```
<?php
/**
 * TreeAndMenu extension - Adds #tree and #menu parser functions for collapsible
treeview's and dropdown menus
 *
 * See http://www.mediawiki.org/wiki/Extension:TreeAndMenu for installation and
usage details
 * See http://www.organicdesign.co.nz/Extension_talk:TreeAndMenu.php for
development notes and disucssion
 *
 * @package MediaWiki
 * @subpackage Extensions
 * @author Aran Dunkley [http://www.organicdesign.co.nz/nad User:Nad]
 * @copyright ¸ 2007 Aran Dunkley
 * @licence GNU General Public Licence 2.0 or later
 */

if (!defined('MEDIAWIKI')) die('Not an entry point.');

define('TREEANDMENU_VERSION','1.0.8, 2008-12-08');

# Set any unset images to default titles
if (!isset($wgTreeViewImages) || !is_array($wgTreeViewImages)) $wgTreeViewImages
= array();

$wgTreeMagic                    = "tree"; # the parser-function name for trees
$wgMenuMagic                    = "menu"; # the parser-function name for dropdown
menus
$wgTreeViewShowLines            = true;  # whether to render the dotted lines
joining nodes
$wgExtensionFunctions[]         = 'wfSetupTreeAndMenu';
$wgHooks['LanguageGetMagic'][] = 'wfTreeAndMenuLanguageGetMagic';

$wgExtensionCredits['parserhook'][] = array(
      'name'        => 'TreeAndMenu',
      'author'      => '[http://www.organicdesign.co.nz/nad Nad],
[http://www.organicdesign.co.nz/User:Sven Sven]',
      'url'         => 'http://www.mediawiki.org/wiki/Extension:Treeview',
      'description' => 'Adds #tree and #menu parser functions which contain
bullet-lists to be rendered as collapsible treeview\'s or dropdown menus.
                        The treeview\'s use the
[http://www.destroydrop.com/javascripts/tree dTree] JavaScript tree menu,
                        and the dropdown menu\'s use
[http://www.htmldog.com/articles/suckerfish/dropdowns/ Son of Suckerfish]',
      'version'     => TREEANDMENU_VERSION
      );

class TreeAndMenu {

      var $version  = TREEANDMENU_VERSION;
      var $uniq     = '';      # uniq part of all tree id's
      var $uniqname = 'tam';   # input name for uniqid
      var $id       = '';      # id for specific tree
      var $baseDir  = 'c:/xampp/htdocs/wiki/extensions/TreeAndMenu';# internal
absolute path to treeview directory
      var $baseUrl  = '';      # external URL to treeview directory (relative to
domain)
      var $images   = 'c:/xampp/htdocs/wiki/extensions/TreeAndMenu/img';      #
internal JS to update dTree images
      var $useLines = true;    # internal variable determining whether to render
connector lines
      var $args     = array(); # args for each tree
```

```
	/**
	 * Constructor
	 */

	function __construct() {
		global $wgOut, $wgHooks, $wgParser, $wgScriptPath, $wgJsMimeType,
			$wgTreeMagic, $wgMenuMagic, $wgTreeViewImages,
$wgTreeViewShowLines;

		# Add hooks
		$wgParser->setFunctionHook($wgTreeMagic,array($this,'expandTree'));
		$wgParser->setFunctionHook($wgMenuMagic,array($this,'expandMenu'));
		$wgHooks['ParserAfterTidy'][] = array($this,'renderTreeAndMenu');

		# Update general tree paths and properties
		    $wgScriptPath='c:/xampp/htdocs/wiki';
		    $this->baseDir  = dirname(__FILE__);
		      $this->baseUrl  = str_replace('\\ ', '/', $this->baseDir);
		$this->baseUrl  = preg_replace('|^.+(?=[/\\\\]extensions)|',
$wgScriptPath, $this->baseDir);
		   #$this->baseDir  = dirname(__FILE__);
		   #$this->baseDir='c:/xampp/htdocs/wiki/extensions/TreeAndMenu/';
		    #$this->baseDir =  preg_replace('|\\\\|', "/", $this->baseDir);
		#$this->baseUrl  =
preg_replace('|^.+(?=/extensions)|',$wgScriptPath,$this->baseDir);

		 #    $this->baseUrl  =
preg_replace('|^.+(?=/extensions)|',$wgScriptPath,$this->baseDir);

		   # $this->baseUrl  = str_replace('\\', '/', $this->baseDir);
		    #$this->baseUrl='/wiki/extension/TreeAndMenu/';
		$this->useLines = $wgTreeViewShowLines ? 'true' : 'false';
		$this->uniq     = uniqid($this->uniqname);

		# Convert image titles to file paths and store as JS to update dTree
		foreach ($wgTreeViewImages as $k => $v) {
			$title = Title::newFromText($v,NS_IMAGE);
			$image = Image::newFromTitle($title);

$wgTreeViewImages="c:/xampp/htdocs/wiki/extensions/TreeAndMenu/img";
			$v = $image && $image->exists() ? $image->getURL() :
$wgTreeViewImages[$k];
			$this->images .= "tree.icon['$k'] = '$v';";
			}

		# Add link to output to load dtree.js script
		$wgOut->addScript("<script type=\"$wgJsMimeType\"
src=\"/extensions/TreeView5/dtree.js\"></script>\n");


	}


	/**
	 * Expand #tree parser-functions
	 */
	public function expandTree() {
		$args = func_get_args();
		return $this->expandTreeAndMenu('tree', $args);
	}

	/**
```

```
	 * Expand #menu parser-functions
	 */
	public function expandMenu() {
