The PHP Layers Menu System 3.0

Copyright information

Disclaimer

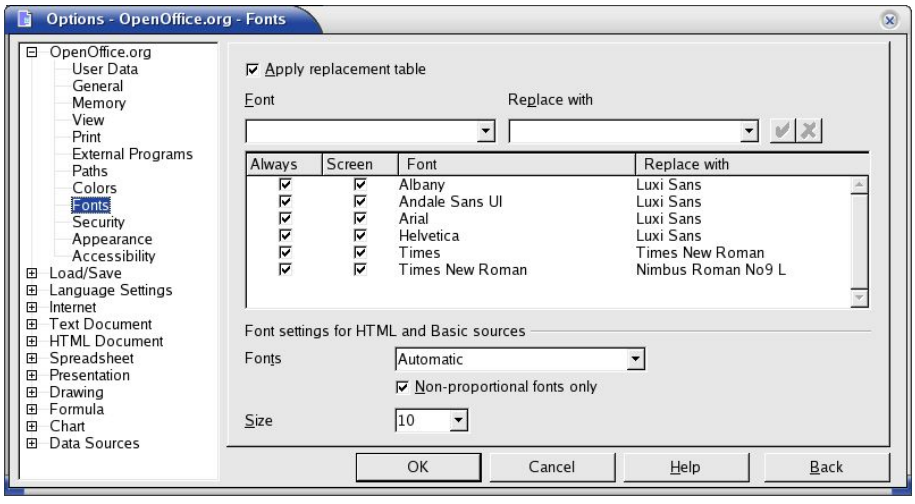
No liability for the contents of this document can be accepted.
Use the concepts, examples, and other content at your own risk.

All copyrights are held by their respective owners, unless
specifically noted otherwise. Use of a term in this document
should not be regarded as affecting the validity of any
trademark or service mark.

Naming of particular products or brands should not be seen as
endorsements, with the exception of the term "GNU/Linux". We
wholeheartedly endorse the use of GNU/Linux in every situation
where it is appropriate. It is an extremely versatile, stable,
and robust operating system that offers an ideal operating
environment for the PHP Layers Menu System.

Visualization of this document

This document uses “Arial” and “Times New Roman” fonts, that are bundled with the MS Windows environments. On GNU/Linux operating systems, the same fonts are available with the `msttcorefonts` package; their usage should be rather easy also on other graphical interfaces based on XFree86, as XFree86 supports the usage of True Type fonts. Alternatively, needed True Type fonts can also be installed only in OpenOffice.org/StarOffice, using the `spadmin` tool. Anyway, either if you do not want to install such fonts or if you want to see more beautiful fonts, you can use the fonts replacement feature; you find it on the main menu, under Tools > Options:



As an example, you can replace “Arial” with “Luxi Sans” and “Times New Roman” with “Nimbus Roman No9 L”; for further details, refer to the OpenOffice.org/StarOffice documentation/help.

Table of Contents

1 - Overview

PHPLM is a hierarchical dynamic menu system that allows to prepare “on the fly” dynamic HTML menus relying on the PHP scripting engine for the processing of data items.

It is released under the GNU Lesser General Public License (LGPL), either Version 2.1, or (at your option) any later version.

It supports a wide range of browsers: Mozilla, Konqueror, Netscape, Opera, Internet Explorer; rather old browser versions are supported, too; accessibility is provided for text-only browsers.

It achieves a compact view and a reasonably small file size for the page also with a very large number of entries.

It provides horizontal and vertical layers-based menus whose behavior is analogous to the Gnome, KDE, and MS Windows main menus.

It provides also JavaScript-based tree menus, whose look is analogous to the most widely used file managers and bookmark handling tools and whose nodes can be expanded and collapsed on sufficiently DOM-compliant browsers (they remain completely expanded for the other browsers).

An extended class is provided to prepare also “server-side based” tree menus (that have just the same look of the above JavaScript-based tree menus, but require the PHP support on the web server) and plain menus that do not require the JavaScript support to the browser.

An arbitrary number of vertical and horizontal menus can be used on the same page.

As much levels as needed can be used and each menu is dynamically generated using data retrieved from a file, a string, or a database table; the data format is rather simple and intuitive.

Multiple languages are supported (i18n) if data are retrieved from a database.

PHPLM is compliant with current recommendations for PHP developers: it works correctly with the following settings

- `register_globals = Off`
- `safe_mode = On`
- `error_reporting = E_ALL`
- `allow_call_time_pass_reference = Off`
- `short_open_tag = Off`

If PEAR is not used (i.e. if the DB support is not used), it works correctly also if the `open_basedir` restriction is in effect.

It is compliant with the following standards:

- XHTML 1.0 Transitional
- CSS 2.0

Some interesting customizations are bundled in the `PATCHES` directory.

2 - Requirements

PHP 4.1+ is needed; furthermore, to support multiple DBMSs, I have used the PEAR DB abstraction layer; hence, if you want to use the DB support without changing the code of PHPLM, you need a working PEAR installation.

If you do not need PEAR, you may want to comment out the following lines in `lib/layersmenu.inc.php`:

```
require_once "PEAR.php";
require_once "DB.php";
```

also to avoid an error message if the `open_basedir` restriction prevents you from including `PEAR.php` and `DB.php`.

Dumps for the PostgreSQL and MySQL DBMSs are bundled with the package in the `DUMPS` directory; PHPLM has been tested only with PostgreSQL and MySQL, but it uses the DB in a very simple way, hence it should work correctly with all DBMSs supported by PEAR DB.

PHPLM uses a “PEARified” version of the PHPLib template class; all methods have been renamed to match the PEAR “standards”; in the original “PEARified” version, the class name has been changed too, from “`Template`” to “`Template_PHPLIB`”. I have made some minor changes to “`Template_PHPLIB`” to remove its dependency from PEAR in order to allow to use PHPLM without PEAR if the DB support is not needed. For brevity, I have also restored the original name, i.e. “`Template`”. This means that, if you are already using the original “`Template`” class of PHPLib, you can hit a conflict between the two classes; in this case, you can overcome this conflict changing a few strings in the PHPLM code to restore the “`Template_PHPLIB`” class name. In the package, I have bundled the `CHANGE_TEMPLATE_CLASS_NAME.sh` shell script just to perform this change if needed.

3 - Workarounds for MS Windows' oddities

In versions before 3.0.0, if you use the MS Windows environment as “web server”, you can get an error like

```
Template Error: filename: file ./C:\Programs\Apache
Group\Apache\htdocs\phplayersmenu/templates/layersmenu-horizontal_menu.ihtml does not exist.
Halted.
```

This is **not** a bug of PHPLM, it is an error due to the buggy “file system” notation of the MS Windows environment, as it is explained in this footnote[1]. To workaround this MS Windows oddity, I have commented the first “`if`” statement in the `_filename()` method of the `Template` class (`lib/template.inc.php`):

```
--- template.inc.php.orig [...]
+++ template.inc.php [...]
@@ -465,9 +465,9 @@
      */
     function _filename($filename)
     {
-        if (substr($filename, 0, 1) != "/") {
-            $filename = $this->root."/".$filename;
-        }
+//        if (substr($filename, 0, 1) != "/") {
+//            $filename = $this->root."/".$filename;
+//        }

         if (!file_exists($filename)) {
             $this->halt("filename: file $filename does not exist.");
```

However, the above error occurred only using the `LayersMenu` class as in the following:

```
$mid = new LayersMenu();
...
$mid->setDirroot("C:\Programs\Apache Group\Apache\htdocs\phplayersmenu");
```

The problem was solvable also without changing the code of the `Template` class, simply replacing the odd notation above with the following correct notation:

```
$mid->setDirroot("/Programs/Apache Group/Apache/htdocs/phplayersmenu");
```

No further details are provided here, as the usage of `setDiroot()` and analogous methods of the `LayersMenu` class is considered further on in this document.

