];
237           if (!(name in matchedNames)) {
238             var link = this.win.document.createElement("link");
239             link.rel = "stylesheet";
240             link.type = "text/css";
241             link.href = name;
242             this.win.document.getElementsByTagName('head')[0].appendChild(link);
243           }
244         }
245       },
246       setTextWrapping: function(on) {
247         if (on == this.options.textWrapping) return;
248         this.win.document.body.style.whiteSpace = on ? "" : "nowrap";
249         this.options.textWrapping = on;
250         if (this.lineNumbers) {
251           this.setLineNumbers(false);
252           this.setLineNumbers(true);
253         }
254       },
255       setIndentUnit: function(unit) {this.win.indentUnit = unit;},
256       setUndoDepth: function(depth) {this.editor.history.maxDepth = depth;},
257       setTabMode: function(mode) {this.options.tabMode = mode;},
258       setLineNumbers: function(on) {
259         if (on && !this.lineNumbers) {
260           this.lineNumbers = addLineNumberDiv(this.wrapping);
261           this.activateLineNumbers();
262         }
263         else if (!on && this.lineNumbers) {
264           this.wrapping.removeChild(this.lineNumbers);
265           this.wrapping.style.marginLeft = "";
266           this.lineNumbers = null;
267         }
268       },
269
270       cursorPosition: function(start) {this.focusIfIE(); return this.editor.
              cursorPosition(start);},
271       firstLine: function() {return this.editor.firstLine();},
272       lastLine: function() {return this.editor.lastLine();},
273       nextLine: function(line) {return this.editor.nextLine(line);},
274       prevLine: function(line) {return this.editor.prevLine(line);},
275       lineContent: function(line) {return this.editor.lineContent(line);},
276       setLineContent: function(line, content) {this.editor.setLineContent(line,
              content);},
277       removeLine: function(line){this.editor.removeLine(line);},
278       insertIntoLine: function(line, position, content) {this.editor.insertIntoLine(
              line, position, content);},
```

```
279       selectLines: function(startLine, startOffset, endLine, endOffset) {
280         this.win.focus();
281         this.editor.selectLines(startLine, startOffset, endLine, endOffset);
282       },
283       nthLine: function(n) {
284         var line = this.firstLine();
285         for (; n > 1 && line !== false; n--)
286           line = this.nextLine(line);
287         return line;
288       },
289       lineNumber: function(line) {
290         var num = 0;
291         while (line !== false) {
292           num++;
293           line = this.prevLine(line);
294         }
295         return num;
296       },
297       jumpToLine: function(line) {
298         if (typeof line == "number") line = this.nthLine(line);
299         this.selectLines(line, 0);
300         this.win.focus();
301       },
302       currentLine: function() { // Deprecated, but still there for backward
              compatibility
303         return this.lineNumber(this.cursorLine());
304       },
305       cursorLine: function() {
306         return this.cursorPosition().line;
307       },
308
309       activateLineNumbers: function() {
310         var frame = this.frame, win = frame.contentWindow, doc = win.document, body =
               doc.body,
311             nums = this.lineNumbers, scroller = nums.firstChild, self = this;
312         var barWidth = null;
313
314         function sizeBar() {
315           if (frame.offsetWidth == 0) return;
316           for (var root = frame; root.parentNode; root = root.parentNode);
317           if (!nums.parentNode || root != document || !win.Editor) {
318             // Clear event handlers (their nodes might already be collected, so try/
                  catch)
319             try{clear();}catch(e){}
320             clearInterval(sizeInterval);
321             return;
322           }
323
324           if (nums.offsetWidth != barWidth) {
325             barWidth = nums.offsetWidth;
326             nums.style.left = "-" + (frame.parentNode.style.marginLeft = barWidth + "
                  px");
327           }
328         }
```

```
329         function doScroll() {
330           nums.scrollTop = body.scrollTop || doc.documentElement.scrollTop || 0;
331         }
332         // Cleanup function, registered by nonWrapping and wrapping.
333         var clear = function(){};
334         sizeBar();
335         var sizeInterval = setInterval(sizeBar, 500);
336 
337         function nonWrapping() {
338           var nextNum = 1, pending;
339           function update() {
340             var target = 50 + Math.max(body.offsetHeight, Math.max(frame.offsetHeight
                  , body.scrollHeight || 0));
341             var endTime = new Date().getTime() + self.options.lineNumberTime;
342             while (scroller.offsetHeight < target && (!scroller.firstChild ||
                  scroller.offsetHeight)) {
343               scroller.appendChild(document.createElement("DIV"));
344               scroller.lastChild.innerHTML = nextNum++;
345               if (new Date().getTime() > endTime) {
346                 if (pending) clearTimeout(pending);
347                 pending = setTimeout(update, self.options.lineNumberDelay);
348                 break;
349               }
350             }
351             doScroll();
352           }
353           var onScroll = win.addEventHandler(win, "scroll", update, true),
354               onResize = win.addEventHandler(win, "resize", update, true);
355           clear = function(){onScroll(); onResize(); if (pending) clearTimeout(
                  pending);};
356           update();
357         }
358         function wrapping() {
359           var node, lineNum, next, pos;
360 
361           function addNum(n) {
362             if (!lineNum) lineNum = scroller.appendChild(document.createElement("DIV
                    "));
363             lineNum.innerHTML = n;
364             pos = lineNum.offsetHeight + lineNum.offsetTop;
365             lineNum = lineNum.nextSibling;
366           }
367           function work() {
368             if (!scroller.parentNode || scroller.parentNode != self.lineNumbers)
                    return;
369 
370             var endTime = new Date().getTime() + self.options.lineNumberTime;
371             while (node) {
372               addNum(next++);
373               for (; node && !win.isBR(node); node = node.nextSibling) {
374                 var bott = node.offsetTop + node.offsetHeight;
375                 while (scroller.offsetHeight && bott - 3 > pos) addNum(" ");
376               }
377               if (node) node = node.nextSibling;
```

```
378               if (new Date().getTime() > endTime) {
379                 pending = setTimeout(work, self.options.lineNumberDelay);
380                 return;
381               }
382             }
383             // While there are un-processed number DIVs, or the scroller is smaller
                    than the frame...
384             var target = 50 + Math.max(body.offsetHeight, Math.max(frame.offsetHeight
                    , body.scrollHeight || 0));
385             while (lineNum || (scroller.offsetHeight < target && (!scroller.
                    firstChild || scroller.offsetHeight)))
386               addNum(next++);
387             doScroll();
388           }
389           function start() {
390             doScroll();
391             node = body.firstChild;
392             lineNum = scroller.firstChild;
393             pos = 0;
394             next = 1;
395             work();
396           }
397
398           start();
399           var pending = null;
400           function update() {
401             if (pending) clearTimeout(pending);
402             if (self.editor.allClean()) start();
403             else pending = setTimeout(update, 200);
404           }
405           self.updateNumbers = update;
406           var onScroll = win.addEventHandler(win, "scroll", doScroll, true),
407               onResize = win.addEventHandler(win, "resize", update, true);
408           clear = function(){
409             if (pending) clearTimeout(pending);
410             if (self.updateNumbers == update) self.updateNumbers = null;
411             onScroll();
412             onResize();
413           };
414         }
415         (this.options.textWrapping ? wrapping : nonWrapping)();
416       }
417     };
418
419     CodeMirror.InvalidLineHandle = {toString: function(){return "CodeMirror.
            InvalidLineHandle";}};
420
421     CodeMirror.replace = function(element) {
422       if (typeof element == "string")
423         element = document.getElementById(element);
424       return function(newElement) {
425         element.parentNode.replaceChild(newElement, element);
426       };
427     };
```

```
428
429   CodeMirror.fromTextArea = function(area, options) {
430     if (typeof area == "string")
431       area = document.getElementById(area);
432
433     options = options || {};
434     if (area.style.width && options.width == null)
435       options.width = area.style.width;
436     if (area.style.height && options.height == null)
437       options.height = area.style.height;
438     if (options.content == null) options.content = area.value;
439
440     if (area.form) {
441       function updateField() {
442         area.value = mirror.getCode();
443       }
444       if (typeof area.form.addEventListener == "function")
445         area.form.addEventListener("submit", updateField, false);
446       else
447         area.form.attachEvent("onsubmit", updateField);
448     }
449
450     function insert(frame) {
451       if (area.nextSibling)
452         area.parentNode.insertBefore(frame, area.nextSibling);
453       else
454         area.parentNode.appendChild(frame);
455     }
456
457     area.style.display = "none";
458     var mirror = new CodeMirror(insert, options);
459     return mirror;
460   };
461   CodeMirror.isProbablySupported = function() {
462     // This is rather awful, but can be useful.
463     var match;
464     if (window.opera)
465       return Number(window.opera.version()) >= 9.52;
466     else if (/Apple Computers, Inc/.test(navigator.vendor) && (match = navigator.
          userAgent.match(/Version\/(\d+(?:\.\d+)?)\./)))
467       return Number(match[1]) >= 3;
468     else if (document.selection && window.ActiveXObject && (match = navigator.
          userAgent.match(/MSIE (\d+(?:\.\d*)?)\b/)))
469       return Number(match[1]) >= 6;
470     else if (match = navigator.userAgent.match(/gecko\/(\d{8})/i))
471       return Number(match[1]) >= 20050901;
472     else if (match = navigator.userAgent.match(/AppleWebKit\/(\d+)/))
473       return Number(match[1]) >= 525;
474     else
475       return null;
476   };
477   return CodeMirror;
478 })();
```

4.3.2. editor.js

```
/* The Editor object manages the content of the editable frame. It
 * catches events, colours nodes, and indents lines. This file also
 * holds some functions for transforming arbitrary DOM structures into
 * plain sequences of <span> and <br> elements
 */

var internetExplorer = document.selection && window.ActiveXObject && /MSIE/.test(
    navigator.userAgent);
var webkit = /AppleWebKit/.test(navigator.userAgent);
var safari = /Apple Computers, Inc/.test(navigator.vendor);
var gecko = /gecko\/(\d{8})/i.test(navigator.userAgent);

// Make sure a string does not contain two consecutive 'collapseable'
// whitespace characters.
function makeWhiteSpace(n) {
  var buffer = [], nb = true;
  for (; n > 0; n--) {
    buffer.push((nb || n == 1) ? nbsp : " ");
    nb = !nb;
  }
  return buffer.join("");
}

// Create a set of white-space characters that will not be collapsed
// by the browser, but will not break text-wrapping either.
function fixSpaces(string) {
  if (string.charAt(0) == " ") string = nbsp + string.slice(1);
  return string.replace(/\t/g, function(){return makeWhiteSpace(indentUnit);})
    .replace(/[ \u00a0]{2,}/g, function(s) {return makeWhiteSpace(s.length);});
}

function cleanText(text) {
  return text.replace(/\u00a0/g, " ").replace(/\u200b/g, "");
}

// Create a SPAN node with the expected properties for document part
// spans.
function makePartSpan(value, doc) {
  var text = value;
  if (value.nodeType == 3) text = value.nodeValue;
  else value = doc.createTextNode(text);

  var span = doc.createElement("SPAN");
  span.isPart = true;
  span.appendChild(value);
  span.currentText = text;
  return span;
}

// On webkit, when the last BR of the document does not have text
// behind it, the cursor can not be put on the line after it. This
```

```
51  // makes pressing enter at the end of the document occasionally do
52  // nothing (or at least seem to do nothing). To work around it, this
53  // function makes sure the document ends with a span containing a
54  // zero-width space character. The traverseDOM iterator filters such
55  // character out again, so that the parsers won't see them. This
56  // function is called from a few strategic places to make sure the
57  // zwsp is restored after the highlighting process eats it.
58  var webkitLastLineHack = webkit ?
59    function(container) {
60      var last = container.lastChild;
61      if (!last || !last.isPart || last.textContent != "\u200b")
62        container.appendChild(makePartSpan("\u200b", container.ownerDocument));
63    } : function() {};
64
65  var Editor = (function(){
66    // The HTML elements whose content should be suffixed by a newline
67    // when converting them to flat text.
68    var newlineElements = {"P": true, "DIV": true, "LI": true};
69
70    function asEditorLines(string) {
71      var tab = makeWhiteSpace(indentUnit);
72      return map(string.replace(/\t/g, tab).replace(/\u00a0/g, " ").replace(/\r\n?/g,
           "\n").split("\n"), fixSpaces);
73    }
74
75    // Helper function for traverseDOM. Flattens an arbitrary DOM node
76    // into an array of textnodes and <br> tags.
77    function simplifyDOM(root, atEnd) {
78      var doc = root.ownerDocument;
79      var result = [];
80      var leaving = true;
81
82      function simplifyNode(node, top) {
83        if (node.nodeType == 3) {
84          var text = node.nodeValue = fixSpaces(node.nodeValue.replace(/[\r\u200b]/g,
               "").replace(/\n/g, " "));
85          if (text.length) leaving = false;
86          result.push(node);
87        }
88        else if (isBR(node) && node.childNodes.length == 0) {
89          leaving = true;
90          result.push(node);
91        }
92        else {
93          forEach(node.childNodes, simplifyNode);
94          if (!leaving && newlineElements.hasOwnProperty(node.nodeName.toUpperCase())
                ) {
95            leaving = true;
96            if (!atEnd || !top)
97              result.push(doc.createElement("BR"));
98          }
99        }
100     }
101
```

```
102       simplifyNode(root, true);
103       return result;
104     }
105
106     // Creates a MochiKit-style iterator that goes over a series of DOM
107     // nodes. The values it yields are strings, the textual content of
108     // the nodes. It makes sure that all nodes up to and including the
109     // one whose text is being yielded have been 'normalized' to be just
110     // <span> and <br> elements.
111     // See the story.html file for some short remarks about the use of
112     // continuation-passing style in this iterator.
113     function traverseDOM(start){
114       function _yield(value, c){cc = c; return value;}
115       function push(fun, arg, c){return function(){return fun(arg, c);};}
116       function stop(){cc = stop; throw StopIteration;};
117       var cc = push(scanNode, start, stop);
118       var owner = start.ownerDocument;
119       var nodeQueue = [];
120
121       // Create a function that can be used to insert nodes after the
122       // one given as argument.
123       function pointAt(node){
124         var parent = node.parentNode;
125         var next = node.nextSibling;
126         return function(newnode) {
127           parent.insertBefore(newnode, next);
128         };
129       }
130       var point = null;
131
132       // This an Opera-specific hack -- always insert an empty span
133       // between two BRs, because Opera's cursor code gets terribly
134       // confused when the cursor is between two BRs.
135       var afterBR = true;
136
137       // Insert a normalized node at the current point. If it is a text
138       // node, wrap it in a <span>, and give that span a currentText
139       // property -- this is used to cache the nodeValue, because
140       // directly accessing nodeValue is horribly slow on some browsers.
141       // The dirty property is used by the highlighter to determine
142       // which parts of the document have to be re-highlighted.
143       function insertPart(part){
144         var text = "\n";
145         if (part.nodeType == 3) {
146           select.snapshotChanged();
147           part = makePartSpan(part, owner);
148           text = part.currentText;
149           afterBR = false;
150         }
151         else {
152           if (afterBR && window.opera)
153             point(makePartSpan("", owner));
154           afterBR = true;
155         }
```

```
156         part.dirty = true;
157         nodeQueue.push(part);
158         point(part);
159         return text;
160       }
161
162       // Extract the text and newlines from a DOM node, insert them into
163       // the document, and yield the textual content. Used to replace
164       // non-normalized nodes.
165       function writeNode(node, c, end) {
166         var toYield = [];
167         forEach(simplifyDOM(node, end), function(part) {
168           toYield.push(insertPart(part));
169         });
170         return _yield(toYield.join(""), c);
171       }
172
173       // Check whether a node is a normalized <span> element.
174       function partNode(node){
175         if (node.isPart && node.childNodes.length == 1 && node.firstChild.nodeType ==
               3) {
176           node.currentText = node.firstChild.nodeValue;
177           return !/[\n\t\r]/.test(node.currentText);
178         }
179         return false;
180       }
181
182       // Handle a node. Add its successor to the continuation if there
183       // is one, find out whether the node is normalized. If it is,
184       // yield its content, otherwise, normalize it (writeNode will take
185       // care of yielding).
186       function scanNode(node, c){
187         if (node.nextSibling)
188           c = push(scanNode, node.nextSibling, c);
189
190         if (partNode(node)){
191           nodeQueue.push(node);
192           afterBR = false;
193           return _yield(node.currentText, c);
194         }
195         else if (isBR(node)) {
196           if (afterBR && window.opera)
197             node.parentNode.insertBefore(makePartSpan("", owner), node);
198           nodeQueue.push(node);
199           afterBR = true;
200           return _yield("\n", c);
201         }
202         else {
203           var end = !node.nextSibling;
204           point = pointAt(node);
205           removeElement(node);
206           return writeNode(node, c, end);
207         }
208       }
```