		$args = func_get_args();
		return $this->expandTreeAndMenu('menu', $args);
	}

	/**
	 * Expand either kind of parser-function (reformats tree rows for matching
later) and store args
	 */
	private function expandTreeAndMenu($magic, $args) {
		$parser = array_shift($args);

		# Store args for this tree for later use
		$text = '';
		foreach ($args as $arg) if (preg_match('/^(\\w+?)\\s*=\\s*(.+)$/s',
$arg, $m)) $args[$m[1]] = $m[2]; else $text = $arg;

		# If root, parse as wikitext
		if (isset($args['root'])) {
			$p = clone $parser;
			$o = clone $parser->mOptions;
			$o->mTidy = $o->mEnableLimitReport = false;
			$html = $p->parse($args['root'], $parser->mTitle, $o, false,
true)->getText();
			$args['root'] = addslashes($html);
		}

		# Create a unique id for this tree or use id supplied in args and
store args wrt id
		$this->id = isset($args['id']) ? $args['id'] : uniqid('');
		$args['type'] = $magic;
		$this->args[$this->id] = $args;

		# Reformat tree rows for matching in ParserAfterStrip
		$text = preg_replace('/(?<=\\*)\\s*\\[\\[Image:(.+?)\\]\\]/',
"{$this->uniq}3$1{$this->uniq}4", $text);
		$text = preg_replace_callback('/^(\\*+)(.*?)$/m', array($this,
'formatRow'), $text);

		return $text;
	}


	/**
	 * Reformat tree bullet structure recording row, depth and id in a format
which is not altered by wiki-parsing
	 * - format is: 1{uniq}-{id}-{depth}-{item}-2{uniq}
	 * - sequences of this format will be matched in ParserAfterTidy and
converted into dTree JavaScript
	 * - NOTE: we can't encode a unique row-id because if the same tree
instranscluded twice a cached version
	 *         may be used (even if parser-cache disabled) this also means
that tree id's may be repeated
	 */
	private function formatRow($m) {
		return "\x7f1{$this->uniq}\x7f{$this->id}\x7f".(strlen($m[1])-
1)."\x7f$m[2]\x7f2{$this->uniq}";
	}


	/**
```

```
       * Called after parser has finished (ParserAfterTidy) so all transcluded
parts can be assembled into final trees
       */
      public function renderTreeAndMenu(&$parser, &$text) {
            global $wgJsMimeType;
            $u = $this->uniq;

            # Determine which trees are sub trees
            # - there should be a more robust way to do this,
            #   it's just based on the fact that all sub-tree's have a minus
preceding their row data
            if (!preg_match_all("/\x7f\x7f1$u\x7f(.+?)\x7f/", $text, $subs))
$subs = array(1 => array());

            # Extract all the formatted tree rows in the page and if any,
replace with dTree JavaScript
            if (preg_match_all("/\x7f1$u\x7f(.+?)\x7f([0-
9]+)\x7f({$u}3(.+?){$u}4)?(.*?)(?=\x7f[12]$u)/", $text, $matches,
PREG_SET_ORDER)) {

                  # PASS-1: build $rows array containing depth, and tree
start/end information
                  $rows   = array();
                  $depths = array('' => 0); # depth of each tree root
                  $rootId = '';             # the id of the current root-tree
(used as tree id in PASS2)
                  $lastId = '';
                  $lastDepth = 0;
                  foreach ($matches as $match) {
                        list(, $id, $depth,, $icon, $item) = $match;
                        $start = false;
                        if ($id != $lastId) {
                              if (!isset($depths[$id])) $depths[$id] =
$depths[$lastId]+$lastDepth;
                              if ($start = $rootId != $id && !in_array($id,
$subs[1])) $depths[$rootId = $id] = 0;
                        }
                        if ($item) $rows[] = array($rootId, $depth+$depths[$id],
$icon, addslashes($item), $start);
                        $lastId    = $id;
                        $lastDepth = $depth;
                  }

                  # PASS-2: build the JavaScript and replace into $text
                  $parents   = array(); # parent node for each depth
                  $parity    = array(); # keep track of odd/even rows for each
depth
                  $node      = 0;
                  $last      = -1;
                  $nodes     = '';
                  $opennodes = array();
                  foreach ($rows as $i => $info) {
                        $node++;
                        list($id, $depth, $icon, $item, $start) = $info;
                        $objid = $this->uniqname . preg_replace('/\W/', '',
$id);
                        $args  = $this->args[$id];
                        $type  = $args['type'];
                        $end   = $i == count($rows)-1 || $rows[$i+1][4];
                        if (!isset($args['root'])) $args['root'] = ''; # tmp -
need to handle rootless trees
                        $openlevels = isset($args['openlevels']) ?