1 The mentioned error is triggered by the `_filename()` method of the `Template` class and is due to the stupid “file system” notation of the MS Windows environments, that use “C:”, “D:”, and so on, to indicate different disks or disk partitions and do not foresee a true root directory (i.e. “`/`”) being the father of all other directories. To realize that this is a really stupid notation, just observe that, as an example, “C:” is both a perfectly valid file name and a perfectly valid directory name, hence in general it is impossible to say if “C:” is either a relative path or an absolute path, or even the name of a file, while there is no doubt that, as an example, “`/home/pratesi/public_html/pgmarket/shopping`” is an absolute path and “`shopping`” is a relative path, as the first one begins with “`/`”, whereas the second does not. Hence, any change to the `_filename ()` method intended to consider also the MS Windows' disks notation would break the correctness of the code.

4 - Documentation of the classes

All the PHP code of the PHPLM package is documented through in-line comments written according to the phpDocumentor [PHPDOC] syntax. Hence, the documentation of the PHPLM classes can be easily and automatically generated running a command like the one provided in the `README.PHPDOC` bundled with the PHPLM package. For this reason, this manual does not provide detailed descriptions of the API of PHPLM. The named documentation is available and browsable on the demo site.

5 - Definition of data to build the menus

Currently, data needed to build the menus can be be retrieved by PHPLM in three different ways, i.e. from a text file, a PHP string, or a database.

5.1 - Text file format

Data related to the menus can be stored in a text file; each menu item corresponds to a line, according to the following format:

```
[dots]|[text]|[link]|[title]|[icon]|[target]|[expanded]
```

Items have to be listed in the same order resulting from a depth-first search over the tree, i.e. in the same order they appear on a completely exploded corresponding tree menu.

The "`|`" (pipe) character is used as a separator between the fields; a different separator can be chosen using related methods, whose documentation can be generated through phpDocumentor, as said above. Let us consider the above mentioned fields.

`[dots]` - some dots to decide the hierarchical level of the item: if the hierarchical level of a generic item is N, then this field must consist of N dots for that item. The hierarchical level of the first item must be equal to 1, hence you have to use only one dot for the first item. If the hierarchical level of a generic item is N, then the hierarchical level of its children is N+1, hence this field must consist of N dots for that item and of N+1 dots for its children. I explicitly point out that you can use also characters that are not dots: currently, PHPLM only counts the number of characters of this field, hence it behave just the same way if you use 3 whatever characters instead of 3 dots, e.g. "`foo`" instead of "`...`". This field is mandatory.

`[text]` - The text label that will be used for the item. This field is mandatory.

`[link]` - The URL associated to the item. This field can be empty, i.e. items that do not link anything are allowed.

`[title]` - The "`title`" attribute to be used for the "`anchor`" tag corresponding to `[link]`. This field is disregarded if the `[link]` field is empty. This field can be empty.

`[icon]` - The filename of an icon associated to the item. The directory path for icons has not to set through the suited method, it has not to be written in this field. According to the habit for favorite icons, such icons have to consist of 16×16 pixels. The content of this field may be either significant or not, depending on the menu type and of the link hierarchical characteristics (leaf item or not); this issue will be clarified later on. This field can be empty.

`[target]` - The "`target`" attribute to be used for the "`anchor`" tag corresponding to `[link]`. This field is disregarded if the `[link]` field is empty. This field can be empty.

`[expanded]` - A boolean field to specify if by default the corresponding branch has to be expanded or not. Currently, this field is significant only for the tree menus, and obviously, it is not significant for leaf item. This field can be empty.

I explicitly point out that, if the last N non mandatory fields are empty, you can completely omit them, i.e. you can omit also the separators... in other words, the separators are needed only when and where they have to separate something :-)

Note: in the menu structure text file, you can comment out an item as it is usually done for configuration files and scripts, i.e. simply inserting a "#" as the first character of the line.

W.r.t. the `[link]` field, if you need, using the `setPrependedUrl()` method, you can set a string to be prepended to the content of this field for each item. An example of usage of `setPrependedUrl()` is provided in the `example-db-hormenu.php` script and commented later.

5.2 - String format

The same data that you would put in the menu structure file can be passed to PHPLM using a PHP string, you only have to take care of using the escape character corresponding to the newline, i.e. "`\n`". An example is provided in the demo script (`index.php`): this is the content of `layersmenu-vertical-1.txt`

```
.|Linus B. Torvalds|http://www.cs.Helsinki.FI/u/torvalds/|The father of Linux||Linus
[...]
...|Bugzilla|http://qa.mandrakesoft.com/|||Linux
..|Slackware|http://www.freesoftware.org/|Slackware Linux||Linux
```

and this is the example code to use a corresponding string

```
$menustring =
".|Linus B. Torvalds|http://www.cs.Helsinki.FI/u/torvalds/|The father of Linux||Linus\n" .
[...]
"...|Bugzilla|http://qa.mandrakesoft.com/|||Linux\n" .
"..|Slackware|http://www.freesoftware.org/|Slackware Linux||Linux\n";
$mid->setMenuStructureString($menustring);
```

5.3 - Database support

Data related to the menus can also be stored in a database. In the DUMPS directory, you can find dumps ready for use with PostgreSQL and MySQL. Preparation of dumps for other DBMSs should be rather straightforward; only very simple SQL commands are used, hence you should not hit any compatibility issue using another DBMS. Let us look at the fields of the `phplayersmenu` table in `pgsql.start.dump`; they are just the same fields foreseen by the format used for the menu structure file/string, apart from some differences inherently due to the storage in a database table:

- `id` is the primary key needed to identify an item in the table; this field has to be greater than 1 for each item;
- `parent_id` identifies the "father" item; its value must be 1 for "first-level" items, i.e. for items with hierarchical level = 1, i.e. for children of the (non-existent) "Top" item;
- the `[dots]` field is not needed: the hierarchical relationship among items is determined by the `id` and `parent_id` fields;
- `orderfield` is needed to sort all children of the same item (an ASCendant order is used); in the menu structure file, this field is not needed, as the same order the items are put in the file is used.

Suited methods allow to change the names of the table and of its fields and to choose the language; please refer to the `example-db-*.php` scripts for examples of use of such methods (look also at the commented out code).