```
209
210       // MochiKit iterators are objects with a next function that
211       // returns the next value or throws StopIteration when there are
212       // no more values.
213       return {next: function(){return cc();}, nodes: nodeQueue};
214     }
215
216     // Determine the text size of a processed node.
217     function nodeSize(node) {
218       return isBR(node) ? 1 : node.currentText.length;
219     }
220
221     // Search backwards through the top-level nodes until the next BR or
222     // the start of the frame.
223     function startOfLine(node) {
224       while (node && !isBR(node)) node = node.previousSibling;
225       return node;
226     }
227     function endOfLine(node, container) {
228       if (!node) node = container.firstChild;
229       else if (isBR(node)) node = node.nextSibling;
230
231       while (node && !isBR(node)) node = node.nextSibling;
232       return node;
233     }
234
235     function time() {return new Date().getTime();}
236
237     // Client interface for searching the content of the editor. Create
238     // these by calling CodeMirror.getSearchCursor. To use, call
239     // findNext on the resulting object -- this returns a boolean
240     // indicating whether anything was found, and can be called again to
241     // skip to the next find. Use the select and replace methods to
242     // actually do something with the found locations.
243     function SearchCursor(editor, string, fromCursor, caseFold) {
244       this.editor = editor;
245       this.caseFold = caseFold;
246       if (caseFold) string = string.toLowerCase();
247       this.history = editor.history;
248       this.history.commit();
249
250       // Are we currently at an occurrence of the search string?
251       this.atOccurrence = false;
252       // The object stores a set of nodes coming after its current
253       // position, so that when the current point is taken out of the
254       // DOM tree, we can still try to continue.
255       this.fallbackSize = 15;
256       var cursor;
257       // Start from the cursor when specified and a cursor can be found.
258       if (fromCursor && (cursor = select.cursorPos(this.editor.container))) {
259         this.line = cursor.node;
260         this.offset = cursor.offset;
261       }
262       else {
```

```
263         this.line = null;
264         this.offset = 0;
265       }
266       this.valid = !!string;
267
268       // Create a matcher function based on the kind of string we have.
269       var target = string.split("\n"), self = this;
270       this.matches = (target.length == 1) ?
271         // For one-line strings, searching can be done simply by calling
272         // indexOf on the current line.
273         function() {
274           var line = cleanText(self.history.textAfter(self.line).slice(self.offset));
275           var match = (self.caseFold ? line.toLowerCase() : line).indexOf(string);
276           if (match > -1)
277             return {from: {node: self.line, offset: self.offset + match},
278                     to: {node: self.line, offset: self.offset + match + string.length
                            }};
279         } :
280         // Multi-line strings require internal iteration over lines, and
281         // some clunky checks to make sure the first match ends at the
282         // end of the line and the last match starts at the start.
283         function() {
284           var firstLine = cleanText(self.history.textAfter(self.line).slice(self.
                offset));
285           var match = (self.caseFold ? firstLine.toLowerCase() : firstLine).
                lastIndexOf(target[0]);
286           if (match == -1 || match != firstLine.length - target[0].length)
287             return false;
288           var startOffset = self.offset + match;
289
290           var line = self.history.nodeAfter(self.line);
291           for (var i = 1; i < target.length - 1; i++) {
292             var lineText = cleanText(self.history.textAfter(line));
293             if ((self.caseFold ? lineText.toLowerCase() : lineText) != target[i])
294               return false;
295             line = self.history.nodeAfter(line);
296           }
297
298           var lastLine = cleanText(self.history.textAfter(line));
299           if ((self.caseFold ? lastLine.toLowerCase() : lastLine).indexOf(target[
                target.length - 1]) != 0)
300             return false;
301
302           return {from: {node: self.line, offset: startOffset},
303                   to: {node: line, offset: target[target.length - 1].length}};
304         };
305     }
306
307     SearchCursor.prototype = {
308       findNext: function() {
309         if (!this.valid) return false;
310         this.atOccurrence = false;
311         var self = this;
312
```

```
313         // Go back to the start of the document if the current line is
314         // no longer in the DOM tree.
315         if (this.line && !this.line.parentNode) {
316           this.line = null;
317           this.offset = 0;
318         }
319
320         // Set the cursor's position one character after the given
321         // position.
322         function saveAfter(pos) {
323           if (self.history.textAfter(pos.node).length > pos.offset) {
324             self.line = pos.node;
325             self.offset = pos.offset + 1;
326           }
327           else {
328             self.line = self.history.nodeAfter(pos.node);
329             self.offset = 0;
330           }
331         }
332
333         while (true) {
334           var match = this.matches();
335           // Found the search string.
336           if (match) {
337             this.atOccurrence = match;
338             saveAfter(match.from);
339             return true;
340           }
341           this.line = this.history.nodeAfter(this.line);
342           this.offset = 0;
343           // End of document.
344           if (!this.line) {
345             this.valid = false;
346             return false;
347           }
348         }
349       },
350
351       select: function() {
352         if (this.atOccurrence) {
353           select.setCursorPos(this.editor.container, this.atOccurrence.from, this.
                atOccurrence.to);
354           select.scrollToCursor(this.editor.container);
355         }
356       },
357
358       replace: function(string) {
359         if (this.atOccurrence) {
360           var end = this.editor.replaceRange(this.atOccurrence.from, this.
                atOccurrence.to, string);
361           this.line = end.node;
362           this.offset = end.offset;
363           this.atOccurrence = false;
364         }
```

```
365       }
366     };
367
368     // The Editor object is the main inside-the-iframe interface.
369     function Editor(options) {
370       this.options = options;
371       window.indentUnit = options.indentUnit;
372       this.parent = parent;
373       this.doc = document;
374       var container = this.container = this.doc.body;
375       this.win = window;
376       this.history = new History(container, options.undoDepth, options.undoDelay,
              this);
377       var self = this;
378
379       if (!Editor.Parser)
380         throw "No parser loaded.";
381       if (options.parserConfig && Editor.Parser.configure)
382         Editor.Parser.configure(options.parserConfig);
383
384       if (!options.readOnly)
385         select.setCursorPos(container, {node: null, offset: 0});
386
387       this.dirty = [];
388       this.importCode(options.content || "");
389       this.history.onChange = options.onChange;
390
391       if (!options.readOnly) {
392         if (options.continuousScanning !== false) {
393           this.scanner = this.documentScanner(options.passTime);
394           this.delayScanning();
395         }
396
397         function setEditable() {
398           // In IE, designMode frames can not run any scripts, so we use
399           // contentEditable instead.
400           if (document.body.contentEditable != undefined && internetExplorer)
401             document.body.contentEditable = "true";
402           else
403             document.designMode = "on";
404
405           document.documentElement.style.borderWidth = "0";
406           if (!options.textWrapping)
407             container.style.whiteSpace = "nowrap";
408         }
409
410         // If setting the frame editable fails, try again when the user
411         // focus it (happens when the frame is not visible on
412         // initialisation, in Firefox).
413         try {
414           setEditable();
415         }
416         catch(e) {
417           var focusEvent = addEventHandler(document, "focus", function() {
```

```
418             focusEvent();
419             setEditable();
420           }, true);
421         }
422
423         //addEventHandler(document, "keydown", method(this, "keyDown"));
424         //addEventHandler(document, "keypress", method(this, "keyPress"));
425         //addEventHandler(document, "keyup", method(this, "keyUp"));
426
427         function cursorActivity() {self.cursorActivity(false);}
428         addEventHandler(document.body, "mouseup", cursorActivity);
429         addEventHandler(document.body, "cut", cursorActivity);
430
431         // workaround for a gecko bug [?] where going forward and then
432         // back again breaks designmode (no more cursor)
433         if (gecko)
434           addEventHandler(this.win, "pagehide", function(){self.unloaded = true;});
435
436         addEventHandler(document.body, "paste", function(event) {
437           cursorActivity();
438           var text = null;
439           try {
440             var clipboardData = event.clipboardData || window.clipboardData;
441             if (clipboardData) text = clipboardData.getData('Text');
442           }
443           catch(e) {}
444           if (text !== null) {
445             event.stop();
446             self.replaceSelection(text);
447             select.scrollToCursor(self.container);
448           }
449         });
450
451         if (this.options.autoMatchParens)
452           addEventHandler(document.body, "click", method(this, "
                scheduleParenHighlight"));
453       }
454       else if (!options.textWrapping) {
455         container.style.whiteSpace = "nowrap";
456       }
457     }
458
459     function isSafeKey(code) {
460       return (code >= 16 && code <= 18) || // shift, control, alt
461              (code >= 33 && code <= 40); // arrows, home, end
462     }
463
464     Editor.prototype = {
465       // Import a piece of code into the editor.
466       importCode: function(code) {
467         this.history.push(null, null, asEditorLines(code));
468         this.history.reset();
469       },
470
```

```
471       // Extract the code from the editor.
472       getCode: function() {
473         if (!this.container.firstChild)
474           return "";
475
476         var accum = [];
477         select.markSelection(this.win);
478         forEach(traverseDOM(this.container.firstChild), method(accum, "push"));
479         webkitLastLineHack(this.container);
480         select.selectMarked();
481         return cleanText(accum.join(""));
482       },
483
484       afterPaste: function(line, pos, caracter) {
485           var newLine = this.cursorPosition().line;
486           var back = this.currentLine() - line;
487
488           for (i=0; i<back; i++) {
489                   newLine = this.prevLine(newLine);
490           }
491
492           if (pos == -0.1)
493                   pos = strlen(this.lineContent(newLine));
494           else if (pos < 0)
495                   pos = strlen(this.lineContent(newLine)) + pos;
496
497           select.setCursorPos(this.container, {node: newLine, offset: pos});
498           select.scrollToCursor(this.container);
499
500       },
501
502       checkLine: function(node) {
503         if (node === false || !(node == null || node.parentNode == this.container))
504           throw parent.CodeMirror.InvalidLineHandle;
505       },
506
507       currentLine: function() {
508           var i = 0;
509           var line = this.cursorPosition().line;
510               while (line != false) {
511                       i = i + 1;
512                       line = this.prevLine(line);
513               }
514               return i;
515       },
516
517       currentPos: function() {
518           var pos = this.cursorPosition().character;
519           var line = this.cursorPosition().line;
520           while (line != false) {
521                   line = this.prevLine(line);
522           }
523           return pos;
524       },
```

```
525
526       cursorPosition: function(start) {
527         if (start == null) start = true;
528         var pos = select.cursorPos(this.container, start);
529         if (pos) return {line: pos.node, character: pos.offset};
530         else return {line: null, character: 0};
531       },
532
533       firstLine: function() {
534         return null;
535       },
536
537       lastLine: function() {
538         if (this.container.lastChild) return startOfLine(this.container.lastChild);
539         else return null;
540       },
541
542       nextLine: function(line) {
543         this.checkLine(line);
544         var end = endOfLine(line, this.container);
545         return end || false;
546       },
547
548       prevLine: function(line) {
549         this.checkLine(line);
550         if (line == null) return false;
551         return startOfLine(line.previousSibling);
552       },
553
554       visibleLineCount: function() {
555         var line = this.container.firstChild;
556         while (line && isBR(line)) line = line.nextSibling; // BR heights are
              unreliable
557         if (!line) return false;
558         var innerHeight = (window.innerHeight
559                            || document.documentElement.clientHeight
560                            || document.body.clientHeight);
561         return Math.floor(innerHeight / line.offsetHeight);
562       },
563
564       selectLines: function(startLine, startOffset, endLine, endOffset) {
565         this.checkLine(startLine);
566         var start = {node: startLine, offset: startOffset}, end = null;
567         if (endOffset !== undefined) {
568           this.checkLine(endLine);
569           end = {node: endLine, offset: endOffset};
570         }
571         select.setCursorPos(this.container, start, end);
572         select.scrollToCursor(this.container);
573       },
574
575       lineContent: function(line) {
576         var accum = [];
577         for (line = line ? line.nextSibling : this.container.firstChild;
```

```
578             line && !isBR(line); line = line.nextSibling)
579         accum.push(nodeText(line));
580       return cleanText(accum.join(""));
581     },
582
583     setLineContent: function(line, content) {
584       this.history.commit();
585       this.replaceRange({node: line, offset: 0},
586                         {node: line, offset: this.history.textAfter(line).length},
587                         content);
588       this.addDirtyNode(line);
589       this.scheduleHighlight();
590     },
591
592     removeLine: function(line) {
593       var node = line ? line.nextSibling : this.container.firstChild;
594       while (node) {
595         var next = node.nextSibling;
596         removeElement(node);
597         if (isBR(node)) break;
598         node = next;
599       }
600       this.addDirtyNode(line);
601       this.scheduleHighlight();
602     },
603
604     insertIntoLine: function(line, position, content) {
605       var before = null;
606       if (position == "end") {
607         before = endOfLine(line, this.container);
608       }
609       else {
610         for (var cur = line ? line.nextSibling : this.container.firstChild; cur;
                 cur = cur.nextSibling) {
611           if (position == 0) {
612             before = cur;
613             break;
614           }
615           var text = nodeText(cur);
616           if (text.length > position) {
617             before = cur.nextSibling;
618             content = text.slice(0, position) + content + text.slice(position);
619             removeElement(cur);
620             break;
621           }
622           position -= text.length;
623         }
624       }
625
626       var lines = asEditorLines(content), doc = this.container.ownerDocument;
627       for (var i = 0; i < lines.length; i++) {
628         if (i > 0) this.container.insertBefore(doc.createElement("BR"), before);
629         this.container.insertBefore(makePartSpan(lines[i], doc), before);
630       }
```

```
631         this.addDirtyNode(line);
632         this.scheduleHighlight();
633       },
634
635       // Retrieve the selected text.
636       selectedText: function() {
637         var h = this.history;
638         h.commit();
639
640         var start = select.cursorPos(this.container, true),
641             end = select.cursorPos(this.container, false);
642         if (!start || !end) return "";
643
644         if (start.node == end.node)
645           return h.textAfter(start.node).slice(start.offset, end.offset);
646
647         var text = [h.textAfter(start.node).slice(start.offset)];
648         for (var pos = h.nodeAfter(start.node); pos != end.node; pos = h.nodeAfter(
              pos))
649           text.push(h.textAfter(pos));
650         text.push(h.textAfter(end.node).slice(0, end.offset));
651         return cleanText(text.join("\n"));
652       },
653
654       // Replace the selection with another piece of text.
655       replaceSelection: function(text) {
656         this.history.commit();
657
658         var start = select.cursorPos(this.container, true),
659             end = select.cursorPos(this.container, false);
660         if (!start || !end) return;
661
662         end = this.replaceRange(start, end, text);
663         select.setCursorPos(this.container, end);
664         webkitLastLineHack(this.container);
665       },
666
667       reroutePasteEvent: function() {
668         if (this.capturingPaste || window.opera) return;
669         this.capturingPaste = true;
670         var te = window.frameElement.CodeMirror.textareaHack;
671         parent.focus();
672         te.value = "";
673         te.focus();
674
675         var self = this;
676         this.parent.setTimeout(function() {
677           self.capturingPaste = false;
678           self.win.focus();
679           if (self.selectionSnapshot) // IE hack
680             self.win.select.setBookmark(self.container, self.selectionSnapshot);
681           var text = te.value;
682           if (text) {
683             self.replaceSelection(text);
```

```
684             select.scrollToCursor(self.container);
685           }
686         }, 10);
687       },
688
689       replaceRange: function(from, to, text) {
690         var lines = asEditorLines(text);
691         lines[0] = this.history.textAfter(from.node).slice(0, from.offset) + lines
                [0];
692         var lastLine = lines[lines.length - 1];
693         lines[lines.length - 1] = lastLine + this.history.textAfter(to.node).slice(to
                .offset);
694         var end = this.history.nodeAfter(to.node);
695         this.history.push(from.node, end, lines);
696         return {node: this.history.nodeBefore(end),
697                 offset: lastLine.length};
698       },
699
700       getSearchCursor: function(string, fromCursor, caseFold) {
701         return new SearchCursor(this, string, fromCursor, caseFold);
702       },
703
704       // Re-indent the whole buffer
705       reindent: function() {
706         if (this.container.firstChild)
707           this.indentRegion(null, this.container.lastChild);
708       },
709
710       reindentSelection: function(direction) {
711         if (!select.somethingSelected(this.win)) {
712           this.indentAtCursor(direction);
713         }
714         else {
715           var start = select.selectionTopNode(this.container, true),
716               end = select.selectionTopNode(this.container, false);
717           if (start === false || end === false) return;
718           this.indentRegion(start, end, direction);
719         }
720       },
721
722       grabKeys: function(eventHandler, filter) {
723         this.frozen = eventHandler;
724         this.keyFilter = filter;
725       },
726       ungrabKeys: function() {
727         this.frozen = "leave";
728         this.keyFilter = null;
729       },
730
731       setParser: function(name) {
732         Editor.Parser = window[name];
733         if (this.container.firstChild) {
734           forEach(this.container.childNodes, function(n) {
735             if (n.nodeType != 3) n.dirty = true;
```