$args['openlevels']+1 : 0;
```

```
                        if ($start) $node = 1;

                        # Append node script for this row
                        if ($depth > $last) $parents[$depth] = $node-1;
                        $parent = $parents[$depth];
                        if ($type == 'tree') {
                              $nodes .= "$objid.add($node, $parent,
'$item');\n";
                              if ($depth > 0 && $openlevels > $depth)
$opennodes[$parent] = true;
                        }
                        else {
                              if (!$start) {
                                    if ($depth < $last) $nodes .=
str_repeat('</ul></li>', $last-$depth);
                                    elseif ($depth > $last) $nodes .= "\n<ul>";
                              }
                              $parity[$depth] = isset($parity[$depth]) ?
$parity[$depth]^1 : 0;
                              $class = $parity[$depth] ? 'odd' : 'even';
                                    $item=str_replace("\\","",$item);
                                    $nodes .= "<li class=\"$class\">$item";
                              if ($depth >= $rows[$node][1]) $nodes .=
"</li>\n";
                        }
                        $last = $depth;

                        # Last row of current root, surround nodes dtree or menu
script and div etc
                        if ($end) {
                              $class = isset($args['class']) ? $args['class'] :
"d$type";
                              if ($type == 'tree') {

                                    # Finalise a tree
                                    $adds[$id] = isset($args[$id]['root']) ?
"tree.add(0,-1,'".$this->args[$id]['root']."');" : '';
                                    $top = $bottom = $root = $opennodesjs = '';
                                    foreach (array_keys($opennodes) as $i)
$opennodesjs .= "$objid.o($i);";
                                    foreach ($args as $arg => $pos)
                                          if (($pos == 'top' || $pos == 'bottom'
|| $pos == 'root') && ($arg == 'open' || $arg == 'close'))
                                                $$pos .= "<a href=\"javascript:
$objid.{$arg}All();\"> {$arg} all</a> ";
                                    if ($top) $top = "<p> $top</p>";
                                    if ($bottom) $bottom =
"<p> $bottom</p>";
                                    $html = "$top<div class='$class' id='$id'>
                                                <script
type=\"$wgJsMimeType\">/*<![CDATA[*/
                                                      // TreeAndMenu{$this-
>version}
                                                      tree = new
dTree('$objid');
                                                      for (i in tree.icon)
tree.icon[i] = 'c:/xampp/htdocs/wiki/extensions/TreeAndMenu/'+tree.icon[i];
                                                      tree.config.useLines =
{$this->useLines};
                                                      $adds[$id]
                                                      $objid = tree;
                                                      $nodes
```

```
      document.getElementById('$id').innerHTML = $objid.toString();
                                                      $opennodesjs
                                                /*]]>*/</script>
                                          </div>$bottom";
                                    }
                                    else {

                                          # Finalise a menu
                                          if ($depth > 0) $nodes .=
str_repeat('</ul></li>', $depth);
                                          $nodes = preg_replace("/<(a.*?
)title=\".+?\".*?>/", "<$1>", $nodes); # IE has problems with title attribute in
suckerfish menus
                                          $html = "
                                                <ul class='$class'
id='$id'>\n$nodes</ul>
                                                <script
type=\"$wgJsMimeType\">/*<![CDATA[*/
                                                      if (window.attachEvent) {
                                                            var sfEls =
document.getElementById('$id').getElementsByTagName('li');
                                                            for (var i=0;
i<sfEls.length; i++) {

      sfEls[i].onmouseover=function() { this.className+=' sfhover'; }

      sfEls[i].onmouseout=function()  {
this.className=this.className.replace(new RegExp(' sfhover *'),''); }
                                                            }
                                                      }
                                                /*]]>*/</script>
                                                ";
                                    }

                                    $text  = preg_replace("/\x7f1$u\x7f$id\x7f.+?$/m",
$html, $text, 1); # replace first occurence of this trees root-id
                                    $nodes = '';
                                    $last  = -1;
                              }
                        }
            }

            $text = preg_replace("/\x7f1$u\x7f.+?[\\r\\n]+/m", '', $text); #
Remove all unreplaced row information
            return true;
      }

}


/**
 * Called from $wgExtensionFunctions array when initialising extensions
 */
function wfSetupTreeAndMenu() {
      global $wgTreeAndMenu;
      $wgTreeAndMenu = new TreeAndMenu();
}


/**
 * Reserve magic words
 */
```

```
function wfTreeAndMenuLanguageGetMagic(&$magicWords, $langCode = 0) {
      global $wgTreeMagic, $wgMenuMagic;
      $magicWords[$wgTreeMagic] = array($langCode, $wgTreeMagic);
      $magicWords[$wgMenuMagic] = array($langCode, $wgMenuMagic);
      return true;
}
```