5.4 - Internationalization support

PHPLM supports internationalization (i18n) if data related to the menus are stored in a database[2]. In other words, you can obtain menus whose text contents may appear in more than one language. Let us consider `DUMPS/pgsql.start.dump`; the `phplayersmenu_i18n` table is devoted just to internationalization of text fields of the `phplayersmenu` table, i.e. "`text`" and "`title`". There are other two fields; "`id`" identifies the corresponding item in the "`phplayersmenu`" table; "`language`" identifies the internationalization language for the tuple at hand. Data stored in the "`text`" and "`title`" fields of the "`phplayersmenu`" correspond to your default language. If you use a non default language, such data are replaced by data stored in the "`phplayersmenu_i18n`" table in the language at hand, if available. Items whose "`text`" and "`title`" are not internationalized through the "`phplayersmenu_i18n`" table are always

2 Currently, i18n support is not available if data are stored in a text file.

shown in the default language, hence the internationalization of items is not mandatory for any item and for any language. In other words, the menu system translates the text, but only if you want and only for items for which you have provided the corresponding translation. Suited methods allow to change the names of the table and of its fields and to choose the language; please refer to the `example-db-*.php` scripts for examples of use of such methods (look also at the commented out code).

6 - Layers Menus

The horizontal and vertical layers menus behavior is analogous to the Gnome, KDE, and MS Windows main menus.

Layers menus require JavaScript and work at least on the following browsers:

- Mozilla 0.7+ (versions 0.9.1+ are suggested)
- Netscape 6.1+ and other browsers based on Mozilla, e.g. Epiphany and Galeon
- Netscape 4.07+
- Konqueror 2.1+ and browsers based on it, e.g. Safari
- Opera 5.x, 6.x, 7.x
- Internet Explorer 4, 5, 5.5, 6.

This menu system is usable but really uncomfortable for users of text-only browsers; either the tree menu system or the accessibility solutions should be given to text-only browsers.

In the following, some notes about Konqueror.

If `shutdownOnClick` is set to 1 in `layersmenu.js`, all the layers should disappear when clicking elsewhere in the browser window; this works on Konqueror 2.2+, but it does not work on Konqueror 2.1. The layers menu is not correctly usable with Konqueror 1.9.8 (KDE 2.0), because K. 1.9.8 triggers onmouseover events also for hidden layers; currently, I do not know if there is a way to avoid this behavior.

The menu system should work also on browsers based on Konqueror; according to reports of users, it works correctly with Apple Safari.

6.1 - The example-hormenu.php script

To get started with the Layers Menus, let us examine the code of the `example-hormenu.php` script. The look of the resulting menu can be controlled through the `.css` style sheet and the `.ihtml` templates.

Let us consider the `<head>...</head>` section. After the `<meta>` tag, I define some PHP variables:

```
<?php
/* TO USE RELATIVE PATHS: */
$myDirPath = "";
$myWwwPath = "";
/* TO USE ABSOLUTE PATHS: */
//$myDirPath = "/home/pratesi/public_html/phplayersmenu/";
//$myWwwPath = "/~pratesi/phplayersmenu/";
?>
```

The definitions corresponding to “absolute paths” are useful to get the script working also in another directory and are used in the subsequent lines of the script. For brevity, these variables are not used in the other examples bundled with the package; the change of the other `example-*.php` scripts to make them work in other directories is left as an exercise.

```
<link rel="stylesheet" href="<?php print $myWwwPath; ?>layersmenu.css" type="text/css"></link>
```

This line is needed to specify the style sheet. In the `layersmenu.css` bundled with the package I have distinguished the styles related to the PHPLM from the styles related only to the demo page. You can easily find the PHPLM related styles in the `layersmenu-horizontal_menu.ihtml` and `layersmenu-sub_menu.ihtml` templates; you can remove from the style sheet all styles not used in your templates.

```
<?php include ($myDirPath . "libjs/layersmenu-browser_detection.js"); ?>
<script language="JavaScript" type="text/javascript" src="<?php print $myWwwPath; ?>libjs/layersmenu-library.js"></script>
<script language="JavaScript" type="text/javascript" src="<?php print $myWwwPath; ?>libjs/layersmenu.js"></script>
```

These lines are needed to include some needed JavaScript code. W.r.t. the first of these three lines, I do know that I could include the browser detection JS code just the same way of the subsequent two lines; I include it this way to avoid some errors with Netscape 4, as you can read in the `CHANGELOG` of the 2.3.4 release:

```
********
* NOTE *
********
[...] include
layersmenu-browser_detection.js in your pages as shown in the examples;
do not include it as you do for layersmenu-library.js, layersmenu.js,
and layerstreemenu-cookies.js; embed it *statically* in your page's code
as in the examples, otherwise you can trigger errors in Netscape 4
if loading of lib/layersmenu-browser_detection.js is completed too late.
```

You can disregard this trick if you are not interested in supporting Netscape 4 on your pages.

Now let us examine the PHP code before the `</head>` tag. We need to include the template class and the layers menu classes:

```
include ($myDirPath . "lib/template.inc.php");
include ($myDirPath . "lib/layersmenu.inc.php");
```

Then we instantiate the `LayersMenu` class:

```
//$mid = new LayersMenu(140, 20, 20);
$mid = new LayersMenu();
```

As you can see in the commented line, you can pass three optional parameters to the class constructor; such parameters are considered in the documentation generated through phpDocumentor; they are used as default values for DOM properties that cannot be estimated by old browsers; I have chosen reasonable values for them, hence I suggest you to instantiate the class without specifying any parameter.

Then there are some commented lines that show how you can set - and eventually change - some default paths.

Using the script in the PHPLM package directory tree “as is” corresponds to the following commented out settings:

```
/* TO USE RELATIVE PATHS: */
//$mid->setDirroot(".");
//$mid->setLibdir("lib/");
//$mid->setLibjsdir("libjs/");
//$mid->setLibjswww("libjs/");
//$mid->setImgdir("images/");
//$mid->setImgwww("images/");
/* either: */
//$mid->setTpldir("templates/");
//$mid->setHorizontalMenuTpl("layersmenu-horizontal_menu.ihtml");
//$mid->setSubMenuTpl("layersmenu-sub_menu.ihtml");
/* or: (disregarding the tpldir) */
//$mid->setHorizontalMenuTpl("templates/layersmenu-horizontal_menu.ihtml");
//$mid->setSubMenuTpl("templates/layersmenu-sub_menu.ihtml");
```

in fact, if you uncomment the code above, you obtain just the same HTML+JS output code. However, if you try to use the script in a different directory, it does not work; as an example, if you create the `FOOBAR` directory in the package directory tree, copy there the `example-hormenu.php` script, and then load `FOOBAR/example-hormenu.php` instead of `example-hormenu.php`, you get many errors. To get `FOOBAR/example-hormenu.php` working, uncomment the subsequent lines:

```
/* TO USE ABSOLUTE PATHS: */
//$mid->setDirroot($myDirPath);
//$mid->setLibdir($myDirPath . "lib/");
//$mid->setLibjsdir($myDirPath . "libjs/");
//$mid->setLibjswww($myWwwPath . "libjs/");
//$mid->setImgdir($myDirPath . "images/");
//$mid->setImgwww($myWwwPath . "images/");
//$mid->setTpldir($myDirPath . "templates/");
//$mid->setHorizontalMenuTpl("layersmenu-horizontal_menu.ihtml");
//$mid->setSubMenuTpl("layersmenu-sub_menu.ihtml");
```

and these initial variables definitions:

```
/* TO USE ABSOLUTE PATHS: */
//$myDirPath = "/home/pratesi/public_html/phplayersmenu/";
//$myWwwPath = "/~pratesi/phplayersmenu/";
```

(obviously, adjust the above paths to your setup).