```
736           });
737           this.addDirtyNode(this.firstChild);
738           this.scheduleHighlight();
739         }
740       },
741
742       // Intercept enter and tab, and assign their new functions.
743       keyDown: function(event) {
744         if (this.frozen == "leave") this.frozen = null;
745         if (this.frozen && (!this.keyFilter || this.keyFilter(event.keyCode, event))
              {
746           event.stop();
747           this.frozen(event);
748           return;
749         }
750
751         var code = event.keyCode;
752         // Don't scan when the user is typing.
753         this.delayScanning();
754         // Schedule a paren-highlight event, if configured.
755         if (this.options.autoMatchParens)
756           this.scheduleParenHighlight();
757
758         // The various checks for !altKey are there because AltGr sets both
759         // ctrlKey and altKey to true, and should not be recognised as
760         // Control.
761         if (code == 13) { // enter
762           if (event.ctrlKey && !event.altKey) {
763             this.reparseBuffer();
764           }
765           else {
766             select.insertNewlineAtCursor(this.win);
767             this.indentAtCursor();
768             select.scrollToCursor(this.container);
769           }
770           event.stop();
771         }
772         else if (code == 9 && this.options.tabMode != "default" && !event.ctrlKey) {
              // tab
773           this.handleTab(!event.shiftKey);
774           event.stop();
775         }
776         else if (code == 32 && event.shiftKey && this.options.tabMode == "default") {
              // space
777           this.handleTab(true);
778           event.stop();
779         }
780         else if (code == 36 && !event.shiftKey && !event.ctrlKey) { // home
781           if (this.home()) event.stop();
782         }
783         else if (code == 35 && !event.shiftKey && !event.ctrlKey) { // end
784           if (this.end()) event.stop();
785         }
786         // Only in Firefox is the default behavior for PgUp/PgDn correct.
```

```
787         else if (code == 33 && !event.shiftKey && !event.ctrlKey && !gecko) { // PgUp
788           if (this.pageUp()) event.stop();
789         }
790         else if (code == 34 && !event.shiftKey && !event.ctrlKey && !gecko) {  //
                PgDn
791           if (this.pageDown()) event.stop();
792         }
793         else if ((code == 219 || code == 221) && event.ctrlKey && !event.altKey) { //
                [, ]
794           this.highlightParens(event.shiftKey, true);
795           event.stop();
796         }
797         else if (event.metaKey && !event.shiftKey && (code == 37 || code == 39)) { //
                Meta-left/right
798           var cursor = select.selectionTopNode(this.container);
799           if (cursor === false || !this.container.firstChild) return;
800
801           if (code == 37) select.focusAfterNode(startOfLine(cursor), this.container);
802           else {
803             var end = endOfLine(cursor, this.container);
804             select.focusAfterNode(end ? end.previousSibling : this.container.
                    lastChild, this.container);
805           }
806           event.stop();
807         }
808         else if ((event.ctrlKey || event.metaKey) && !event.altKey) {
809           if ((event.shiftKey && code == 90) || code == 89) { // shift-Z, Y
810             select.scrollToNode(this.history.redo());
811             event.stop();
812           }
813           else if (code == 90 || (safari && code == 8)) { // Z, backspace
814             select.scrollToNode(this.history.undo());
815             event.stop();
816           }
817           else if (code == 83 && this.options.saveFunction) { // S
818             this.options.saveFunction();
819             event.stop();
820           }
821           else if (internetExplorer && code == 86) {
822             this.reroutePasteEvent();
823           }
824         }
825       },
826
827       // Check for characters that should re-indent the current line,
828       // and prevent Opera from handling enter and tab anyway.
829       keyPress: function(event) {
830         var electric = Editor.Parser.electricChars, self = this;
831         // Hack for Opera, and Firefox on OS X, in which stopping a
832         // keydown event does not prevent the associated keypress event
833         // from happening, so we have to cancel enter and tab again
834         // here.
835
```

```
836         if ((this.frozen && (!this.keyFilter || this.keyFilter(event.keyCode, event))
                ) ||
837             event.code == 13 || (event.code == 9 && this.options.tabMode != "default"
                    ) ||
838             (event.keyCode == 32 && event.shiftKey && this.options.tabMode == "
                    default"))
839           event.stop();
840         else if (electric && electric.indexOf(event.character) != -1)
841           this.parent.setTimeout(function(){self.indentAtCursor(null);}, 0);
842         else if ((event.character == "v" || event.character == "V")
843                   && (event.ctrlKey || event.metaKey) && !event.altKey) // ctrl-V
844           this.reroutePasteEvent();
845       },
846
847       // Mark the node at the cursor dirty when a non-safe key is
848       // released.
849       keyUp: function(event) {
850         this.cursorActivity(isSafeKey(event.keyCode));
851       },
852
853       // Indent the line following a given <br>, or null for the first
854       // line. If given a <br> element, this must have been highlighted
855       // so that it has an indentation method. Returns the whitespace
856       // element that has been modified or created (if any).
857       indentLineAfter: function(start, direction) {
858         // whiteSpace is the whitespace span at the start of the line,
859         // or null if there is no such node.
860         var whiteSpace = start ? start.nextSibling : this.container.firstChild;
861         if (whiteSpace && !hasClass(whiteSpace, "whitespace"))
862           whiteSpace = null;
863
864         // Sometimes the start of the line can influence the correct
865         // indentation, so we retrieve it.
866         var firstText = whiteSpace ? whiteSpace.nextSibling : (start ? start.
                nextSibling : this.container.firstChild);
867         var nextChars = (start && firstText && firstText.currentText) ? firstText.
                currentText : "";
868
869         // Ask the lexical context for the correct indentation, and
870         // compute how much this differs from the current indentation.
871         var newIndent = 0, curIndent = whiteSpace ? whiteSpace.currentText.length :
                0;
872         if (direction != null && this.options.tabMode == "shift")
873           newIndent = direction ? curIndent + indentUnit : Math.max(0, curIndent -
                  indentUnit)
874         else if (start)
875           newIndent = start.indentation(nextChars, curIndent, direction);
876         else if (Editor.Parser.firstIndentation)
877           newIndent = Editor.Parser.firstIndentation(nextChars, curIndent, direction)
                ;
878         var indentDiff = newIndent - curIndent;
879
880         // If there is too much, this is just a matter of shrinking a span.
881         if (indentDiff < 0) {
```

```
882           if (newIndent == 0) {
883             if (firstText) select.snapshotMove(whiteSpace.firstChild, firstText.
                    firstChild, 0);
884             removeElement(whiteSpace);
885             whiteSpace = null;
886           }
887           else {
888             select.snapshotMove(whiteSpace.firstChild, whiteSpace.firstChild,
                    indentDiff, true);
889             whiteSpace.currentText = makeWhiteSpace(newIndent);
890             whiteSpace.firstChild.nodeValue = whiteSpace.currentText;
891           }
892         }
893         // Not enough...
894         else if (indentDiff > 0) {
895           // If there is whitespace, we grow it.
896           if (whiteSpace) {
897             whiteSpace.currentText = makeWhiteSpace(newIndent);
898             whiteSpace.firstChild.nodeValue = whiteSpace.currentText;
899           }
900           // Otherwise, we have to add a new whitespace node.
901           else {
902             whiteSpace = makePartSpan(makeWhiteSpace(newIndent), this.doc);
903             whiteSpace.className = "whitespace";
904             if (start) insertAfter(whiteSpace, start);
905             else this.container.insertBefore(whiteSpace, this.container.firstChild);
906           }
907           var fromNode = firstText && (firstText.firstChild || firstText);
908           select.snapshotMove(fromNode, whiteSpace.firstChild, newIndent, false, true
                  );
909         }
910         if (indentDiff != 0) this.addDirtyNode(start);
911       },
912
913       // Re-highlight the selected part of the document.
914       highlightAtCursor: function() {
915         var pos = select.selectionTopNode(this.container, true);
916         var to = select.selectionTopNode(this.container, false);
917         if (pos === false || to === false) return false;
918
919         select.markSelection(this.win);
920         if (this.highlight(pos, endOfLine(to, this.container), true, 20) === false)
921           return false;
922         select.selectMarked();
923         return true;
924       },
925
926       // When tab is pressed with text selected, the whole selection is
927       // re-indented, when nothing is selected, the line with the cursor
928       // is re-indented.
929       handleTab: function(direction) {
930         if (this.options.tabMode == "spaces")
931           select.insertTabAtCursor(this.win);
932         else
```

```
933           this.reindentSelection(direction);
934       },
935
936       // Custom home behaviour that doesn't land the cursor in front of
937       // leading whitespace unless pressed twice.
938       home: function() {
939         var cur = select.selectionTopNode(this.container, true), start = cur;
940         if (cur === false || !(!cur || cur.isPart || isBR(cur)) || !this.container.
              firstChild)
941           return false;
942
943         while (cur && !isBR(cur)) cur = cur.previousSibling;
944         var next = cur ? cur.nextSibling : this.container.firstChild;
945         if (next && next != start && next.isPart && hasClass(next, "whitespace"))
946           select.focusAfterNode(next, this.container);
947         else
948           select.focusAfterNode(cur, this.container);
949
950         select.scrollToCursor(this.container);
951         return true;
952       },
953
954       // Some browsers (Opera) don't manage to handle the end key
955       // properly in the face of vertical scrolling.
956       end: function() {
957         var cur = select.selectionTopNode(this.container, true);
958         if (cur === false) return false;
959         cur = endOfLine(cur, this.container);
960         if (!cur) return false;
961         select.focusAfterNode(cur.previousSibling, this.container);
962         select.scrollToCursor(this.container);
963         return true;
964       },
965
966       pageUp: function() {
967         var line = this.cursorPosition().line, scrollAmount = this.visibleLineCount()
              ;
968         if (line === false || scrollAmount === false) return false;
969         // Try to keep one line on the screen.
970         scrollAmount -= 2;
971         for (var i = 0; i < scrollAmount; i++) {
972           line = this.prevLine(line);
973           if (line === false) break;
974         }
975         if (i == 0) return false; // Already at first line
976         select.setCursorPos(this.container, {node: line, offset: 0});
977         select.scrollToCursor(this.container);
978         return true;
979       },
980
981       pageDown: function() {
982         var line = this.cursorPosition().line, scrollAmount = this.visibleLineCount()
              ;
983         if (line === false || scrollAmount === false) return false;
```

```
984         // Try to move to the last line of the current page.
985         scrollAmount -= 2;
986         for (var i = 0; i < scrollAmount; i++) {
987           var nextLine = this.nextLine(line);
988           if (nextLine === false) break;
989           line = nextLine;
990         }
991         if (i == 0) return false; // Already at last line
992         select.setCursorPos(this.container, {node: line, offset: 0});
993         select.scrollToCursor(this.container);
994         return true;
995       },
996
997       // Delay (or initiate) the next paren highlight event.
998       scheduleParenHighlight: function() {
999         if (this.parenEvent) this.parent.clearTimeout(this.parenEvent);
1000        var self = this;
1001        this.parenEvent = this.parent.setTimeout(function(){self.highlightParens();},
              300);
1002      },
1003
1004      // Take the token before the cursor. If it contains a character in
1005      // '()[]{}', search for the matching paren/brace/bracket, and
1006      // highlight them in green for a moment, or red if no proper match
1007      // was found.
1008      highlightParens: function(jump, fromKey) {
1009        var self = this;
1010        // give the relevant nodes a colour.
1011        function highlight(node, ok) {
1012          if (!node) return;
1013          if (self.options.markParen) {
1014            self.options.markParen(node, ok);
1015          }
1016          else {
1017            node.style.fontWeight = "bold";
1018            node.style.color = ok ? "#8F8" : "#F88";
1019          }
1020        }
1021        function unhighlight(node) {
1022          if (!node) return;
1023          if (self.options.unmarkParen) {
1024            self.options.unmarkParen(node);
1025          }
1026          else {
1027            node.style.fontWeight = "";
1028            node.style.color = "";
1029          }
1030        }
1031        if (!fromKey && self.highlighted) {
1032          unhighlight(self.highlighted[0]);
1033          unhighlight(self.highlighted[1]);
1034        }
1035
1036        if (!window.select) return;
```

```
1037        // Clear the event property.
1038        if (this.parenEvent) this.parent.clearTimeout(this.parenEvent);
1039        this.parenEvent = null;
1040
1041        // Extract a 'paren' from a piece of text.
1042        function paren(node) {
1043          if (node.currentText) {
1044            var match = node.currentText.match(/^[\s\u00a0]*([\(\)\[\]{}])[\s\u00a0]*
                  $/);
1045            return match && match[1];
1046          }
1047        }
1048        // Determine the direction a paren is facing.
1049        function forward(ch) {
1050          return /[\(\[\{]/.test(ch);
1051        }
1052
1053        var ch, cursor = select.selectionTopNode(this.container, true);
1054        if (!cursor || !this.highlightAtCursor()) return;
1055        cursor = select.selectionTopNode(this.container, true);
1056        if (!(cursor && ((ch = paren(cursor)) || (cursor = cursor.nextSibling) && (ch
               = paren(cursor)))))
1057          return;
1058        // We only look for tokens with the same className.
1059        var className = cursor.className, dir = forward(ch), match = matching[ch];
1060
1061        // Since parts of the document might not have been properly
1062        // highlighted, and it is hard to know in advance which part we
1063        // have to scan, we just try, and when we find dirty nodes we
1064        // abort, parse them, and re-try.
1065        function tryFindMatch() {
1066          var stack = [], ch, ok = true;
1067          for (var runner = cursor; runner; runner = dir ? runner.nextSibling :
                  runner.previousSibling) {
1068            if (runner.className == className && isSpan(runner) && (ch = paren(runner
                    ))) {
1069              if (forward(ch) == dir)
1070                stack.push(ch);
1071              else if (!stack.length)
1072                ok = false;
1073              else if (stack.pop() != matching[ch])
1074                ok = false;
1075              if (!stack.length) break;
1076            }
1077            else if (runner.dirty || !isSpan(runner) && !isBR(runner)) {
1078              return {node: runner, status: "dirty"};
1079            }
1080          }
1081          return {node: runner, status: runner && ok};
1082        }
1083
1084        while (true) {
1085          var found = tryFindMatch();
1086          if (found.status == "dirty") {
```

```
1087            this.highlight(found.node, endOfLine(found.node));
1088            // Needed because in some corner cases a highlight does not
1089            // reach a node.
1090            found.node.dirty = false;
1091            continue;
1092          }
1093          else {
1094            highlight(cursor, found.status);
1095            highlight(found.node, found.status);
1096            if (fromKey)
1097              self.parent.setTimeout(function() {unhighlight(cursor); unhighlight(
                      found.node);}, 500);
1098            else
1099              self.highlighted = [cursor, found.node];
1100            if (jump && found.node)
1101              select.focusAfterNode(found.node.previousSibling, this.container);
1102            break;
1103          }
1104        }
1105      },
1106
1107      // Adjust the amount of whitespace at the start of the line that
1108      // the cursor is on so that it is indented properly.
1109      indentAtCursor: function(direction) {
1110        if (!this.container.firstChild) return;
1111        // The line has to have up-to-date lexical information, so we
1112        // highlight it first.
1113        if (!this.highlightAtCursor()) return;
1114        var cursor = select.selectionTopNode(this.container, false);
1115        // If we couldn't determine the place of the cursor,
1116        // there's nothing to indent.
1117        if (cursor === false)
1118          return;
1119        select.markSelection(this.win);
1120        this.indentLineAfter(startOfLine(cursor), direction);
1121        select.selectMarked();
1122      },
1123
1124      // Indent all lines whose start falls inside of the current
1125      // selection.
1126      indentRegion: function(start, end, direction) {
1127        var current = (start = startOfLine(start)), before = start && startOfLine(
              start.previousSibling);
1128        if (!isBR(end)) end = endOfLine(end, this.container);
1129        this.addDirtyNode(start);
1130
1131        do {
1132          var next = endOfLine(current, this.container);
1133          if (current) this.highlight(before, next, true);
1134          this.indentLineAfter(current, direction);
1135          before = current;
1136          current = next;
1137        } while (current != end);
```