To provide another example, let us suppose that we unpack PHPLM in the "system wide" `DocumentRoot`, that, on Mandrake Linux, is by default `/var/www/html`, and then copy `/var/www/html/phplayersmenu/example-hormenu.php` as `/var/www/html/phplayersmenu/FOOBAR/example-hormenu.php`. Everything is OK if you load `http://localhost/phplayersmenu/example-hormenu.php`, whereas you get errors if you load `http://localhost/phplayersmenu/FOOBAR/example-hormenu.php`. To get the `FOOBAR/example-hormenu.php` working, as before, uncomment the subsequent lines:

```
/* TO USE ABSOLUTE PATHS: */
//$mid->setDirroot($myDirPath);
[...]
```

and use these initial variables definitions:

```
/* TO USE ABSOLUTE PATHS: */
//$myDirPath = "/home/pratesi/public_html/phplayersmenu/";
//$myWwwPath = "/~pratesi/phplayersmenu/";
$myDirPath = "/var/www/html/phplayersmenu/";
$myWwwPath = "/phplayersmenu/";
```

As you can see, in this example the difference between `$myDirPath` and `$myWwwPath` corresponds to the `DocumentRoot`, i.e. to `/var/www/html`. A frequent error consists of using the same string both as a "`dirPath`" and as a "`wwwPath`"... ehm... for obvious reasons, usually, the web server `DocumentRoot` does not correspond to the operating system root directory :-).

Obviously, through a proper use of the `setDirroot()-setTpldir()` methods, you can completely change the directories organization w.r.t. the defaults used in PHPLM.

A note about the usage of `setTpldir()`: when you change the templates directory through this method, the **default** templates filenames+paths are updated accordingly; however, if you use non default filenames for the templates, after the usage of `setTpldir()`, to get things working, you have to set explicitly the templates filenames through the `set*MenuTpl()` methods. In general, you should proceed just this way: set the templates directory through `setTpldir()`, and then set the templates filenames through `set*MenuTpl()`... if you choose another way and things do not work, then please do not complain :-).

Subsequent lines are needed to set the down and the forward arrow images:

```
$mid->setDownArrowImg("down-nautilus.png");
$mid->setForwardArrowImg("forward-nautilus.png");
```

in my opinion, they are rather self-explanatory.

```
$mid->setMenuStructureFile($myDirPath . "layersmenu-horizontal-1.txt");
$mid->parseStructureForMenu("hormenu1");
```

The first line above sets the menu structure file; the second line tells the menu system to parse the

current menu structure and correspondingly update related variables; the argument of `parseStructureForMenu()` is the name to be given to the menu whose structure has to be parsed.

```
//$mid->newHorizontalMenu("hormenu1", 12);
$mid->newHorizontalMenu("hormenu1");
```

This method prepares a horizontal menu; the commented line evidences that the method has an optional parameter that is a vertical margin (in pixels) to set the position of a layer a some pixels **above** the ordinate of the "father" link.

The final part of the "`head`" section:

```
$mid->printHeader();
/* alternatively:
$header = $mid->makeHeader();
print $header;
*/
```

outputs some JavaScript code; I suggest you to put this method call just here, before the end of the "`head`" section, to send to the browser such JS code before than any page element.

Further on, we can find PHP code that outputs a table corresponding to the "menu bar":

```
<?php
$mid->printMenu("hormenu1");
/* alternatively:
$hormenu1 = $mid->getMenu("hormenu1");
print $hormenu1;
*/
?>
```

the menu bar obtained is shown in the following screenshot.

Finally, just before the `</body>` tag, we have to put the following PHP code:

```
<?php
$mid->printFooter();
/* alternatively:
$footer = $mid->makeFooter();
print $footer;
*/
?>
```

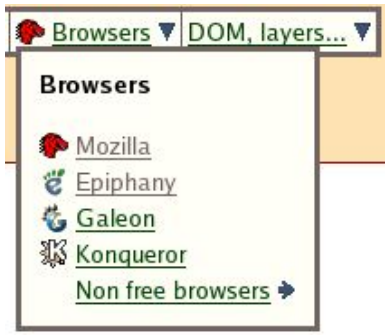
This method outputs the HTML+JS code for all submenu layers and, finally,

```
<script language="JavaScript" type="text/javascript">
<!--
loaded = 1;
// -->
</script>
```

This flag allows to the browser to show the menu layers, hence it can be set to `1` only at the end of the page, when all layers have been loaded and, hence, they are completely defined and everything is ready. I place the footer at the end of the page because this way the loading of the sub-menu layers does not delay the visualization of the other page elements.

Probably, the menu system may work for your browser also if put elsewhere some of the above pieces of code. However, in general, if you put them elsewhere, you may hit problems with a supported browser and/or you can obtain non valid HTML code, whereas all examples in the package output valid XHTML 1.0 Transitional pages. Hence, I recommend to use the menu system just as explained above.

Let us consider the following screenshot:



The “menu bar”, that contains the “Browsers” and “DOM, layers...” items, corresponds to the `layersmenu-horizontal_menu.ihtml` template, while the layer corresponding to the “Browsers” submenu, that contains the “Mozilla”, “Galeon”, “Konqueror”, and “Non free browsers”, corresponds to the `layersmenu-sub_menu.ihtml` template. You can customize the look of the menu changing the style sheet entries and the named templates; when you edit the templates, keep in mind the following things:

- only the parts between “`<!-- BEGIN template -->`” and “`<!-- END template -->`” are parsed and used by the menu system; outside you can write everything you want; as an example, the parts outside can be used to write some notes;
- do not touch the “`<!-- BEGIN foobar -->`” and “`<!-- END foobar -->`”: they are block delimiters;
- the parts inside braces, as “`{link}`”, are replaced by the template engine with the values of the corresponding variables; you can safely remove patterns that you do not need, if you know what you are doing.

The available templates can be deeply customized, depending on how much you are skilled.

The behavior of the menu system can be customized through some JS variables in `libjs/layersmenu.js`:

```
useTimeouts = 1;
timeoutLength = 1000;   // time in ms; not significant if useTimeouts = 0;
shutdownOnClick = 0;
```

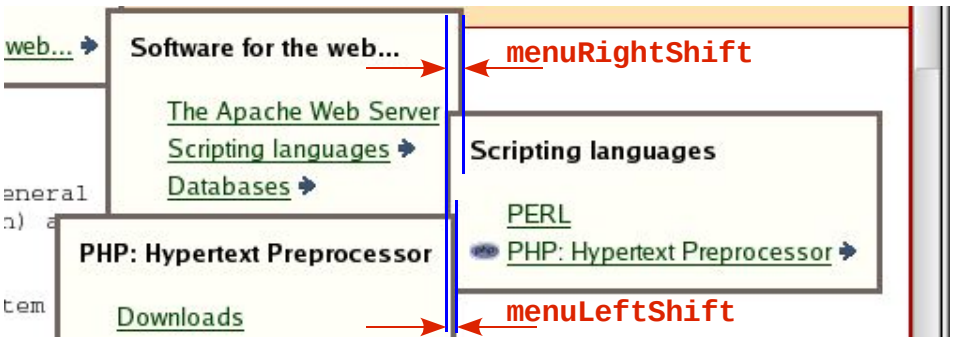
By default, if the mouse pointer is outside from the menu layers, all sub-menu layers disappear after a given amount of time, corresponding to `timeoutLength` milliseconds; obviously, you can change the value of `timeoutLength` according to your preferences. This “timeout feature” can be easily disabled setting `useTimeouts = 0`. Furthermore, if you set `shutdownOnClick = 1`, all sub-menu layers disappear when you click outside the menu. This two feature can coexist, i.e., if you set `useTimeouts = 1` and `shutdownOnClick = 1`, all sub-menu layers disappear if the timeout elapses or if you click outside the menu. You should avoid to enable also the `shutdownOnClick` if the onclick event has to be used in a different way by other code used on the page. Please note that subsequent lines of `libjs/layersmenu.js` automatically enable `shutdownOnClick` in any case for browsers that, with PHPLM, either have troubles with the timeout feature or are not able to use it, i.e. for Netscape 4, Opera 5 and 6, and Internet Explorer 4:

```
if (Konqueror21 || Opera56 || IE4) {
        useTimeouts = 0;
}
if (NS4 || Opera56 || IE4) {
        shutdownOnClick = 1;
}
```

Let us consider other variables set in `libjs/layersmenu.js`:

```
menuLeftShift = 6;
menuRightShift = 10;
```

These values are expressed in pixels; their effect can be easily evidenced through an example, referring to the following screenshot.



The right part of the "Software for the web" layer overlaps with the "Scripting languages" layer by an amount equal to `menuRightShift` pixels; the left part of the "Scripting languages" layer overlaps with the "PHP: Hypertext Preprocessor" layer by an amount equal to `menuLeftShift` pixels.

6.2 - The example-vermenu.php script

The `example-vermenu.php` script is completely analogous to `example-hormenu.php`, hence there is no need to examine also its code. The only significant difference between the two scripts is that `example-vermenu.php` does not contain the portions of code needed to use absolute paths.

6.3 - The example-hormenu_and_vermenu.php script

This example script foresees inclusion of two menus on the same page. Let us consider the following lines of code, where we find the significant differences w.r.t. `example-vermenu.php`:

```
$mid->setDownArrowImg("down-nautilus.png");
$mid->setForwardArrowImg("forward-nautilus.png");
$mid->setMenuStructureFile("layersmenu-horizontal-1.txt");
$mid->parseStructureForMenu("hormenu1");
//$mid->newHorizontalMenu("hormenu1", 12);
$mid->newHorizontalMenu("hormenu1");
$mid->setMenuStructureFile("layersmenu-vertical-2.txt");
$mid->parseStructureForMenu("vermenu1");
//$mid->newVerticalMenu("vermenu1", 12);
$mid->newVerticalMenu("vermenu1");
```

After calling the `newHorizontalMenu()` method, you must not instantiate another object from the `LayersMenu` class. Using the same object, i.e. `$mid`, you have to choose another menu structure file (that, eventually, could be a file already used elsewhere), i.e. `layersmenu-vertical-2.txt`. Then you ask the menu system to parse such menu structure file using again the `parseStructureForMenu()` method, and, subsequently, you use the `newVerticalMenu()` method to generate another menu. Finally, in the page you use twice the `printMenu()` method: once for the "`hormenu1`" menu and once for the "`vermenu1`" menu. There is no difference for the use of the `printHeader()` and `printFooter()` methods, that are called once (not twice), just as for previous examples. The above considerations can be easily generalized if you want to put more than two menus on the same page... you can look at the `index.php` example script if you want to see something really complicated (and unusual;-).

6.4 - Some underlying rationales in the Layers Menus design

As it has been shown in the previous section, you can put more than one layers menu on the same page and, in any case, you have to put only one header and only one footer on the page. In other words, to put N menus on the page, you will use N menu bars, but only one header and only one footer, and, furthermore, you must instantiate and use only one object of the `LayersMenu` class. This approach could seem a bit strange at a first glance: maybe you could expect that, to prepare N menus, you have to instantiate N objects of the `LayersMenu` class and to prepare N headers, N menu bars, and N footers. This approach would allow to aggregate, for each menu, the header, the menu bar, and the footer into a single block of output code. This way, N menus would simply correspond to N blocks of HTML+JS output code, just as for the JS Tree Menu, that is considered further on in this document. In my opinion, this approach would imply simpler and more intuitive APIs, but it would imply some relevant problems, as the resulting menus would behave just as independent systems of layers. As an example, you could open two menus on the same page and some visible layers of the first menu could overlap some visible layers of the second menu; this is not the normal behavior of menus of the most commonly used desktop environments and of the most commonly used program GUIs. To avoid this problem, each menu should be aware of the existence of the other menus on the same page. Furthermore, a JS function like `shutdown()`, that is used to hide all layers on the page, has to hide the layers of all menus on the page, not only the layers of one menu, hence only one `shutdown()` function has to be used, and such function needs to know informations about all the menus on the page; in my opinion, this is difficult and tricky to be implemented with this approach, i.e. if each layers menu corresponds to a different object. Hence, I have considered all layers menus on the page as a single menu system, prepared through a single instance of a PHP class; within an object, each menu is identified through its name.

6.5 - The DB support: the example-db-hormenu.php script

Let us examine the differences due to retrieval of data from the DB instead of from a text file.

The first, evident difference: you do not choose a menu structure file through the `setMenuStructureFile()` method. To define the DB connection parameters, you have to use the `setDBConnParms()` method, that is completely analogous to the PEAR `DB::connect()` method:

```
$mid->setDBConnParms("pgsql://postgres:postgres@localhost/phplayersmenu");
//$mid->setDBConnParms("mysql://mysql:mysql@localhost/phplayersmenu");
```

As you can guess looking at this example code, we pass to the method the Data Source Name, i.e. just the connection string that will be used by PEAR DB to open the DB connection. For a PostgreSQL/MySQL DB, the `dsn` has the following syntax:

```
pgsql://dbuser:dbpass@dbhost/dbname
mysql://dbuser:dbpass@dbhost/dbname
```

Just as the `DB::connect()` method, `setDBConnParms()` has two parameters; the second one (not used in the above example) is optional and is a boolean variable to choose if the DB connection has to be persistent; the default value is "`false`", i.e. non persistent connection are the default.

Now we have to set the table where the menu items are stored:

```
/* THE DEFAULTS FOR THE DEFAULT LANGUAGE TABLE:
$mid->setTableName("phplayersmenu");
```

The default table name, according to the dumps provided in the `DUMPS` directory, is just "`phplayersmenu`", but you may choose a different table name through the `setTableName()` method.

The `setTableFields()` method allows to choose non-default field names. You have to pass to it an associative array that associates the default field names to the corresponding field names that

you want to use. Specification of all fields is not mandatory: you have to specify only non-default field names; specification of the other field names is optional. You can use the `setTableFields ()` method also to disable the use of non mandatory fields; as an example, to disable the use of icons, leaving unchanged all other field names, you can use the following code:

```
$mid->setTableFields(array(
        "icon"  => ""
));
```

Some commented out lines of code in the `example-db-hormenu.php` script provide other useful hints about the `setTableFields()` method.

The `setTableName_i18n()` method allows to choose a non default name for the i18n table.

The `setTableFields_i18n()` method is completely analogous to the `setTableFields()` method.