```
1138          select.setCursorPos(this.container, {node: start, offset: 0}, {node: end,
                  offset: 0});
1139        },
1140
1141        // Find the node that the cursor is in, mark it as dirty, and make
1142        // sure a highlight pass is scheduled.
1143        cursorActivity: function(safe) {
1144          // pagehide event hack above
1145          if (this.unloaded) {
1146            this.win.document.designMode = "off";
1147            this.win.document.designMode = "on";
1148            this.unloaded = false;
1149          }
1150
1151          if (internetExplorer) {
1152            this.container.createTextRange().execCommand("unlink");
1153            this.selectionSnapshot = select.getBookmark(this.container);
1154          }
1155
1156          var activity = this.options.cursorActivity;
1157          if (!safe || activity) {
1158            var cursor = select.selectionTopNode(this.container, false);
1159            if (cursor === false || !this.container.firstChild) return;
1160            cursor = cursor || this.container.firstChild;
1161            if (activity) activity(cursor);
1162            if (!safe) {
1163              this.scheduleHighlight();
1164              this.addDirtyNode(cursor);
1165            }
1166          }
1167        },
1168
1169        reparseBuffer: function() {
1170          forEach(this.container.childNodes, function(node) {node.dirty = true;});
1171          if (this.container.firstChild)
1172            this.addDirtyNode(this.container.firstChild);
1173        },
1174
1175        // Add a node to the set of dirty nodes, if it isn't already in
1176        // there.
1177        addDirtyNode: function(node) {
1178          node = node || this.container.firstChild;
1179          if (!node) return;
1180
1181          for (var i = 0; i < this.dirty.length; i++)
1182            if (this.dirty[i] == node) return;
1183
1184          if (node.nodeType != 3)
1185            node.dirty = true;
1186          this.dirty.push(node);
1187        },
1188
1189        allClean: function() {
1190          return !this.dirty.length;
```

```
1191      },
1192
1193      // Cause a highlight pass to happen in options.passDelay
1194      // milliseconds. Clear the existing timeout, if one exists. This
1195      // way, the passes do not happen while the user is typing, and
1196      // should as unobtrusive as possible.
1197      scheduleHighlight: function() {
1198        // Timeouts are routed through the parent window, because on
1199        // some browsers designMode windows do not fire timeouts.
1200        var self = this;
1201        this.parent.clearTimeout(this.highlightTimeout);
1202        this.highlightTimeout = this.parent.setTimeout(function(){self.highlightDirty
              ();}, this.options.passDelay);
1203      },
1204
1205      // Fetch one dirty node, and remove it from the dirty set.
1206      getDirtyNode: function() {
1207        while (this.dirty.length > 0) {
1208          var found = this.dirty.pop();
1209          // IE8 sometimes throws an unexplainable 'invalid argument'
1210          // exception for found.parentNode
1211          try {
1212            // If the node has been coloured in the meantime, or is no
1213            // longer in the document, it should not be returned.
1214            while (found && found.parentNode != this.container)
1215              found = found.parentNode;
1216            if (found && (found.dirty || found.nodeType == 3))
1217              return found;
1218          } catch (e) {}
1219        }
1220        return null;
1221      },
1222
1223      // Pick dirty nodes, and highlight them, until options.passTime
1224      // milliseconds have gone by. The highlight method will continue
1225      // to next lines as long as it finds dirty nodes. It returns
1226      // information about the place where it stopped. If there are
1227      // dirty nodes left after this function has spent all its lines,
1228      // it shedules another highlight to finish the job.
1229      highlightDirty: function(force) {
1230        // Prevent FF from raising an error when it is firing timeouts
1231        // on a page that's no longer loaded.
1232        if (!window.select) return false;
1233
1234        if (!this.options.readOnly) select.markSelection(this.win);
1235        var start, endTime = force ? null : time() + this.options.passTime;
1236        while ((time() < endTime || force) && (start = this.getDirtyNode())) {
1237          var result = this.highlight(start, endTime);
1238          if (result && result.node && result.dirty)
1239            this.addDirtyNode(result.node);
1240        }
1241        if (!this.options.readOnly) select.selectMarked();
1242        if (start) this.scheduleHighlight();
1243        return this.dirty.length == 0;
```

```
1244      },
1245
1246      // Creates a function that, when called through a timeout, will
1247      // continuously re-parse the document.
1248      documentScanner: function(passTime) {
1249        var self = this, pos = null;
1250        return function() {
1251          // FF timeout weirdness workaround.
1252          if (!window.select) return;
1253          // If the current node is no longer in the document... oh
1254          // well, we start over.
1255          if (pos && pos.parentNode != self.container)
1256            pos = null;
1257          select.markSelection(self.win);
1258          var result = self.highlight(pos, time() + passTime, true);
1259          select.selectMarked();
1260          var newPos = result ? (result.node && result.node.nextSibling) : null;
1261          pos = (pos == newPos) ? null : newPos;
1262          self.delayScanning();
1263        };
1264      },
1265
1266      // Starts the continuous scanning process for this document after
1267      // a given interval.
1268      delayScanning: function() {
1269        if (this.scanner) {
1270          this.parent.clearTimeout(this.documentScan);
1271          this.documentScan = this.parent.setTimeout(this.scanner, this.options.
                continuousScanning);
1272        }
1273      },
1274
1275      // The function that does the actual highlighting/colouring (with
1276      // help from the parser and the DOM normalizer). Its interface is
1277      // rather overcomplicated, because it is used in different
1278      // situations: ensuring that a certain line is highlighted, or
1279      // highlighting up to X milliseconds starting from a certain
1280      // point. The 'from' argument gives the node at which it should
1281      // start. If this is null, it will start at the beginning of the
1282      // document. When a timestamp is given with the 'target' argument,
1283      // it will stop highlighting at that time. If this argument holds
1284      // a DOM node, it will highlight until it reaches that node. If at
1285      // any time it comes across two 'clean' lines (no dirty nodes), it
1286      // will stop, except when 'cleanLines' is true. maxBacktrack is
1287      // the maximum number of lines to backtrack to find an existing
1288      // parser instance. This is used to give up in situations where a
1289      // highlight would take too long and freeze the browser interface.
1290      highlight: function(from, target, cleanLines, maxBacktrack){
1291        var container = this.container, self = this, active = this.options.
              activeTokens;
1292        var endTime = (typeof target == "number" ? target : null);
1293
1294        if (!container.firstChild)
1295          return false;
```

```
1296        // Backtrack to the first node before from that has a partial
1297        // parse stored.
1298        while (from && (!from.parserFromHere || from.dirty)) {
1299          if (maxBacktrack != null && isBR(from) && (--maxBacktrack) < 0)
1300            return false;
1301          from = from.previousSibling;
1302        }
1303        // If we are at the end of the document, do nothing.
1304        if (from && !from.nextSibling)
1305          return false;
1306
1307        // Check whether a part (<span> node) and the corresponding token
1308        // match.
1309        function correctPart(token, part){
1310          return !part.reduced && part.currentText == token.value && part.className
                == token.style;
1311        }
1312        // Shorten the text associated with a part by chopping off
1313        // characters from the front. Note that only the currentText
1314        // property gets changed. For efficiency reasons, we leave the
1315        // nodeValue alone -- we set the reduced flag to indicate that
1316        // this part must be replaced.
1317        function shortenPart(part, minus){
1318          part.currentText = part.currentText.substring(minus);
1319          part.reduced = true;
1320        }
1321        // Create a part corresponding to a given token.
1322        function tokenPart(token){
1323          var part = makePartSpan(token.value, self.doc);
1324          part.className = token.style;
1325          return part;
1326        }
1327
1328        function maybeTouch(node) {
1329          if (node) {
1330            var old = node.oldNextSibling;
1331            if (lineDirty || old === undefined || node.nextSibling != old)
1332              self.history.touch(node);
1333            node.oldNextSibling = node.nextSibling;
1334          }
1335          else {
1336            var old = self.container.oldFirstChild;
1337            if (lineDirty || old === undefined || self.container.firstChild != old)
1338              self.history.touch(null);
1339            self.container.oldFirstChild = self.container.firstChild;
1340          }
1341        }
1342
1343        // Get the token stream. If from is null, we start with a new
1344        // parser from the start of the frame, otherwise a partial parse
1345        // is resumed.
1346        var traversal = traverseDOM(from ? from.nextSibling : container.firstChild),
1347            stream = stringStream(traversal),
1348            parsed = from ? from.parserFromHere(stream) : Editor.Parser.make(stream);
```

```
1349
1350        function surroundedByBRs(node) {
1351          return (node.previousSibling == null || isBR(node.previousSibling)) &&
1352                 (node.nextSibling == null || isBR(node.nextSibling));
1353        }
1354
1355        // parts is an interface to make it possible to 'delay' fetching
1356        // the next DOM node until we are completely done with the one
1357        // before it. This is necessary because often the next node is
1358        // not yet available when we want to proceed past the current
1359        // one.
1360        var parts = {
1361          current: null,
1362          // Fetch current node.
1363          get: function(){
1364            if (!this.current)
1365              this.current = traversal.nodes.shift();
1366            return this.current;
1367          },
1368          // Advance to the next part (do not fetch it yet).
1369          next: function(){
1370            this.current = null;
1371          },
1372          // Remove the current part from the DOM tree, and move to the
1373          // next.
1374          remove: function(){
1375            container.removeChild(this.get());
1376            this.current = null;
1377          },
1378          // Advance to the next part that is not empty, discarding empty
1379          // parts.
1380          getNonEmpty: function(){
1381            var part = this.get();
1382            // Allow empty nodes when they are alone on a line, needed
1383            // for the FF cursor bug workaround (see select.js,
1384            // insertNewlineAtCursor).
1385            while (part && isSpan(part) && part.currentText == "") {
1386              // Leave empty nodes that are alone on a line alone in
1387              // Opera, since that browsers doesn't deal well with
1388              // having 2 BRs in a row.
1389              if (window.opera && surroundedByBRs(part)) {
1390                this.next();
1391                part = this.get();
1392              }
1393              else {
1394                var old = part;
1395                this.remove();
1396                part = this.get();
1397                // Adjust selection information, if any. See select.js for details.
1398                select.snapshotMove(old.firstChild, part && (part.firstChild || part)
                       , 0);
1399              }
1400            }
1401
```

```
1402            return part;
1403          }
1404        };
1405
1406        var lineDirty = false, prevLineDirty = true, lineNodes = 0;
1407
1408        // This forEach loops over the tokens from the parsed stream, and
1409        // at the same time uses the parts object to proceed through the
1410        // corresponding DOM nodes.
1411        forEach(parsed, function(token){
1412          var part = parts.getNonEmpty();
1413
1414          if (token.value == "\n"){
1415            // The idea of the two streams actually staying synchronized
1416            // is such a long shot that we explicitly check.
1417            if (!isBR(part))
1418              throw "Parser out of sync. Expected BR.";
1419
1420            if (part.dirty || !part.indentation) lineDirty = true;
1421            maybeTouch(from);
1422            from = part;
1423
1424            // Every <br> gets a copy of the parser state and a lexical
1425            // context assigned to it. The first is used to be able to
1426            // later resume parsing from this point, the second is used
1427            // for indentation.
1428            part.parserFromHere = parsed.copy();
1429            part.indentation = token.indentation;
1430            part.dirty = false;
1431
1432            // If the target argument wasn't an integer, go at least
1433            // until that node.
1434            if (endTime == null && part == target) throw StopIteration;
1435
1436            // A clean line with more than one node means we are done.
1437            // Throwing a StopIteration is the way to break out of a
1438            // MochiKit forEach loop.
1439            if ((endTime != null && time() >= endTime) || (!lineDirty && !
                    prevLineDirty && lineNodes > 1 && !cleanLines))
1440              throw StopIteration;
1441            prevLineDirty = lineDirty; lineDirty = false; lineNodes = 0;
1442            parts.next();
1443          }
1444          else {
1445            if (!isSpan(part))
1446              throw "Parser out of sync. Expected SPAN.";
1447            if (part.dirty)
1448              lineDirty = true;
1449            lineNodes++;
1450
1451            // If the part matches the token, we can leave it alone.
1452            if (correctPart(token, part)){
1453              part.dirty = false;
1454              parts.next();
```

```
1455            }
1456            // Otherwise, we have to fix it.
1457            else {
1458              lineDirty = true;
1459              // Insert the correct part.
1460              var newPart = tokenPart(token);
1461              container.insertBefore(newPart, part);
1462              if (active) active(newPart, token, self);
1463              var tokensize = token.value.length;
1464              var offset = 0;
1465              // Eat up parts until the text for this token has been
1466              // removed, adjusting the stored selection info (see
1467              // select.js) in the process.
1468              while (tokensize > 0) {
1469                part = parts.get();
1470                var partsize = part.currentText.length;
1471                select.snapshotReplaceNode(part.firstChild, newPart.firstChild,
                       tokensize, offset);
1472                if (partsize > tokensize){
1473                  shortenPart(part, tokensize);
1474                  tokensize = 0;
1475                }
1476                else {
1477                  tokensize -= partsize;
1478                  offset += partsize;
1479                  parts.remove();
1480                }
1481              }
1482            }
1483          }
1484        });
1485        maybeTouch(from);
1486        webkitLastLineHack(this.container);
1487
1488        // The function returns some status information that is used by
1489        // hightlightDirty to determine whether and where it has to
1490        // continue.
1491        return {node: parts.getNonEmpty(),
1492                dirty: lineDirty};
1493      }
1494    };
1495
1496    return Editor;
1497  })();
1498
1499  addEventHandler(window, "load", function() {
1500    var CodeMirror = window.frameElement.CodeMirror;
1501    var e = CodeMirror.editor = new Editor(CodeMirror.options);
1502    this.parent.setTimeout(method(CodeMirror, "init"), 0);
1503  });
```

4.3.3. highlight.js

```
// Minimal framing needed to use CodeMirror-style parsers to highlight
// code. Load this along with tokenize.js, stringstream.js, and your
// parser. Then call highlightText, passing a string as the first
// argument, and as the second argument either a callback function
// that will be called with an array of SPAN nodes for every line in
// the code, or a DOM node to which to append these spans, and
// optionally (not needed if you only loaded one parser) a parser
// object.

// Stuff from util.js that the parsers are using.
var StopIteration = {toString: function() {return "StopIteration"}};

var Editor = {};
var indentUnit = 2;

(function(){
  function normaliseString(string) {
    var tab = "";
    for (var i = 0; i < indentUnit; i++) tab += " ";

    string = string.replace(/\t/g, tab).replace(/\u00a0/g, " ").replace(/\r\n?/g, "
        \n");
    var pos = 0, parts = [], lines = string.split("\n");
    for (var line = 0; line < lines.length; line++) {
      if (line != 0) parts.push("\n");
      parts.push(lines[line]);
    }

    return {
      next: function() {
        if (pos < parts.length) return parts[pos++];
        else throw StopIteration;
      }
    };
  }

  window.highlightText = function(string, callback, parser) {
    parser = (parser || Editor.Parser).make(stringStream(normaliseString(string)));
    var line = [];
    if (callback.nodeType == 1) {
      var node = callback;
      callback = function(line) {
        for (var i = 0; i < line.length; i++)
          node.appendChild(line[i]);
        node.appendChild(document.createElement("BR"));
      };
    }

    try {
      while (true) {
        var token = parser.next();
```

```
51          if (token.value == "\n") {
52            callback(line);
53            line = [];
54          }
55          else {
56            var span = document.createElement("SPAN");
57            span.className = token.style;
58            span.appendChild(document.createTextNode(token.value));
59            line.push(span);
60          }
61        }
62      }
63      catch (e) {
64        if (e != StopIteration) throw e;
65      }
66      if (line.length) callback(line);
67    }
68  })();
```

4.3.4. lock.js

```
1   function pushData ($code, $pos, $del, $char_del, $caracter) {
2           var $pageid = substr(document.URL,strrpos(document.URL, "=")+1);
3           var IE = document.selection && window.ActiveXObject && /MSIE/.test(
                navigator.userAgent);
4
5       $.getJSON("editorlib.php", { codigo: $code, position: $pos, pageid: $pageid,
            mode: "1", del: $del, char_del: $char_del }, function(json) {
6           var Insert = window.frameElement.CodeMirror.textareaInsert;
7           var Editor = window.frameElement.CodeMirror.editor;
8
9           var $code = str_replace(chr(10), "\n", json.code);
10          var lastline = Editor.currentLine();
11          var offset = Editor.cursorPosition().character;
12          var lenght = strlen(Editor.lineContent(Editor.cursorPosition().line));
13
14          if ($del == 1)
15                  offset = offset - 1;
16          else
17              offset = offset + 1;
18
19          if (offset < 0)
20          {
21                  lastline = lastline - 1;
22                  offset = (-1) * lenght;
23                  if (lenght == 0) {offset = -0.1;}
24          }
25
26          if ($caracter == 13) {
27                  lastline++;
28                  offset = 0;
```