In the script you can find also some other commented lines of code that show how to use data taken from the PgMarket "`categories`" and "`categories_i18n`" tables. It is also shown how to use the `setPrependedUrl()`: obviously, the "`categories`" table cannot store the URLs corresponding to products categories, as they depend on the PgMarket configuration variables. However, just use the category "`id`" for the `[link]` field:

```
$mid->setTableName("categories");
$mid->setTableFields(array(
[...]
        "link"          => "id",
[...]
));
```

and then set the string to be prepended

```
$mid->setPrependedUrl("/~pratesi/pgmarket/shopping/index.php?id=");
```

Obviously, in a real implementation, the string passed to `setPrependedUrl()` will not be "hardcoded" as above: it will be prepared using the PgMarket's `wwwroot` configuration variable.

When you have chosen all the above DB-related parameters, instead of `parseStructureForMenu()`, you have to use the `scanTableForMenu()` method, that foresees two parameters. The first one is the name to be attributed to the menu whose structure has to be parsed, just as for `parseStructureForMenu()`. The second field is optional; it is a string corresponding to the "`language`" field of the i18n table and it sets the i18n language; you can omit it (or pass an empty string) if you do not need internationalization. Let us consider the following line of the `example-db-hormenu.php` script:

```
//$mid->scanTableForMenu("hormenu1");
$mid->scanTableForMenu("hormenu1", "it");
```

The "`it`" string sets the internationalization language to Italian. In fact, in the resulting menu, according to `DUMPS/[pgsql|mysql].demo_data.dump`, on the resulting page you will read "Navigatori" instead of "Browsers" and "Navigatori non liberi" instead of "Non free browsers".

7 - JavaScript Tree Menus

The JS Tree Menus provide a user interface analogous to the Mozilla Bookmarks Manager and to the most commonly used file managers.

They have more strict requirements w.r.t. the layers menus and provide complete functionality only to browsers sufficiently DOM-compliant for the purpose at hand, i.e.:

- Mozilla (versions 0.9.1+ are suggested)
- Netscape 6.1+ and other browsers based on Mozilla, e.g. Epiphany and Galeon
- Konqueror 3.0+ and browsers based on it, e.g. Safari
- Opera 7.x
- Internet Explorer 4, 5, 5.5, 6

The following browsers are not supported, as supporting them is either not possible at all or really too hard:

- Netscape 4.x
- Konqueror 2.x
- Lynx and Links
- Opera 5.x and 6.x

However, full accessibility is provided for the above browsers: the Tree Menus always appear completely exploded (and no node can be collapsed) on them, and this guarantees a rather good accessibility for them.

Sorry for Netscape 4; my choice of not supporting it for the JavaScript Tree Menu makes me very sad, also because it has been the browser that has disclosed me the world of Internet; but, alas, it is time to begin abandoning it... thank you so much, and goodbye...

Adding a JS Tree Menu to your page is easier than adding a Layers Menu, as you do not need to output three parts of code, i.e. header, menu bar, footer: you need to output only one block of HTML+JS code. Contrarily to what is foreseen for the layers menus, to put N tree menus on the page, you have to output only N blocks of code, one for each Tree Menu. Furthermore, if a page contains more than one tree menu, you want to expand and collapse items of each menu without affecting the expansion state of the other menus on the page, just as when you run more than one instance of a file manager on your desktop. Hence, you could think that, if a page contains more than one tree menu, there is no reason to consider the set of tree menus as a single menu system and that instantiating a menu object for each tree menu on the page is a good approach. However, as it will be clarified further on in this document, the management of cookies suggests to use a unique numbering of items for all trees in the page, hence the same approach used for layers menus is used also for tree menus: the same class instance is used for all tree menus on the page. Furthermore, as we will see further on, if you put on the page both layers menus and tree menus, you can use the same class instance for all menus on the page, but you can use, as well, two distinct class instances: one for all the layers menus on the page, one for all the tree menus on the page.

7.1 - The example-treemenu.php script

To get started with the JS Tree Menus, let us examine the code of the `example-treemenu.php` script.

Let us consider the `<head>...</head>` section. Two style sheets are used:

```
<link rel="stylesheet" href="layersmenu.css" type="text/css"></link>
<style type="text/css">
```

```
<!--
@import url("layerstreemenu.css");
//-->
</style>
```

The second style sheet, i.e. `layerstreemenu.css`, is linked with a different syntax to hide it to old browsers like Netscape 4, because the style defined in it is needed for new browsers, but Netscape 4 has troubles with it... if you are curious about this point, include `layerstreemenu.css` using the same syntax used for `layersmenu.css`, i.e.

```
<link rel="stylesheet" href="layerstreemenu.css" type="text/css"></link>
```

then load the page with Netscape 4, and look at the result of this change :-)

Then we include some JavaScript code:

```
<?php include ("libjs/layersmenu-browser_detection.js"); ?>
<script language="JavaScript" type="text/javascript" src="libjs/layerstreemenu-cookies.js"></script>
```

The first line is needed for the browser detection, while `layerstreemenu-cookies.js` provides some code needed to handle cookies related to the JS Tree Menus. I know that some users do not like cookies, but, in this case, cookies are really needed. In fact, let us suppose that you expand some items and then click on an item that links another page of the site, using the same navigation tree. When you load the new page, the expansion state of the tree gets lost, because we are speaking about a client-side based menu system, hence the expansion state is known to the browser, but it is not known for the server. To preserve the expansion state when you load a new page, the browser has to store it some way on the client side, and this is the reason why the JS Tree Menu uses cookies. Two cookies are stored: "`expand`" and "`collapse`"; they list, respectively, items that are collapsed (i.e. not expanded) by default (see the "`expanded`" field in the text file format) but have been expanded by the user, and items that are expanded by default but have been collapsed by the user. If cookies are disabled on the client's browser, the tree menu still works, but the expansion state of the tree is lost every time the page is reloaded.

Further on, we include the needed PHP classes:

```
<?php
include ("lib/template.inc.php");
include ("lib/layersmenu.inc.php");
...
```

As we are including the template class, you could wonder which HTML templates are used by the tree menus... well, tree menus do not use any HTML template, but PHPLM uses the templating also for some JavaScript code: look at `*.ijs` files in the `libjs` directory. In particular, the JS tree menus use the `layerstreemenu.ijs` JavaScript template, hence the template class is needed also by the JS tree menus.

Then we instantiate the `LayersMenu` class, define the menu structure file, parse the menu structure for a menu that we call "`treemenu1`", prepare and output the HTML+JS code of a new Tree Menu:

```
//$mid = new LayersMenu(140, 20, 20);
$mid = new LayersMenu();
...
$mid->setMenuStructureFile("layersmenu-vertical-1.txt");
$mid->parseStructureForMenu("treemenu1");
print $mid->newTreeMenu("treemenu1");
/* alternatively:
$mid->newTreeMenu("treemenu1");
$mid->printTreeMenu("treemenu1");
*/
/* alternatively:
$mid->newTreeMenu("treemenu1");
$tree_menu1 = $mid->getTreeMenu("treemenu1");
print $tree_menu1;
*/
```

```
?>
```

7.2 - The example-two_treemenus.php script

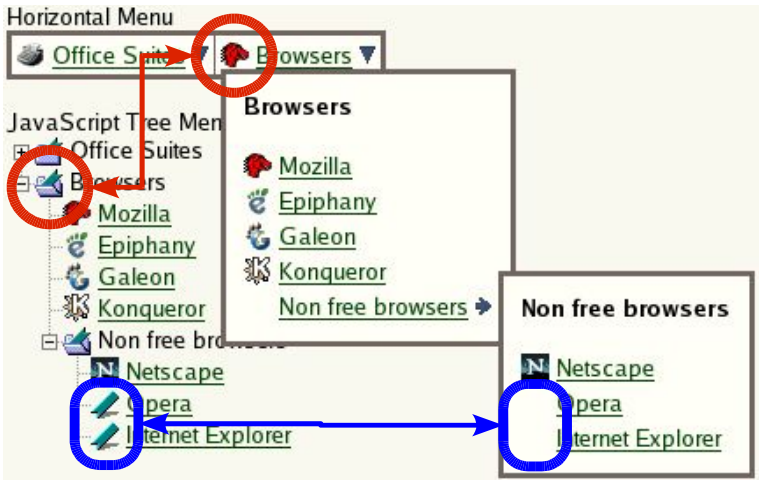
The only difference between this example script and the previous one is in the presence of two tree menus on the page instead of only one. How already explained, the same instance of the menu object is used for all menus on the page. In fact, look at the significant parts of code:

```
A JS Tree Menu
<?php
include ("lib/template.inc.php");
include ("lib/layersmenu.inc.php");
...
$mid = new LayersMenu();
...
$mid->setMenuStructureFile("layersmenu-vertical-1.txt");
$mid->parseStructureForMenu("treemenu1");
print $mid->newTreeMenu("treemenu1");
?>
<br />
Another JS Tree Menu
<?php
$mid->setMenuStructureFile("layersmenu-horizontal-1.txt");
$mid->parseStructureForMenu("treemenu2");
print $mid->newTreeMenu("treemenu2");
?>
```

to prepare the second menu, we do not instantiate another object of the `LayersMenu` class, we use again the `$mid` object; we simply define a different menu name when we call the `parseStructureForMenu()` method. The class could be implemented in another way, to foresee the instantiation of a menu object for each tree menu, but in my opinion this approach would be more difficult and tricky; the current implementation uses the same menu object for all tree menus on the page and, hence, a unique numbering of items for all trees on the page, hence the "`expand`" and "`collapse`" cookies have only to store a list of expanded items and of collapsed items, respectively.

7.3 - The Tree Menu icons and the icons associated to the menu items

Let us consider the following screenshot, that shows a horizontal layers menu and a JS tree menu built using just the same data.



As it has been evidenced in the drawing,

1. an icon has been specified for the "Browsers" item, in fact such icon is show in the horizontal layers menu; however, in the tree menu, the usual folder icon is used for the "Browsers" item

and such icon is ignored;

2. no icon has been specified for the “Opera” and “Internet Explorer” items, in fact no icons are shown for such items in the horizontal layers menu; however, in the tree menu, a default “bookmark” icon is used for such items.

W.r.t. the point 1, the leaf items of the tree menu use only one icon; all the other items use two icons: one for the closed folder, one for the open folder; the data format foresees only one icon for each item, hence icons specified for non-leaf items are ignored.

W.r.t. the point 2, this behavior is borrowed from the Mozilla's Bookmarks Manager and avoids an unpleasant appearance for the tree menus.

7.4 - Themes

The appearance of the JS Tree Menu (and of the PHP Tree Menu, mentioned further on) can be customized through the style sheet entries (`.phplmnormal`, `a.phplm:*`, but do not change `.treemenudiv`) and changing the icons. Some icons themes are available in the `THEMES` directory; icons are named `tree_*.png`, i.e. `tree_collapse.png`, `tree_collapse_corner.png`, and so on. You should avoid to use Jpeg icons for the tree menus, as Jpeg does not foresee transparencies; I suggest to use PNG icons. As you can understand reading `layerstreemenu.css`, the size of icons of the tree menu theme is 16×18 pixels. This size leads to a perfect imitation of the “Manage Bookmarks...” window of Mozilla; I have chosen this size also because

- it is almost the same size used for shortcut icons (“favicons” are 16×16);
- a height larger than 16 pixels allows to use sufficiently large (i.e. readable) fonts for the text; 18 pixels seems to correspond to a fine trade off.

I recommend to use 16×18 images for the tree menu theme icons, and (like for the layers menus) 16×16 images as icons associated to the items; if you want to use a different size for the tree menu theme icons, change `layerstreemenu.css` accordingly.

In the following, some remarks about PNGs.

Note that Internet Explorer has problems with transparencies on RGB PNGs, while it seems that it renders correctly transparencies for indexed PNGs... no progress since from the past century: IE 4.0 was affected by this problem, IE 6.0 is still affected by this problem and, nowadays, it is the only browser still affected by this elementary problem... Not any surprise: after all, Internet Explorer is a Microsoft product. If you want to avoid problems to that retrograde browser and you need to use transparencies, use indexed PNGs and avoid RGB PNGs.

No GIF icons are bundled with the package, for patent related issues and because PNG is a better format than GIF. However, some old browsers do not render correctly transparencies for (both RGB and indexed) PNGs, whereas they do not have any problem with GIF transparencies. If you do not care of patent issues related to GIF and you want to avoid rendering concerns to old browsers, to use GIF icons for the tree menu theme, use the `setTreeMenuImagesType()` method, passing it the `"gif"` extension; then use the same file names with the `.gif` extension instead of the `.png` extension: `tree_*.gif`.

8 - How to put Layers Menus and JavaScript Tree Menus on the same page

8.1 - The example-hormenu_and_treemenu.php script

This example script shows how you can put both a layers menu and a tree menu on the same page using the same class instance for both them.

We have to include also the JavaScript sources needed by the layers menu:

```
<?php include ("libjs/layersmenu-browser_detection.js"); ?>
<script language="JavaScript" type="text/javascript" src="libjs/layersmenu-library.js"></script>
<script language="JavaScript" type="text/javascript" src="libjs/layersmenu.js"></script>
<script language="JavaScript" type="text/javascript" src="libjs/layerstreemenu-cookies.js"></script>
```

We include the needed PHP classes and instantiate the unique object that we will use to prepare the two menus:

```
<?php
include ("lib/template.inc.php");
include ("lib/layersmenu.inc.php");

//$mid = new LayersMenu(140, 20, 20);
$mid = new LayersMenu();
```

We define a horizontal layers menu and output the header of the menu system, as already seen:

```
$mid->setDownArrowImg("down-nautilus.png");
$mid->setForwardArrowImg("forward-nautilus.png");
$mid->setMenuStructureFile("layersmenu-horizontal-1.txt");
$mid->parseStructureForMenu("hormenu1");
//$mid->newHorizontalMenu("hormenu1", 12);
$mid->newHorizontalMenu("hormenu1");

$mid->printHeader();
?>
```

Further on, within the HTML code of the page, we output the menu bar of the layers menu:

```
<?php
$mid->printMenu("hormenu1");
?>
```

Then we define a tree menu and output its HTML+JS code within the HTML code of the page:

```
<?php
$mid->setMenuStructureFile("layersmenu-vertical-1.txt");
$mid->parseStructureForMenu("treemenu1");
print $mid->newTreeMenu("treemenu1");
?>
```

Finally, we output the footer of the menu system:

```
<?php
$mid->printFooter();
?>
```

Note: if the layers menu and the tree menu use the same data (i.e. the same items), the tree menu can be prepared using the following lines of code

```
<?php
$mid->setMenuStructureFile("layersmenu-horizontal-1.txt");
$mid->parseStructureForMenu("hormenu1");
print $mid->newTreeMenu("hormenu1");
?>
```

However, we do not need to call the `setMenuStructureFile()` and the `parseStructureForMenu()` methods twice, both times with the same argument; i.e. the following lines are not needed

```
$mid->setMenuStructureFile("layersmenu-horizontal-1.txt");
$mid->parseStructureForMenu("hormenu1");
```

and the following line is sufficient to prepare the tree menu

```
<?php
print $mid->newTreeMenu("hormenu1");
?>
```

Analogous considerations hold true if data are retrieved either from a string or from a database instead of from a menu structure file.