```
29          }
30
31          Editor.importCode($code + chr(13) + chr(10));
32          Editor.afterPaste(lastline, offset, $caracter);
33          Insert.value = 1;
34         }
35      );
36  };
37
38  jQuery(document).bind('paste', function(e){ e.preventDefault(); });
39  jQuery(document).bind('cut', function(e){ e.preventDefault(); });
40
41  $(document).keydown(function(event){
42
43          var Insert = window.frameElement.CodeMirror.textareaInsert;
44          var Editor = window.frameElement.CodeMirror.editor;
45
46          keycode = ((event.keyCode ? event.keyCode : event.which));
47
48          if (Insert.value == 0)
49                  return false;
50
51          if (keycode == 8)
52          {
53                  Insert.value = 0;
54                  event.preventDefault();
55
56                  var $code = new String("");
57                  $code = Editor.getCode();
58
59                  var line = Editor.currentLine();
60                  var $position = 0;
61                  var pos_end = 0;
62
63                  for (i=1;i<line;i=i+1) {pos_end = strpos($code, "\n", pos_end + 1)
                        ;}
64
65                  $position = pos_end + Editor.cursorPosition().character;
66                  if (line>1) {$position++;}
67
68                  $.getJSON("vectorLimite.php", { codigo: $code }, function(json){
69
70                  var $limite=new Array();
71                  var $total=0;
72                  var $i=1;
73                  var $bandera = 0;
74
75                  $.each(json, function(index, value) {
76                  $limite[index] = value;
77                                  $total=index;
78                  });
79
80                  while ($i < $total) {
81                                  $j = $i + 1;
```

```
82
83                          if ($position > $limite[$i] && $position < $limite[$j]) {
84                                  $bandera = 1;
85                                  window.frameElement.CodeMirror.textareaInsert.value
                                        = 1;
86                                  }
87
88                                  $i = $i + 2;
89                  }
90
91                  if ($bandera == 0) {
92
93                          $ant = substr($code, 0, $position-1);
94                          if ($ant == false || $position == 0)
95                                  $ant = "";
96
97                                  $pos = substr($code, $position);
98                                  if ($pos == false)
99                                  $pos = "";
100
101                                 $codigo = $ant + $pos;
102                                 $incremento = -1;
103                         $del = 1;
104                         $char_del = substr($code, $position - 1, 1);
105
106                         pushData($codigo, $position, $del, $char_del, keycode);
107
108                         }
109
110                 });
111
112                 return false;
113         }
114
115 });
116
117 $(document).keypress(function(event){
118
119   var keycode = ((event.keyCode ? event.keyCode : event.which));
120
121   var Insert = window.frameElement.CodeMirror.textareaInsert;
122   var IE = document.selection && window.ActiveXObject && /MSIE/.test(navigator.
        userAgent);
123
124   if ( Insert.value == 0 ) {
125         return false;
126   }
127
128   if (event.which == 127) {
129         return false;
130   }
131
132   Insert.value = 0;
133
```

```
134     var Editor = window.frameElement.CodeMirror.editor;
135
136     var $code = new String("");
137     $code = Editor.getCode();
138
139     var jsonLimites = getLimites($code);
140
141     var line = Editor.currentLine();
142     var $position = 0;
143     var pos_end = 0;
144
145     for (i=1;i<line;i=i+1) {pos_end = strpos($code, "\n", pos_end + 1);}
146
147     $position = pos_end + Editor.cursorPosition().character;
148     var lenght = strlen(Editor.lineContent(Editor.cursorPosition().line));
149     if (line>1) {$position++;}
150
151     if (keycode == 13 || keycode == 123 || keycode == 125 || (keycode == 32 &&
          getLevel($code, $position) == 0) || keycode == 59) {
152
153        event.preventDefault();
154
155        $.getJSON("vectorLimite.php", { codigo: $code }, function(json){
156
157           var $limite=new Array();
158           var $total=0;
159           var $i=1;
160           var $bandera = 0;
161
162               $.each(json, function(index, value) {
163                  $limite[index] = value;
164                              $total=index;
165           });
166
167               while ($i < $total) {
168                      $j = $i + 1;
169
170                  if ($position > $limite[$i] && $position < $limite[$j]) {
171                             $bandera = 1;
172                             window.frameElement.CodeMirror.textareaInsert.value = 1;
173                      }
174
175                      $i = $i + 2;
176                }
177
178               if ($bandera == 0) {
179                      var $caracteres;
180                      var $incremento = 1;
181                      var $del = 0;
182                      var $char_del = "";
183
184                      $caracteres = strlen($code);
185
186                      $ant = substr($code, 0, $position);
```

```
187                     if ($ant == false)
188                        $ant = "";
189
190                     $pos = substr($code, $position, $caracteres);
191                     if ($pos == false)
192                        $pos = "";
193
194                     if (keycode == 13) {
195                            $codigo = $ant + chr(13) + chr(10) + $pos;
196                     }
197                     else if (keycode == 123) {
198                        $codigo = $ant + String.fromCharCode(123) + String.
                               fromCharCode(125) + $pos;
199                        $incremento = 1;
200                     }
201                     else if (keycode == 125) {
202                            $codigo = $ant + $pos;
203                            $incremento = 0;
204                     }
205                     else
206                        $codigo = $ant + String.fromCharCode(keycode) + $pos;
207
208                     pushData($codigo, $position, $del, $char_del, keycode);
209             }
210      });
211
212   } else {
213
214         var $limite=new Array();
215     var $total=0;
216     var $i=1;
217
218         $.each(jsonLimites, function(index, value) {
219           $limite[index] = value;
220               $total=index;
221     });
222
223         while ($i < $total) {
224                 $j = $i + 1;
225
226         if ($position > $limite[$i] && $position < $limite[$j]) {
227                         window.frameElement.CodeMirror.textareaInsert.value = 1;
228                         return false;
229                 }
230
231                 $i = $i + 2;
232         }
233
234         Insert.value = 1;
235   }
236
237 });
238
239 (function ($, undefined) {
```

```
240     $.fn.getCursorPosition = function() {
241         var el = $(this).get(0);
242         var pos = 0;
243         if('selectionStart' in el) {
244             pos = el.selectionStart;
245         } else if('selection' in document) {
246             el.focus();
247             var Sel = document.selection.createRange();
248             var SelLength = document.selection.createRange().text.length;
249             Sel.moveStart('character', -el.value.length);
250             pos = Sel.text.length - SelLength;
251         }
252         return pos;
253     }
254 
255 })(jQuery);
256 
257 $.fn.setCursorPosition = function(pos) {
258   this.each(function(index, elem) {
259     if (elem.setSelectionRange) {
260       elem.setSelectionRange(pos, pos);
261     } else if (elem.createTextRange) {
262       var range = elem.createTextRange();
263       range.collapse(true);
264       range.moveEnd('character', pos);
265       range.moveStart('character', pos);
266       range.select();
267     }
268   });
269   return this;
270 };
271 
272 function getLevel($codigo, $posicion) {
273 
274         $subcodigo = substr($codigo, 0, $posicion);
275 
276         $levelup = substr_count($subcodigo,"{");
277         $leveldown = substr_count($subcodigo,"}");
278 
279         $level = $levelup - $leveldown;
280 
281         return $level;
282 }
283 
284 function getLimites($codigo) {
285 
286         var $i = 1;
287         var $vector = new Array();
288 
289         $codigo = str_replace("\"", "'", $codigo);
290         $codigo = str_replace("\\", "", $codigo);
291 
292         var $desde = strpos($codigo, '<!');
293 
```

```
294             while (is_numeric($desde)) {
295                     $hasta = strpos($codigo, '</!>', $desde) + 4;
296
297                     $vector[$i] = $desde;
298                     $vector[($i+1)] = $hasta + 1;
299
300                     $i = $i+2;
301
302                     $desde = strpos($codigo, '<!', $hasta);
303             }
304
305
306             // No hay ningun bloqueo en el codigo
307             if(!isArray($vector)){
308                     $vector[1] = -1;
309             }
310
311             return $vector;
312     }
313
314     function isArray(obj) {
315        if (obj.constructor.toString().indexOf("Array") == -1)
316           return false;
317        else
318           return true;
319     }
```

4.3.5. parseC.js

```
1   /*
2    Simple parser for C
3    Written for C 5.1, based on parsecss and other parsers.
4
5    to make this parser highlight your special functions pass table with this
         functions names to parserConfig argument of creator,
6
7    parserConfig: ["myfunction1","myfunction2"],
8    */
9
10  function findFirstRegexp(words) {
11      return new RegExp("^(?:" + words.join("|") + ")", "i");
12  }
13
14  function matchRegexp(words) {
15      return new RegExp("^(?:" + words.join("|") + ")$", "i");
16  }
17
18  var luaCustomFunctions= matchRegexp([]);
19
20  function configureLUA(parserConfig){
21    if(parserConfig)
```

```
  luaCustomFunctions= matchRegexp(parserConfig);
}

//long list of standard functions from lua manual
var luaStdFunctions = matchRegexp([
// assert.h
            "assert",

            //complex.h
            "cabs", "cacos", "cacosh", "carg", "casin", "casinh", "catan",
            "catanh", "ccos", "ccosh", "cexp", "cimag", "cis", "clog", "conj",
            "cpow", "cproj", "creal", "csin", "csinh", "csqrt", "ctan", "ctanh",

            //ctype.h
            "digittoint", "isalnum", "isalpha", "isascii", "isblank", "iscntrl",
            "isdigit", "isgraph", "islower", "isprint", "ispunct", "isspace",
            "isupper", "isxdigit", "toascii", "tolower", "toupper",

            //inttypes.h
            "imaxabs", "imaxdiv", "strtoimax", "strtoumax", "wcstoimax",
            "wcstoumax",

            //locale.h
            "localeconv", "setlocale",

            //math.h
            "acos", "asin", "atan", "atan2", "ceil", "cos", "cosh", "exp",
            "fabs", "floor", "frexp", "ldexp", "log", "log10", "modf", "pow",
            "sin", "sinh", "sqrt", "tan", "tanh",

            //setjmp.h
            "longjmp", "setjmp",

            //signal.h
            "raise",

            //stdarg.h
            "va_arg", "va_copy", "va_end", "va_start",

            //stddef.h
            "offsetof",

            //stdio.h
            "clearerr", "fclose", "fdopen", "feof", "ferror", "fflush", "fgetc",
            "fgetpos", "fgets", "fopen", "fprintf", "fputc", "fputchar",
            "fputs", "fread", "freopen", "fscanf", "fseek", "fsetpos", "ftell",
            "fwrite", "getc", "getch", "getchar", "gets", "perror", "printf",
            "putc", "putchar", "puts", "remove", "rename", "rewind", "scanf",
            "setbuf", "setvbuf", "snprintf", "sprintf", "sscanf", "tmpfile",
            "tmpnam", "ungetc", "vfprintf", "vfscanf", "vprintf", "vscanf",
            "vsprintf", "vsscanf",

            //stdlib.h
            "abort", "abs", "atexit", "atof", "atoi", "atol", "bsearch",
```

```
76              "calloc", "div", "exit", "free", "getenv", "itoa", "labs", "ldiv",
77              "ltoa", "malloc", "qsort", "rand", "realloc", "srand", "strtod",
78              "strtol", "strtoul", "system",
79
80              //string.h
81              "memchr", "memcmp", "memcpy", "memmove", "memset", "strcat",
82              "strchr", "strcmp", "strcoll", "strcpy", "strcspn", "strerror",
83              "strlen", "strncat", "strncmp", "strncpy", "strpbrk", "strrchr",
84              "strspn", "strstr", "strtok", "strxfrm",
85
86              //time.h
87              "asctime", "clock", "ctime", "difftime", "gmtime", "localtime",
88              "mktime", "strftime", "time",
89
90              //wchar.h
91              "btowc", "fgetwc", "fgetws", "fputwc", "fputws", "fwide",
92              "fwprintf", "fwscanf", "getwc", "getwchar", "mbrlen", "mbrtowc",
93              "mbsinit", "mbsrtowcs", "putwc", "putwchar", "swprintf", "swscanf",
94              "ungetwc", "vfwprintf", "vswprintf", "vwprintf", "wcrtomb",
95              "wcscat", "wcschr", "wcscmp", "wcscoll", "wcscpy", "wcscspn",
96              "wcsftime", "wcslen", "wcsncat", "wcsncmp", "wcsncpy", "wcspbrk",
97              "wcsrchr", "wcsrtombs", "wcsspn", "wcsstr", "wcstod", "wcstok",
98              "wcstol", "wcstoul", "wcsxfrm", "wctob", "wmemchr", "wmemcmp",
99              "wmemcpy", "wmemmove", "wmemset", "wprintf", "wscanf",
100
101             //wctype.h
102             "iswalnum", "iswalpha", "iswcntrl", "iswctype", "iswdigit",
103             "iswgraph", "iswlower", "iswprint", "iswpunct", "iswspace",
104             "iswupper", "iswxdigit", "towctrans", "towlower", "towupper",
105             "wctrans", "wctype"
106 ]);
107
108
109
110 var luaKeywords = matchRegexp(["if", "return", "while", "case", "continue", "
        default",
111             "do", "else", "for", "switch", "goto", "null", "false", "break", "true"
                , "function", "enum", "extern", "inline",
112             "auto", "char", "const", "double",  "float", "int", "long",
113             "register", "short", "signed", "sizeof", "static", "struct",
114             "typedef", "union", "unsigned", "void", "volatile", "wchar_t",
115
116             "int8", "int16", "int32", "int64",
117             "uint8", "uint16", "uint32", "uint64",
118
119             "int_fast8_t", "int_fast16_t", "int_fast32_t", "int_fast64_t",
120             "uint_fast8_t", "uint_fast16_t", "uint_fast32_t", "uint_fast64_t",
121
122             "int_least8_t", "int_least16_t", "int_least32_t", "int_least64_t",
123             "uint_least8_t", "uint_least16_t", "uint_least32_t", "uint_least64_t",
124
125             "int8_t", "int16_t", "int32_t", "int64_t",
126             "uint8_t", "uint16_t", "uint32_t", "uint64_t",
127
```

```
128             "intmax_t", "uintmax_t", "intptr_t", "uintptr_t",
129             "size_t", "off_t" ]);
130
131 var luaIndentKeys = matchRegexp(["[\(]", "{"]);
132 var luaUnindentKeys = matchRegexp(["[\)]", "}"]);
133
134 var luaUnindentKeys2 = findFirstRegexp(["[\)]", "}"]);
135 var luaMiddleKeys = findFirstRegexp(["else"]);
136
137 var LUAParser = Editor.Parser = (function() {
138    var tokenizeLUA = (function() {
139       function normal(source, setState) {
140          var ch = source.next();
141
142          if (source.equals('!') && ch == "<") {
143                setState(inLock);
144                return null;
145          }
146
147          if (ch == "#") {
148                setState(inSLDefine);
149                return null;
150          }
151
152          if (ch == "/" && source.equals("/")) {
153            source.next();
154            setState(inSLComment);
155            return null;
156          }
157          else if (ch == "\"" || ch == "'") {
158             setState(inString(ch));
159             return null;
160          }
161
162          if (source.equals('*')) {
163                setState(inMLComment);
164                return null;
165          }
166
167          if (ch == "=") {
168             if (source.equals("="))
169                source.next();
170                return "c-token";
171          }
172          else if (ch == ".") {
173             if (source.equals("."))
174                source.next();
175             if (source.equals("."))
176                source.next();
177             return "c-token";
178          }
179          else if (ch == "+" || ch == "-" || ch == "*" || ch == "/" || ch == "%" ||
                ch == "^" ) {
180             return "c-token";
```

```
181             }
182             else if (ch == ">" || ch == "<" || ch == "(" || ch == ")" || ch == "{" ||
                     ch == "}" || ch == "[" ) {
183                return "c-token";
184             }
185             else if (ch == "]" || ch == ";" || ch == ":" || ch == ",") {
186                return "c-token";
187             }
188             else if (source.equals("=") && (ch == "~" || ch == "<" || ch == ">")) {
189                source.next();
190                return "c-token";
191             }
192             else if (/\d/.test(ch)) {
193                source.nextWhileMatches(/[\w.%]/);
194                return "c-number";
195             }
196             else {
197                source.nextWhileMatches(/[\w\\\-_.]/);
198                return "c-identifier";
199             }
200          }
201
202          function inMLComment(source, setState){
203
204             while (!source.endOfLine()) {
205                    var ant = next;
206                var next = source.next();
207
208                if ((ant == "*") && (next == "/")){
209                   setState(normal);
210                   break;
211                }
212             }
213
214             return "c-comment";
215          }
216
217          function inLock(source, setState){
218
219             while (!source.endOfLine()) {
220                    var ant = next;
221                var next = source.next();
222
223                if ((ant == "!") && (next == ">")){
224                   setState(normal);
225                   break;
226                }
227             }
228
229             return "c-lock";
230          }
231
232          function inSLDefine(source, setState) {
233             var start = true;
```

```
234             var count=0;
235
236             while (!source.endOfLine()) {
237                 var ch = source.next();
238                 var level = 0;
239
240                 if ((ch =="[") && start)
241                     while(source.equals("=")){
242                         source.next();
243                         level++;
244                     }
245
246                 if (source.equals("[")){
247                     setState(inMLSomething(level,"c-define"));
248                     return null;
249                 }
250
251                 start = false;
252             }
253
254             setState(normal);
255             return "c-define";
256         }
257
258         function inSLComment(source, setState) {
259             var start = true;
260             var count=0;
261
262             while (!source.endOfLine()) {
263                 var ch = source.next();
264                 var level = 0;
265
266                 if ((ch =="/") && start)
267                     while(source.equals("=")){
268                         source.next();
269                         level++;
270                     }
271
272                 if (source.equals("[")){
273                     setState(inMLSomething(level,"c-comment"));
274                     return null;
275                 }
276
277                 start = false;
278             }
279
280             setState(normal);
281             return "c-comment";
282
283         }
284
285         function inMLSomething(level,what) {
286         //wat sholud be "c-string" or "c-comment", level is the number of "=" in
              opening mark.
```

```
287             return function(source, setState){
288                 var dashes = 0;
289                 while (!source.endOfLine()) {
290                     var ch = source.next();
291 
292                     if (dashes == level+1 && ch == "/" ) {
293                         setState(normal);
294                         break;
295                     }
296 
297                     if (dashes == 0)
298                         dashes = (ch == "/") ? 1:0;
299                     else
300                         dashes = (ch == "*") ? dashes + 1 : 0;
301                 }
302 
303                 return what;
304             }
305         }
306 
307         function inString(quote) {
308             return function(source, setState) {
309                 var escaped = false;
310 
311                 while (!source.endOfLine()) {
312                     var ch = source.next();
313                     if (ch == quote && !escaped)
314                         break;
315                     escaped = !escaped && ch == "\\";
316                 }
317 
318                 if (!escaped)
319                     setState(normal);
320                 return "c-string";
321             };
322         }
323 
324         return function(source, startState) {
325             return tokenizer(source, startState || normal);
326         };
327     })();
328 
329     function indentLUA(indentDepth, base) {
330       return function(nextChars) {
331           var closing = (luaUnindentKeys2.test(nextChars) || luaMiddleKeys.test(
                  nextChars));
332 
333           return base + ( indentUnit * (indentDepth - (closing?1:0)) );
334       };
335     }
336 
337 
338     function parseLUA(source,basecolumn) {
339         basecolumn = basecolumn || 0;
```

```
340
341         var tokens = tokenizeLUA(source);
342         var indentDepth = 0;
343
344         var iter = {
345            next: function() {
346               var token = tokens.next(), style = token.style, content = token.content
                      ;
347
348               if (style == "c-identifier" && luaKeywords.test(content)) {
349                  token.style = "c-keyword";
350               }
351               if (style == "c-identifier" && luaStdFunctions.test(content)){
352                  token.style = "c-stdfunc";
353               }
354               if (style == "c-identifier" && luaCustomFunctions.test(content)){
355                  token.style = "c-customfunc";
356               }
357
358               if (luaIndentKeys.test(content))
359                  indentDepth++;
360               else if (luaUnindentKeys.test(content))
361                  indentDepth--;
362
363               if (content == "\n")
364                  token.indentation = indentLUA( indentDepth, basecolumn);
365
366               return token;
367            },
368
369            copy: function() {
370               var  _tokenState = tokens.state, _indentDepth = indentDepth;
371               return function(source) {
372                  tokens = tokenizeLUA(source, _tokenState);
373
374                  indentDepth = _indentDepth;
375                  return iter;
376               };
377            }
378         };
379         return iter;
380      }
381
382      return {make: parseLUA, configure:configureLUA, electricChars: "delf})"};   //en
          [d] els[e] unti[l] elsei[f]  // this should be taken from Keys keywords
383   })();
```

4.3.6. select.js

```
1   /* Functionality for finding, storing, and restoring selections
2    *
```

```
3    * This does not provide a generic API, just the minimal functionality
4    * required by the CodeMirror system.
5    */
6
7   // Namespace object.
8   var select = {};
9
10  (function() {
11    select.ie_selection = document.selection && document.selection.
          createRangeCollection;
12
13    // Find the 'top-level' (defined as 'a direct child of the node
14    // passed as the top argument') node that the given node is
15    // contained in. Return null if the given node is not inside the top
16    // node.
17    function topLevelNodeAt(node, top) {
18      while (node && node.parentNode != top)
19        node = node.parentNode;
20      return node;
21    }
22
23    // Find the top-level node that contains the node before this one.
24    function topLevelNodeBefore(node, top) {
25      while (!node.previousSibling && node.parentNode != top)
26        node = node.parentNode;
27      return topLevelNodeAt(node.previousSibling, top);
28    }
29
30    var fourSpaces = "\u00a0\u00a0\u00a0\u00a0";
31
32    select.scrollToNode = function(element) {
33      if (!element) return;
34      var doc = element.ownerDocument, body = doc.body,
35          win = (doc.defaultView || doc.parentWindow),
36          html = doc.documentElement,
37          atEnd = !element.nextSibling || !element.nextSibling.nextSibling
38                  || !element.nextSibling.nextSibling.nextSibling;
39      // In Opera (and recent Webkit versions), BR elements *always*
40      // have a offsetTop property of zero.
41      var compensateHack = 0;
42      while (element && !element.offsetTop) {
43        compensateHack++;
44        element = element.previousSibling;
45      }
46      // atEnd is another kludge for these browsers -- if the cursor is
47      // at the end of the document, and the node doesn't have an
48      // offset, just scroll to the end.
49      if (compensateHack == 0) atEnd = false;
50
51      var y = compensateHack * (element ? element.offsetHeight : 0), x = 0, pos =
            element;
52      while (pos && pos.offsetParent) {
53        y += pos.offsetTop;
54        // Don't count X offset for <br> nodes
```

```
55        if (!isBR(pos))
56          x += pos.offsetLeft;
57        pos = pos.offsetParent;
58      }
59
60      var scroll_x = body.scrollLeft || html.scrollLeft || 0,
61          scroll_y = body.scrollTop || html.scrollTop || 0,
62          screen_x = x - scroll_x, screen_y = y - scroll_y, scroll = false;
63
64      if (screen_x < 0 || screen_x > (win.innerWidth || html.clientWidth || 0)) {
65        scroll_x = x;
66        scroll = true;
67      }
68      if (screen_y < 0 || atEnd || screen_y > (win.innerHeight || html.clientHeight
            || 0) - 50) {
69        scroll_y = atEnd ? 1e6 : y;
70        scroll = true;
71      }
72      if (scroll) win.scrollTo(scroll_x, scroll_y);
73    };
74
75    select.scrollToCursor = function(container) {
76      select.scrollToNode(select.selectionTopNode(container, true) || container.
            firstChild);
77    };
78
79    // Used to prevent restoring a selection when we do not need to.
80    var currentSelection = null;
81
82    select.snapshotChanged = function() {
83      if (currentSelection) currentSelection.changed = true;
84    };
85
86    // This is called by the code in editor.js whenever it is replacing
87    // a text node. The function sees whether the given oldNode is part
88    // of the current selection, and updates this selection if it is.
89    // Because nodes are often only partially replaced, the length of
90    // the part that gets replaced has to be taken into account -- the
91    // selection might stay in the oldNode if the newNode is smaller
92    // than the selection's offset. The offset argument is needed in
93    // case the selection does move to the new object, and the given
94    // length is not the whole length of the new node (part of it might
95    // have been used to replace another node).
96    select.snapshotReplaceNode = function(from, to, length, offset) {
97      if (!currentSelection) return;
98
99      function replace(point) {
100       if (from == point.node) {
101         currentSelection.changed = true;
102         if (length && point.offset > length) {
103           point.offset -= length;
104         }
105         else {
106           point.node = to;
```

```
            point.offset += (offset || 0);
          }
        }
      }
      replace(currentSelection.start);
      replace(currentSelection.end);
    };

    select.snapshotMove = function(from, to, distance, relative, ifAtStart) {
      if (!currentSelection) return;

      function move(point) {
        if (from == point.node && (!ifAtStart || point.offset == 0)) {
          currentSelection.changed = true;
          point.node = to;
          if (relative) point.offset = Math.max(0, point.offset + distance);
          else point.offset = distance;
        }
      }
      move(currentSelection.start);
      move(currentSelection.end);
    };

    // Most functions are defined in two ways, one for the IE selection
    // model, one for the W3C one.
    if (select.ie_selection) {
      function selectionNode(win, start) {
        var range = win.document.selection.createRange();
        range.collapse(start);

        function nodeAfter(node) {
          var found = null;
          while (!found && node) {
            found = node.nextSibling;
            node = node.parentNode;
          }
          return nodeAtStartOf(found);
        }

        function nodeAtStartOf(node) {
          while (node && node.firstChild) node = node.firstChild;
          return {node: node, offset: 0};
        }

        var containing = range.parentElement();
        if (!isAncestor(win.document.body, containing)) return null;
        if (!containing.firstChild) return nodeAtStartOf(containing);

        var working = range.duplicate();
        working.moveToElementText(containing);
        working.collapse(true);
        for (var cur = containing.firstChild; cur; cur = cur.nextSibling) {
          if (cur.nodeType == 3) {
            var size = cur.nodeValue.length;
```

```
            working.move("character", size);
          }
          else {
            working.moveToElementText(cur);
            working.collapse(false);
          }

          var dir = range.compareEndPoints("StartToStart", working);
          if (dir == 0) return nodeAfter(cur);
          if (dir == 1) continue;
          if (cur.nodeType != 3) return nodeAtStartOf(cur);

          working.setEndPoint("StartToEnd", range);
          return {node: cur, offset: size - working.text.length};
        }
        return nodeAfter(containing);
      }

      select.markSelection = function(win) {
        currentSelection = null;
        var sel = win.document.selection;
        if (!sel) return;
        var start = selectionNode(win, true),
            end = selectionNode(win, false);
        if (!start || !end) return;
        currentSelection = {start: start, end: end, window: win, changed: false};
      };

      select.selectMarked = function() {
        if (!currentSelection || !currentSelection.changed) return;
        var win = currentSelection.window, doc = win.document;

        function makeRange(point) {
          var range = doc.body.createTextRange(),
              node = point.node;
          if (!node) {
            range.moveToElementText(currentSelection.window.document.body);
            range.collapse(false);
          }
          else if (node.nodeType == 3) {
            range.moveToElementText(node.parentNode);
            var offset = point.offset;
            while (node.previousSibling) {
              node = node.previousSibling;
              offset += (node.innerText || "").length;
            }
            range.move("character", offset);
          }
          else {
            range.moveToElementText(node);
            range.collapse(true);
          }
          return range;
        }
```

```
215
216         var start = makeRange(currentSelection.start), end = makeRange(
                currentSelection.end);
217         start.setEndPoint("StartToEnd", end);
218         start.select();
219       };
220
221       // Get the top-level node that one end of the cursor is inside or
222       // after. Note that this returns false for 'no cursor', and null
223       // for 'start of document'.
224       select.selectionTopNode = function(container, start) {
225         var selection = container.ownerDocument.selection;
226         if (!selection) return false;
227
228         var range = selection.createRange(), range2 = range.duplicate();
229         range.collapse(start);
230         var around = range.parentElement();
231         if (around && isAncestor(container, around)) {
232           // Only use this node if the selection is not at its start.
233           range2.moveToElementText(around);
234           if (range.compareEndPoints("StartToStart", range2) == 1)
235             return topLevelNodeAt(around, container);
236         }
237
238         // Move the start of a range to the start of a node,
239         // compensating for the fact that you can't call
240         // moveToElementText with text nodes.
241         function moveToNodeStart(range, node) {
242           if (node.nodeType == 3) {
243             var count = 0, cur = node.previousSibling;
244             while (cur && cur.nodeType == 3) {
245               count += cur.nodeValue.length;
246               cur = cur.previousSibling;
247             }
248             if (cur) {
249               try{range.moveToElementText(cur);}
250               catch(e){return false;}
251               range.collapse(false);
252             }
253             else range.moveToElementText(node.parentNode);
254             if (count) range.move("character", count);
255           }
256           else {
257             try{range.moveToElementText(node);}
258             catch(e){return false;}
259           }
260           return true;
261         }
262
263         // Do a binary search through the container object, comparing
264         // the start of each node to the selection
265         var start = 0, end = container.childNodes.length - 1;
266         while (start < end) {
```

```
267           var middle = Math.ceil((end + start) / 2), node = container.childNodes[
                  middle];
268           if (!node) return false; // Don't ask. IE6 manages this sometimes.
269           if (!moveToNodeStart(range2, node)) return false;
270           if (range.compareEndPoints("StartToStart", range2) == 1)
271             start = middle;
272           else
273             end = middle - 1;
274         }
275         return container.childNodes[start] || null;
276       };
277
278       // Place the cursor after this.start. This is only useful when
279       // manually moving the cursor instead of restoring it to its old
280       // position.
281       select.focusAfterNode = function(node, container) {
282         var range = container.ownerDocument.body.createTextRange();
283         range.moveToElementText(node || container);
284         range.collapse(!node);
285         range.select();
286       };
287
288       select.somethingSelected = function(win) {
289         var sel = win.document.selection;
290         return sel && (sel.createRange().text != "");
291       };
292
293       function insertAtCursor(window, html) {
294         var selection = window.document.selection;
295         if (selection) {
296           var range = selection.createRange();
297           range.pasteHTML(html);
298           range.collapse(false);
299           range.select();
300         }
301       }
302
303       // Used to normalize the effect of the enter key, since browsers
304       // do widely different things when pressing enter in designMode.
305       select.insertNewlineAtCursor = function(window) {
306         insertAtCursor(window, "<br>");
307       };
308
309       select.insertTabAtCursor = function(window) {
310         insertAtCursor(window, fourSpaces);
311       };
312
313       // Get the BR node at the start of the line on which the cursor
314       // currently is, and the offset into the line. Returns null as
315       // node if cursor is on first line.
316       select.cursorPos = function(container, start) {
317         var selection = container.ownerDocument.selection;
318         if (!selection) return null;
319
```

```
320         var topNode = select.selectionTopNode(container, start);
321         while (topNode && !isBR(topNode)) {
322           topNode = topNode.previousSibling;
323         }
324
325         var range = selection.createRange(), range2 = range.duplicate();
326         range.collapse(start);
327         if (topNode) {
328           range2.moveToElementText(topNode);
329           range2.collapse(false);
330         }
331         else {
332           // When nothing is selected, we can get all kinds of funky errors here.
333           try { range2.moveToElementText(container); }
334           catch (e) { return null; }
335           range2.collapse(true);
336         }
337         range.setEndPoint("StartToStart", range2);
338
339         return {node: topNode, offset: range.text.length};
340       };
341
342       select.setCursorPos = function(container, from, to) {
343         function rangeAt(pos) {
344           var range = container.ownerDocument.body.createTextRange();
345           if (!pos.node) {
346             range.moveToElementText(container);
347             range.collapse(true);
348           }
349           else {
350             range.moveToElementText(pos.node);
351             range.collapse(false);
352           }
353           range.move("character", pos.offset);
354           return range;
355         }
356
357         var range = rangeAt(from);
358         if (to && to != from)
359           range.setEndPoint("EndToEnd", rangeAt(to));
360         range.select();
361       }
362
363       // Some hacks for storing and re-storing the selection when the editor loses
             and regains focus.
364       select.getBookmark = function (container) {
365         var from = select.cursorPos(container, true), to = select.cursorPos(container
             , false);
366         if (from && to) return {from: from, to: to};
367       };
368
369       // Restore a stored selection.
370       select.setBookmark = function(container, mark) {
371         if (!mark) return;
```

```
372         select.setCursorPos(container, mark.from, mark.to);
373       };
374     }
375     // W3C model
376     else {
377       // Store start and end nodes, and offsets within these, and refer
378       // back to the selection object from those nodes, so that this
379       // object can be updated when the nodes are replaced before the
380       // selection is restored.
381       select.markSelection = function (win) {
382         var selection = win.getSelection();
383         if (!selection || selection.rangeCount == 0)
384           return (currentSelection = null);
385         var range = selection.getRangeAt(0);
386 
387         currentSelection = {
388           start: {node: range.startContainer, offset: range.startOffset},
389           end: {node: range.endContainer, offset: range.endOffset},
390           window: win,
391           changed: false
392         };
393 
394         // We want the nodes right at the cursor, not one of their
395         // ancestors with a suitable offset. This goes down the DOM tree
396         // until a 'leaf' is reached (or is it *up* the DOM tree?).
397         function normalize(point){
398           while (point.node.nodeType != 3 && !isBR(point.node)) {
399             var newNode = point.node.childNodes[point.offset] || point.node.
                  nextSibling;
400             point.offset = 0;
401             while (!newNode && point.node.parentNode) {
402               point.node = point.node.parentNode;
403               newNode = point.node.nextSibling;
404             }
405             point.node = newNode;
406             if (!newNode)
407               break;
408           }
409         }
410 
411         normalize(currentSelection.start);
412         normalize(currentSelection.end);
413       };
414 
415       select.selectMarked = function () {
416         var cs = currentSelection;
417         // on webkit-based browsers, it is apparently possible that the
418         // selection gets reset even when a node that is not one of the
419         // endpoints get messed with. the most common situation where
420         // this occurs is when a selection is deleted or overwritten. we
421         // check for that here.
422         function focusIssue() {
423           return cs.start.node == cs.end.node && cs.start.offset == 0 && cs.end.
                offset == 0;
```

```
424         }
425         if (!cs || !(cs.changed || (webkit && focusIssue()))) return;
426         var win = cs.window, range = win.document.createRange();
427
428         function setPoint(point, which) {
429           if (point.node) {
430             // Some magic to generalize the setting of the start and end
431             // of a range.
432             if (point.offset == 0)
433               range["set" + which + "Before"](point.node);
434             else
435               range["set" + which](point.node, point.offset);
436           }
437           else {
438             range.setStartAfter(win.document.body.lastChild || win.document.body);
439           }
440         }
441
442         setPoint(cs.end, "End");
443         setPoint(cs.start, "Start");
444         selectRange(range, win);
445       };
446
447       // Helper for selecting a range object.
448       function selectRange(range, window) {
449         var selection = window.getSelection();
450         selection.removeAllRanges();
451         selection.addRange(range);
452       }
453       function selectionRange(window) {
454         var selection = window.getSelection();
455         if (!selection || selection.rangeCount == 0)
456           return false;
457         else
458           return selection.getRangeAt(0);
459       }
460
461       // Finding the top-level node at the cursor in the W3C is, as you
462       // can see, quite an involved process.
463       select.selectionTopNode = function(container, start) {
464         var range = selectionRange(container.ownerDocument.defaultView);
465         if (!range) return false;
466
467         var node = start ? range.startContainer : range.endContainer;
468         var offset = start ? range.startOffset : range.endOffset;
469         // Work around (yet another) bug in Opera's selection model.
470         if (window.opera && !start && range.endContainer == container && range.
              endOffset == range.startOffset + 1 &&
471             container.childNodes[range.startOffset] && isBR(container.childNodes[
                  range.startOffset]))
472           offset--;
473
474         // For text nodes, we look at the node itself if the cursor is
475         // inside, or at the node before it if the cursor is at the
```

```
476         // start.
477         if (node.nodeType == 3){
478           if (offset > 0)
479             return topLevelNodeAt(node, container);
480           else
481             return topLevelNodeBefore(node, container);
482         }
483         // Occasionally, browsers will return the HTML node as
484         // selection. If the offset is 0, we take the start of the frame
485         // ('after null'), otherwise, we take the last node.
486         else if (node.nodeName.toUpperCase() == "HTML") {
487           return (offset == 1 ? null : container.lastChild);
488         }
489         // If the given node is our 'container', we just look up the
490         // correct node by using the offset.
491         else if (node == container) {
492           return (offset == 0) ? null : node.childNodes[offset - 1];
493         }
494         // In any other case, we have a regular node. If the cursor is
495         // at the end of the node, we use the node itself, if it is at
496         // the start, we use the node before it, and in any other
497         // case, we look up the child before the cursor and use that.
498         else {
499           if (offset == node.childNodes.length)
500             return topLevelNodeAt(node, container);
501           else if (offset == 0)
502             return topLevelNodeBefore(node, container);
503           else
504             return topLevelNodeAt(node.childNodes[offset - 1], container);
505         }
506       };
507
508       select.focusAfterNode = function(node, container) {
509         var win = container.ownerDocument.defaultView,
510             range = win.document.createRange();
511         range.setStartBefore(container.firstChild || container);
512         // In Opera, setting the end of a range at the end of a line
513         // (before a BR) will cause the cursor to appear on the next
514         // line, so we set the end inside of the start node when
515         // possible.
516         if (node && !node.firstChild)
517           range.setEndAfter(node);
518         else if (node)
519           range.setEnd(node, node.childNodes.length);
520         else
521           range.setEndBefore(container.firstChild || container);
522         range.collapse(false);
523         selectRange(range, win);
524       };
525
526       select.somethingSelected = function(win) {
527         var range = selectionRange(win);
528         return range && !range.collapsed;
529       };
```

```
530
531       function insertNodeAtCursor(window, node) {
532         var range = selectionRange(window);
533         if (!range) return;
534
535         range.deleteContents();
536         range.insertNode(node);
537         webkitLastLineHack(window.document.body);
538         range = window.document.createRange();
539         range.selectNode(node);
540         range.collapse(false);
541         selectRange(range, window);
542       }
543
544       select.insertNewlineAtCursor = function(window) {
545         insertNodeAtCursor(window, window.document.createElement("BR"));
546       };
547
548       select.insertTabAtCursor = function(window) {
549         insertNodeAtCursor(window, window.document.createTextNode(fourSpaces));
550       };
551
552       select.cursorPos = function(container, start) {
553         var range = selectionRange(window);
554         if (!range) return;
555
556         var topNode = select.selectionTopNode(container, start);
557         while (topNode && !isBR(topNode)) {
558           topNode = topNode.previousSibling;
559         }
560
561         range = range.cloneRange();
562         range.collapse(start);
563         if (topNode)
564           range.setStartAfter(topNode);
565         else
566           range.setStartBefore(container);
567         return {node: topNode, offset: range.toString().length};
568       };
569
570       select.setCursorPos = function(container, from, to) {
571         var win = container.ownerDocument.defaultView,
572             range = win.document.createRange();
573
574         function setPoint(node, offset, side) {
575           if (offset == 0 && node && !node.nextSibling) {
576             range["set" + side + "After"](node);
577             return true;
578           }
579
580           if (!node)
581             node = container.firstChild;
582           else
583             node = node.nextSibling;
```

```
584
585           if (!node) return;
586
587           if (offset == 0) {
588             range["set" + side + "Before"](node);
589             return true;
590           }
591
592           var backlog = []
593           function decompose(node) {
594             if (node.nodeType == 3)
595               backlog.push(node);
596             else
597               forEach(node.childNodes, decompose);
598           }
599           while (true) {
600             while (node && !backlog.length) {
601               decompose(node);
602               node = node.nextSibling;
603             }
604             var cur = backlog.shift();
605             if (!cur) return false;
606
607             var length = cur.nodeValue.length;
608             if (length >= offset) {
609               range["set" + side](cur, offset);
610               return true;
611             }
612             offset -= length;
613           }
614         }
615
616         to = to || from;
617         if (setPoint(to.node, to.offset, "End") && setPoint(from.node, from.offset, "
              Start"))
618           selectRange(range, win);
619       };
620     }
621   })();
```

4.3.7. stringstream.js

```
1  /* String streams are the things fed to parsers (which can feed them
2   * to a tokenizer if they want). They provide peek and next methods
3   * for looking at the current character (next 'consumes' this
4   * character, peek does not), and a get method for retrieving all the
5   * text that was consumed since the last time get was called.
6   *
7   * An easy mistake to make is to let a StopIteration exception finish
8   * the token stream while there are still characters pending in the
9   * string stream (hitting the end of the buffer while parsing a
```

```
 * token). To make it easier to detect such errors, the stringstreams
 * throw an exception when this happens.
 */

// Make a stringstream stream out of an iterator that returns strings.
// This is applied to the result of traverseDOM (see codemirror.js),
// and the resulting stream is fed to the parser.
var stringStream = function(source){
  // String that's currently being iterated over.
  var current = "";
  // Position in that string.
  var pos = 0;
  // Accumulator for strings that have been iterated over but not
  // get()-ed yet.
  var accum = "";
  // Make sure there are more characters ready, or throw
  // StopIteration.
  function ensureChars() {
    while (pos == current.length) {
      accum += current;
      current = ""; // In case source.next() throws
      pos = 0;
      try {current = source.next();}
      catch (e) {
        if (e != StopIteration) throw e;
        else return false;
      }
    }
    return true;
  }

  return {
    // Return the next character in the stream.
    peek: function() {
      if (!ensureChars()) return null;
      return current.charAt(pos);
    },
    // Get the next character, throw StopIteration if at end, check
    // for unused content.
    next: function() {
      if (!ensureChars()) {
        if (accum.length > 0)
          throw "End of stringstream reached without emptying buffer ('" + accum +
              "').";
        else
          throw StopIteration;
      }
      return current.charAt(pos++);
    },
    // Return the characters iterated over since the last call to
    // .get().
    get: function() {
      var temp = accum;
      accum = "";
```

```
63        if (pos > 0){
64          temp += current.slice(0, pos);
65          current = current.slice(pos);
66          pos = 0;
67        }
68        return temp;
69      },
70      // Push a string back into the stream.
71      push: function(str) {
72        current = current.slice(0, pos) + str + current.slice(pos);
73      },
74      lookAhead: function(str, consume, skipSpaces, caseInsensitive) {
75        function cased(str) {return caseInsensitive ? str.toLowerCase() : str;}
76        str = cased(str);
77        var found = false;
78
79        var _accum = accum, _pos = pos;
80        if (skipSpaces) this.nextWhileMatches(/[\s\u00a0]/);
81
82        while (true) {
83          var end = pos + str.length, left = current.length - pos;
84          if (end <= current.length) {
85            found = str == cased(current.slice(pos, end));
86            pos = end;
87            break;
88          }
89          else if (str.slice(0, left) == cased(current.slice(pos))) {
90            accum += current; current = "";
91            try {current = source.next();}
92            catch (e) {break;}
93            pos = 0;
94            str = str.slice(left);
95          }
96          else {
97            break;
98          }
99        }
100
101       if (!(found && consume)) {
102         current = accum.slice(_accum.length) + current;
103         pos = _pos;
104         accum = _accum;
105       }
106
107       return found;
108     },
109
110     // Utils built on top of the above
111     more: function() {
112       return this.peek() !== null;
113     },
114     applies: function(test) {
115       var next = this.peek();
116       return (next !== null && test(next));
```

```
    },
    nextWhile: function(test) {
      var next;
      while ((next = this.peek()) !== null && test(next))
        this.next();
    },
    matches: function(re) {
      var next = this.peek();
      return (next !== null && re.test(next));
    },
    nextWhileMatches: function(re) {
      var next;
      while ((next = this.peek()) !== null && re.test(next))
        this.next();
    },
    equals: function(ch) {
      return ch === this.peek();
    },
    endOfLine: function() {
      var next = this.peek();
      return next == null || next == "\n";
    }
  };
};
```

4.3.8. tokenize.js

```
// A framework for simple tokenizers. Takes care of newlines and
// white-space, and of getting the text from the source stream into
// the token object. A state is a function of two arguments -- a
// string stream and a setState function. The second can be used to
// change the tokenizer's state, and can be ignored for stateless
// tokenizers. This function should advance the stream over a token
// and return a string or object containing information about the next
// token, or null to pass and have the (new) state be called to finish
// the token. When a string is given, it is wrapped in a {style, type}
// object. In the resulting object, the characters consumed are stored
// under the content property. Any whitespace following them is also
// automatically consumed, and added to the value property. (Thus,
// content is the actual meaningful part of the token, while value
// contains all the text it spans.)

function tokenizer(source, state) {
  // Newlines are always a separate token.
  function isWhiteSpace(ch) {
    // The messy regexp is because IE's regexp matcher is of the
    // opinion that non-breaking spaces are no whitespace.
    return ch != "\n" && /^[\s\u00a0]*$/.test(ch);
  }

  var tokenizer = {
```

```
    state: state,

    take: function(type) {
      if (typeof(type) == "string")
        type = {style: type, type: type};

      type.content = (type.content || "") + source.get();
      if (!/\n$/.test(type.content))
        source.nextWhile(isWhiteSpace);
      type.value = type.content + source.get();
      return type;
    },

    next: function () {
      if (!source.more()) throw StopIteration;

      var type;
      if (source.equals("\n")) {
        source.next();
        return this.take("whitespace");
      }

      if (source.applies(isWhiteSpace))
        type = "whitespace";
      else
        while (!type)
          type = this.state(source, function(s) {tokenizer.state = s;});

      return this.take(type);
    }
  };
  return tokenizer;
}
```

4.3.9. undo.js

```
/**
 * Storage and control for undo information within a CodeMirror
 * editor. 'Why on earth is such a complicated mess required for
 * that?', I hear you ask. The goal, in implementing this, was to make
 * the complexity of storing and reverting undo information depend
 * only on the size of the edited or restored content, not on the size
 * of the whole document. This makes it necessary to use a kind of
 * 'diff' system, which, when applied to a DOM tree, causes some
 * complexity and hackery.
 *
 * In short, the editor 'touches' BR elements as it parses them, and
 * the History stores these. When nothing is touched in commitDelay
 * milliseconds, the changes are committed: It goes over all touched
 * nodes, throws out the ones that did not change since last commit or
 * are no longer in the document, and assembles the rest into zero or
```

```
16    * more 'chains' -- arrays of adjacent lines. Links back to these
17    * chains are added to the BR nodes, while the chain that previously
18    * spanned these nodes is added to the undo history. Undoing a change
19    * means taking such a chain off the undo history, restoring its
20    * content (text is saved per line) and linking it back into the
21    * document.
22    */
23
24  // A history object needs to know about the DOM container holding the
25  // document, the maximum amount of undo levels it should store, the
26  // delay (of no input) after which it commits a set of changes, and,
27  // unfortunately, the 'parent' window -- a window that is not in
28  // designMode, and on which setTimeout works in every browser.
29  function History(container, maxDepth, commitDelay, editor) {
30    this.container = container;
31    this.maxDepth = maxDepth; this.commitDelay = commitDelay;
32    this.editor = editor; this.parent = editor.parent;
33    // This line object represents the initial, empty editor.
34    var initial = {text: "", from: null, to: null};
35    // As the borders between lines are represented by BR elements, the
36    // start of the first line and the end of the last one are
37    // represented by null. Since you can not store any properties
38    // (links to line objects) in null, these properties are used in
39    // those cases.
40    this.first = initial; this.last = initial;
41    // Similarly, a 'historyTouched' property is added to the BR in
42    // front of lines that have already been touched, and 'firstTouched'
43    // is used for the first line.
44    this.firstTouched = false;
45    // History is the set of committed changes, touched is the set of
46    // nodes touched since the last commit.
47    this.history = []; this.redoHistory = []; this.touched = [];
48  }
49
50  History.prototype = {
51    // Schedule a commit (if no other touches come in for commitDelay
52    // milliseconds).
53    scheduleCommit: function() {
54      var self = this;
55      this.parent.clearTimeout(this.commitTimeout);
56      this.commitTimeout = this.parent.setTimeout(function(){self.tryCommit();}, this
          .commitDelay);
57    },
58
59    // Mark a node as touched. Null is a valid argument.
60    touch: function(node) {
61      this.setTouched(node);
62      this.scheduleCommit();
63    },
64
65    // Undo the last change.
66    undo: function() {
67      // Make sure pending changes have been committed.
68      this.commit();
```

```
69
70      if (this.history.length) {
71        // Take the top diff from the history, apply it, and store its
72        // shadow in the redo history.
73        var item = this.history.pop();
74        this.redoHistory.push(this.updateTo(item, "applyChain"));
75        this.notifyEnvironment();
76        return this.chainNode(item);
77      }
78    },
79
80    // Redo the last undone change.
81    redo: function() {
82      this.commit();
83      if (this.redoHistory.length) {
84        // The inverse of undo, basically.
85        var item = this.redoHistory.pop();
86        this.addUndoLevel(this.updateTo(item, "applyChain"));
87        this.notifyEnvironment();
88        return this.chainNode(item);
89      }
90    },
91
92    clear: function() {
93      this.history = [];
94      this.redoHistory = [];
95    },
96
97    // Ask for the size of the un/redo histories.
98    historySize: function() {
99      return {undo: this.history.length, redo: this.redoHistory.length};
100   },
101
102   // Push a changeset into the document.
103   push: function(from, to, lines) {
104     var chain = [];
105     for (var i = 0; i < lines.length; i++) {
106       var end = (i == lines.length - 1) ? to : this.container.ownerDocument.
              createElement("BR");
107       chain.push({from: from, to: end, text: cleanText(lines[i])});
108       from = end;
109     }
110     this.pushChains([chain], from == null && to == null);
111     this.notifyEnvironment();
112   },
113
114   pushChains: function(chains, doNotHighlight) {
115     this.commit(doNotHighlight);
116     this.addUndoLevel(this.updateTo(chains, "applyChain"));
117     this.redoHistory = [];
118   },
119
120   // Retrieve a DOM node from a chain (for scrolling to it after undo/redo).
121   chainNode: function(chains) {
```

```
122       for (var i = 0; i < chains.length; i++) {
123         var start = chains[i][0], node = start && (start.from || start.to);
124         if (node) return node;
125       }
126     },
127
128     // Clear the undo history, make the current document the start
129     // position.
130     reset: function() {
131       this.history = []; this.redoHistory = [];
132     },
133
134     textAfter: function(br) {
135       return this.after(br).text;
136     },
137
138     nodeAfter: function(br) {
139       return this.after(br).to;
140     },
141
142     nodeBefore: function(br) {
143       return this.before(br).from;
144     },
145
146     // Commit unless there are pending dirty nodes.
147     tryCommit: function() {
148       if (!window.History) return; // Stop when frame has been unloaded
149       if (this.editor.highlightDirty()) this.commit(true);
150       else this.scheduleCommit();
151     },
152
153     // Check whether the touched nodes hold any changes, if so, commit
154     // them.
155     commit: function(doNotHighlight) {
156       this.parent.clearTimeout(this.commitTimeout);
157       // Make sure there are no pending dirty nodes.
158       if (!doNotHighlight) this.editor.highlightDirty(true);
159       // Build set of chains.
160       var chains = this.touchedChains(), self = this;
161
162       if (chains.length) {
163         this.addUndoLevel(this.updateTo(chains, "linkChain"));
164         this.redoHistory = [];
165         this.notifyEnvironment();
166       }
167     },
168
169     // [ end of public interface ]
170
171     // Update the document with a given set of chains, return its
172     // shadow. updateFunc should be "applyChain" or "linkChain". In the
173     // second case, the chains are taken to correspond the the current
174     // document, and only the state of the line data is updated. In the
175     // first case, the content of the chains is also pushed iinto the
```

```
176     // document.
177     updateTo: function(chains, updateFunc) {
178       var shadows = [], dirty = [];
179       for (var i = 0; i < chains.length; i++) {
180         shadows.push(this.shadowChain(chains[i]));
181         dirty.push(this[updateFunc](chains[i]));
182       }
183       if (updateFunc == "applyChain")
184         this.notifyDirty(dirty);
185       return shadows;
186     },
187
188     // Notify the editor that some nodes have changed.
189     notifyDirty: function(nodes) {
190       forEach(nodes, method(this.editor, "addDirtyNode"))
191       this.editor.scheduleHighlight();
192     },
193
194     notifyEnvironment: function() {
195       // Used by the line-wrapping line-numbering code.
196       if (window.frameElement && window.frameElement.CodeMirror.updateNumbers)
197         window.frameElement.CodeMirror.updateNumbers();
198       if (this.onChange) this.onChange();
199     },
200
201     // Link a chain into the DOM nodes (or the first/last links for null
202     // nodes).
203     linkChain: function(chain) {
204       for (var i = 0; i < chain.length; i++) {
205         var line = chain[i];
206         if (line.from) line.from.historyAfter = line;
207         else this.first = line;
208         if (line.to) line.to.historyBefore = line;
209         else this.last = line;
210       }
211     },
212
213     // Get the line object after/before a given node.
214     after: function(node) {
215       return node ? node.historyAfter : this.first;
216     },
217     before: function(node) {
218       return node ? node.historyBefore : this.last;
219     },
220
221     // Mark a node as touched if it has not already been marked.
222     setTouched: function(node) {
223       if (node) {
224         if (!node.historyTouched) {
225           this.touched.push(node);
226           node.historyTouched = true;
227         }
228       }
229       else {
```

```
230         this.firstTouched = true;
231       }
232     },
233
234     // Store a new set of undo info, throw away info if there is more of
235     // it than allowed.
236     addUndoLevel: function(diffs) {
237       this.history.push(diffs);
238       if (this.history.length > this.maxDepth)
239         this.history.shift();
240     },
241
242     // Build chains from a set of touched nodes.
243     touchedChains: function() {
244       var self = this;
245
246       // The temp system is a crummy hack to speed up determining
247       // whether a (currently touched) node has a line object associated
248       // with it. nullTemp is used to store the object for the first
249       // line, other nodes get it stored in their historyTemp property.
250       var nullTemp = null;
251       function temp(node) {return node ? node.historyTemp : nullTemp;}
252       function setTemp(node, line) {
253         if (node) node.historyTemp = line;
254         else nullTemp = line;
255       }
256
257       function buildLine(node) {
258         var text = [];
259         for (var cur = node ? node.nextSibling : self.container.firstChild;
260              cur && !isBR(cur); cur = cur.nextSibling)
261           if (cur.currentText) text.push(cur.currentText);
262         return {from: node, to: cur, text: cleanText(text.join(""))};
263       }
264
265       // Filter out unchanged lines and nodes that are no longer in the
266       // document. Build up line objects for remaining nodes.
267       var lines = [];
268       if (self.firstTouched) self.touched.push(null);
269       forEach(self.touched, function(node) {
270         if (node && node.parentNode != self.container) return;
271
272         if (node) node.historyTouched = false;
273         else self.firstTouched = false;
274
275         var line = buildLine(node), shadow = self.after(node);
276         if (!shadow || shadow.text != line.text || shadow.to != line.to) {
277           lines.push(line);
278           setTemp(node, line);
279         }
280       });
281
282       // Get the BR element after/before the given node.
283       function nextBR(node, dir) {
```

```
284         var link = dir + "Sibling", search = node[link];
285         while (search && !isBR(search))
286           search = search[link];
287         return search;
288       }
289
290       // Assemble line objects into chains by scanning the DOM tree
291       // around them.
292       var chains = []; self.touched = [];
293       forEach(lines, function(line) {
294         // Note that this makes the loop skip line objects that have
295         // been pulled into chains by lines before them.
296         if (!temp(line.from)) return;
297
298         var chain = [], curNode = line.from, safe = true;
299         // Put any line objects (referred to by temp info) before this
300         // one on the front of the array.
301         while (true) {
302           var curLine = temp(curNode);
303           if (!curLine) {
304             if (safe) break;
305             else curLine = buildLine(curNode);
306           }
307           chain.unshift(curLine);
308           setTemp(curNode, null);
309           if (!curNode) break;
310           safe = self.after(curNode);
311           curNode = nextBR(curNode, "previous");
312         }
313         curNode = line.to; safe = self.before(line.from);
314         // Add lines after this one at end of array.
315         while (true) {
316           if (!curNode) break;
317           var curLine = temp(curNode);
318           if (!curLine) {
319             if (safe) break;
320             else curLine = buildLine(curNode);
321           }
322           chain.push(curLine);
323           setTemp(curNode, null);
324           safe = self.before(curNode);
325           curNode = nextBR(curNode, "next");
326         }
327         chains.push(chain);
328       });
329
330       return chains;
331     },
332
333     // Find the 'shadow' of a given chain by following the links in the
334     // DOM nodes at its start and end.
335     shadowChain: function(chain) {
336       var shadows = [], next = this.after(chain[0].from), end = chain[chain.length -
          1].to;
```

```
337       while (true) {
338         shadows.push(next);
339         var nextNode = next.to;
340         if (!nextNode || nextNode == end)
341           break;
342         else
343           next = nextNode.historyAfter || this.before(end);
344         // (The this.before(end) is a hack -- FF sometimes removes
345         // properties from BR nodes, in which case the best we can hope
346         // for is to not break.)
347       }
348       return shadows;
349     },
350
351     // Update the DOM tree to contain the lines specified in a given
352     // chain, link this chain into the DOM nodes.
353     applyChain: function(chain) {
354       // Some attempt is made to prevent the cursor from jumping
355       // randomly when an undo or redo happens. It still behaves a bit
356       // strange sometimes.
357       var cursor = select.cursorPos(this.container, false), self = this;
358
359       // Remove all nodes in the DOM tree between from and to (null for
360       // start/end of container).
361       function removeRange(from, to) {
362         var pos = from ? from.nextSibling : self.container.firstChild;
363         while (pos != to) {
364           var temp = pos.nextSibling;
365           removeElement(pos);
366           pos = temp;
367         }
368       }
369
370       var start = chain[0].from, end = chain[chain.length - 1].to;
371       // Clear the space where this change has to be made.
372       removeRange(start, end);
373
374       // Insert the content specified by the chain into the DOM tree.
375       for (var i = 0; i < chain.length; i++) {
376         var line = chain[i];
377         // The start and end of the space are already correct, but BR
378         // tags inside it have to be put back.
379         if (i > 0)
380           self.container.insertBefore(line.from, end);
381
382         // Add the text.
383         var node = makePartSpan(fixSpaces(line.text), this.container.ownerDocument);
384         self.container.insertBefore(node, end);
385         // See if the cursor was on this line. Put it back, adjusting
386         // for changed line length, if it was.
387         if (cursor && cursor.node == line.from) {
388           var cursordiff = 0;
389           var prev = this.after(line.from);
390           if (prev && i == chain.length - 1) {
```

```
391             // Only adjust if the cursor is after the unchanged part of
392             // the line.
393             for (var match = 0; match < cursor.offset &&
394                  line.text.charAt(match) == prev.text.charAt(match); match++);
395             if (cursor.offset > match)
396               cursordiff = line.text.length - prev.text.length;
397           }
398           select.setCursorPos(this.container, {node: line.from, offset: Math.max(0,
                  cursor.offset + cursordiff)});
399         }
400         // Cursor was in removed line, this is last new line.
401         else if (cursor && (i == chain.length - 1) && cursor.node && cursor.node.
                parentNode != this.container) {
402           select.setCursorPos(this.container, {node: line.from, offset: line.text.
                  length});
403         }
404       }
405
406       // Anchor the chain in the DOM tree.
407       this.linkChain(chain);
408       return start;
409     }
410   };
```

4.3.10. unlock.js

```
1   $(".unlock").click(function () {
2           var $pageid = substr(document.URL,strrpos(document.URL, "=")+1);
3           var $b = $("input[type=checkbox]");
4
5           $b.each(function (index) {
6                   if ( $(this).is(":checked") ) {
7
8                           $code = $(this).val();
9
10                          $.getJSON("editorlib.php", { codigo: $code, pageid: $pageid
                                , mode: "3" }, function(json) {
11                          location.reload();
12                  }
13                  );
14                  }
15          })
16  });
17
18  $("#btnunlock").click(function() {
19          var $pageid = substr(document.URL,strrpos(document.URL, "=")+1);
20          var opciones="toolbar=no, location=no, directories=no, status=no, menubar=
                no, scrollbars=no, resizable=yes, width=508, height=365, top=85, left
                =140";
21          window.open("unlock.php?pageid=" + $pageid,"",opciones);
22  });
```

```
23
24  $("#btnref").click(function() {
25          location.reload();
26  });
27
28  $(".cancelUnlock").click(function() {
29          window.close();
30  });
```

4.3.11. util.js

```
1   /* A few useful utility functions. */
2
3   // Capture a method on an object.
4   function method(obj, name) {
5     return function() {obj[name].apply(obj, arguments);};
6   }
7
8   // The value used to signal the end of a sequence in iterators.
9   var StopIteration = {toString: function() {return "StopIteration"}};
10
11  // Apply a function to each element in a sequence.
12  function forEach(iter, f) {
13    if (iter.next) {
14      try {while (true) f(iter.next());}
15      catch (e) {if (e != StopIteration) throw e;}
16    }
17    else {
18      for (var i = 0; i < iter.length; i++)
19        f(iter[i]);
20    }
21  }
22
23  // Map a function over a sequence, producing an array of results.
24  function map(iter, f) {
25    var accum = [];
26    forEach(iter, function(val) {accum.push(f(val));});
27    return accum;
28  }
29
30  // Create a predicate function that tests a string againsts a given
31  // regular expression. No longer used but might be used by 3rd party
32  // parsers.
33  function matcher(regexp){
34    return function(value){return regexp.test(value);};
35  }
36
37  // Test whether a DOM node has a certain CSS class. Much faster than
38  // the MochiKit equivalent, for some reason.
39  function hasClass(element, className){
40    var classes = element.className;
```

```
41    return classes && new RegExp("(^| )" + className + "($| )").test(classes);
42  }
43
44  // Insert a DOM node after another node.
45  function insertAfter(newNode, oldNode) {
46    var parent = oldNode.parentNode;
47    parent.insertBefore(newNode, oldNode.nextSibling);
48    return newNode;
49  }
50
51  function removeElement(node) {
52    if (node.parentNode)
53      node.parentNode.removeChild(node);
54  }
55
56  function clearElement(node) {
57    while (node.firstChild)
58      node.removeChild(node.firstChild);
59  }
60
61  // Check whether a node is contained in another one.
62  function isAncestor(node, child) {
63    while (child = child.parentNode) {
64      if (node == child)
65        return true;
66    }
67    return false;
68  }
69
70  // The non-breaking space character.
71  var nbsp = "\u00a0";
72  var matching = {"{": "}", "[": "]", "(": ")",
73                  "}": "{", "]": "[", ")": "("};
74
75  // Standardize a few unportable event properties.
76  function normalizeEvent(event) {
77    if (!event.stopPropagation) {
78      event.stopPropagation = function() {this.cancelBubble = true;};
79      event.preventDefault = function() {this.returnValue = false;};
80    }
81    if (!event.stop) {
82      event.stop = function() {
83        this.stopPropagation();
84        this.preventDefault();
85      };
86    }
87
88    if (event.type == "keypress") {
89      event.code = (event.charCode == null) ? event.keyCode : event.charCode;
90      event.character = String.fromCharCode(event.code);
91    }
92    return event;
93  }
94
```

```
95  // Portably register event handlers.
96  function addEventHandler(node, type, handler, removeFunc) {
97    function wrapHandler(event) {
98      handler(normalizeEvent(event || window.event));
99    }
100   if (typeof node.addEventListener == "function") {
101     node.addEventListener(type, wrapHandler, false);
102     if (removeFunc) return function() {node.removeEventListener(type, wrapHandler,
          false);};
103   }
104   else {
105     node.attachEvent("on" + type, wrapHandler);
106     if (removeFunc) return function() {node.detachEvent("on" + type, wrapHandler)
          ;};
107   }
108 }
109
110 function nodeText(node) {
111   return node.textContent || node.innerText || node.nodeValue || "";
112 }
113
114 function nodeTop(node) {
115   var top = 0;
116   while (node.offsetParent) {
117     top += node.offsetTop;
118     node = node.offsetParent;
119   }
120   return top;
121 }
122
123 function isBR(node) {
124   var nn = node.nodeName;
125   return nn == "BR" || nn == "br";
126 }
127 function isSpan(node) {
128   var nn = node.nodeName;
129   return nn == "SPAN" || nn == "span";
130 }
```

4.4. Hojas de Estilo

4.4.1. CColors.css

```
1  html {
2    cursor: text;
3  }
4
5  .editbox {
6    margin: .4em;
```

```
7     padding: 0;
8     font-family: monospace;
9     font-size: 10pt;
10    color: black;
11  }
12
13  pre.code, .editbox {
14    color: #666666;
15  }
16
17  .editbox p {
18    margin: 0;
19  }
20
21  span.c-comment {
22    color: #BB9977;
23  }
24
25  span.c-keyword {
26    font-weight: bold;
27    color: blue;
28  }
29
30  span.c-string {
31    color: #AA2222;
32  }
33
34  span.c-stdfunc {
35    font-weight: bold;
36    color: #077;
37  }
38  span.c-customfunc {
39    font-weight: bold;
40    color: #0AA;
41  }
42
43  span.c-identifier {
44    color: black;
45  }
46
47  span.c-number {
48    color: #3A3;
49  }
50
51  span.c-token {
52    color: #151;
53  }
54
55  span.c-error {
56    color: #FFF;
57    background-color: #F00;
58  }
59
60  span.c-define {
```

```
  color: green;
}

span.c-lock {
  color: #DCDCDC;
}
```

4.4.2. chat/style.css

```

/* CSS Document */
body {
        font:12px arial;
        color: #222;
        text-align:left;
        padding:35px; }

form, p, span {
        margin:0;
        padding:0; }

input { font:12px arial; }

a {
        color:#0000FF;
        text-decoration:none; }

        a:hover { text-decoration:underline; }

#wrapper {
        margin:0;
        padding-bottom:25px;
        background:#EBF4FB;
        width:504px;
        border:1px solid #ACD8F0; }

#chatbox {
        text-align:left;
        margin:0 auto;
        margin-bottom:25px;
        padding:10px;
        background:#fff;
        height:270px;
        width:430px;
        border:1px solid #ACD8F0;
        overflow:auto; }

#usermsg {
        text-align:left;
        margin:0 auto;
        margin-left: 25px;
```

```
43          width:395px;
44          border:1px solid #ACD8F0;}
45
46  #submit { width: 60px; margin:0 auto }
47
48  #menu { padding:12.5px 25px 12.5px 25px; }
```