8.2 - The example-hormenu_and_treemenu-bis.php script

This example script shows another way to put both a layers menu and a tree menu on the same page, i.e. using two distinct class instances, one for each menu, instead of using the same class instance for both them.

The code is almost the same of `example-hormenu_and_treemenu.php`, as it can be evidenced using `diff`:

```
--- example-hormenu_and_treemenu.php [...]
+++ example-hormenu_and_treemenu-bis.php [...]
[...]
 JavaScript Tree Menu
 <?php
-$mid->setMenuStructureFile("layersmenu-vertical-1.txt");
-$mid->parseStructureForMenu("treemenu1");
-print $mid->newTreeMenu("treemenu1");
+$treemid = new LayersMenu();
+$treemid->setMenuStructureFile("layersmenu-vertical-1.txt");
+$treemid->parseStructureForMenu("treemenu1");
+print $treemid->newTreeMenu("treemenu1");
 ?>
[...]
```

There are only two differences between the two scripts; in the second one, a second object of the `LayersMenu` class is instantiated for the tree menu:

```
...
$mid = new LayersMenu();
...
$treemid = new LayersMenu();
```

and then, the subsequent lines of PHP code related to the tree menu are just the same, apart from the fact that the `$treemid` object is used instead of `$mid`.

The two scripts do not output exactly the same HTML+JS code. In fact, with the first one, the numbering of tree items is consecutive with the numbering of items of the horizontal menu, i.e. a unique numbering is used for items of both menus, whereas, with the second one, each menu uses its own numbering, and numbering of the tree menu starts with 1. This also means that the tree menu of `example-hormenu_and_treemenu-bis.php` uses the same items numbering as in `example-treemenu.php` and `example-two_treemenus.php` and, hence, its management of the "`expand`" and "`collapse`" cookies is perfectly compatible with the one performed by `example-treemenu.php` and `example-two_treemenus.php`.

8.3 - The example-layersmenus_and_treemenus.php script

This example script shows how you can put two layers menus and two tree menus on the same page using the same class instance for both them.

In practice, it is a "merge" of `example-hormenu_and_vermenu.php` and `example-two_treemenus.php`; as in `example-hormenu_and_treemenu.php`, the same class instance is used for the layers menus and the tree menus.

8.4 - The example-layersmenus_and_treemenus-bis.php script

If you put some layers menus and some tree menus on the page, for the tree menus you can use the same class instance used for the layers menus, but you can use another class instance as well.

This example script shows another way to put two layers menus and two tree menus on the same page, i.e. using two distinct class instances, one for the layers menus and one for the tree menus, instead of using the same class instance for both them.

The code is almost the same of `example-layersmenus_and_treemenus-bis.php`, as it can be evidenced using `diff`:

```
--- example-layersmenus_and_treemenus.php [...]
+++ example-layersmenus_and_treemenus-bis.php [...]
[...]
 A Tree Menu
 <?php
-$mid->setMenuStructureFile("layersmenu-vertical-1.txt");
-$mid->parseStructureForMenu("treemenu1");
-print $mid->newTreeMenu("treemenu1");
+$treemid = new LayersMenu();
+$treemid->setMenuStructureFile("layersmenu-vertical-1.txt");
+$treemid->parseStructureForMenu("treemenu1");
+print $treemid->newTreeMenu("treemenu1");
 ?>
 <br />
 Another Tree Menu
 <?php
-$mid->setMenuStructureFile("layersmenu-horizontal-1.txt");
-$mid->parseStructureForMenu("treemenu2");
-print $mid->newTreeMenu("treemenu2");
+$treemid->setMenuStructureFile("layersmenu-horizontal-1.txt");
+$treemid->parseStructureForMenu("treemenu2");
+print $treemid->newTreeMenu("treemenu2");
 ?>
[...]
```

The differences are completely analogous to the ones evidenced between `example-hormenu_and_treemenu.php` and `example-hormenu_and_treemenu-bis.php`.

9 - Accessibility solutions: PHP Tree Menu, Plain Menu

PHPLM provides also two menu systems that do not require JavaScript to work: the PHP Tree Menu and the Plain Menu. To use these two menu system you have to instantiate an object from the `XLayersMenu` class, that extends the `LayersMenu` class.

9.1 - The PHP Tree Menu

The PHP Tree is similar to the Dijkstra's Tree Menu[3] [DIJKSTRA] and uses some of its code; it has just the same look as the JavaScript Tree Menu (the same HTML tags and styles are used), but it requires a server supporting PHP, i.e. it is a server-side based menu system, rather than a client-side one.

Let us see how to use this menu system. The `example-treemenu.php` is related to the use of the JS Tree Menu; the use of the PHP Tree Menu is almost the same, but there are some differences, highlighted in the following.

The PHP Tree Menu does not use JavaScript, hence the following lines are not needed:

```
<?php include ("libjs/layersmenu-browser_detection.js"); ?>
<script language="JavaScript" type="text/javascript" src="libjs/layerstreemenu-cookies.js"></script>
```

(the expansion status of the tree is considered further on). Another difference: the template class is not needed, as the PHP Tree Menu does not use HTML and/or JS templates; hence, the following line of PHP code is not needed

```
include ("lib/template.inc.php");
```

We have to include also `lib/layersmenu-noscript.inc.php` and to instantiate the `XLayersMenu` class instead of `LayersMenu`:

```
include ("lib/layersmenu.inc.php");
include ("lib/layersmenu-noscript.inc.php");
$mid = new XLayersMenu();
```

Finally, we have to use the `newPHPTreeMenu()` method instead of `newTreeMenu()`:

```
print $mid->newPHPTreeMenu("treemenu1");
```

The expansion status of a tree (i.e. the set of expanded items) is decided through

- the value of the "`p`" variable in the URL query string;
- the `setPHPTreeMenuDefaultExpansion()` method, that allows to set a default value for "`p`"; the `$phpTreeMenuDefaultExpansion` value is considered only if "`p`" is not defined in the query string; by default, `$phpTreeMenuDefaultExpansion` is an empty string.

Using the "`expanded`" field as it is done for the JS Tree Menu would be not easy, hence for the PHP Tree Menu I have preferred to provide the `setPHPTreeMenuDefaultExpansion()` method; if you want to use only the PHP Tree Menu, you can omit the "`expanded`" field for all items, as such field is ignored.

To decide the default expansion to be passed to `setPHPTreeMenuDefaultExpansion()`, you can proceed as in the following:

- start without using that method;
- click to open items to obtain the expansion state that you want to obtain by default;
- at this point, the URL query string provides the value of "`p`" to be passed to the `setPHPTreeMenuDefaultExpansion()` method.

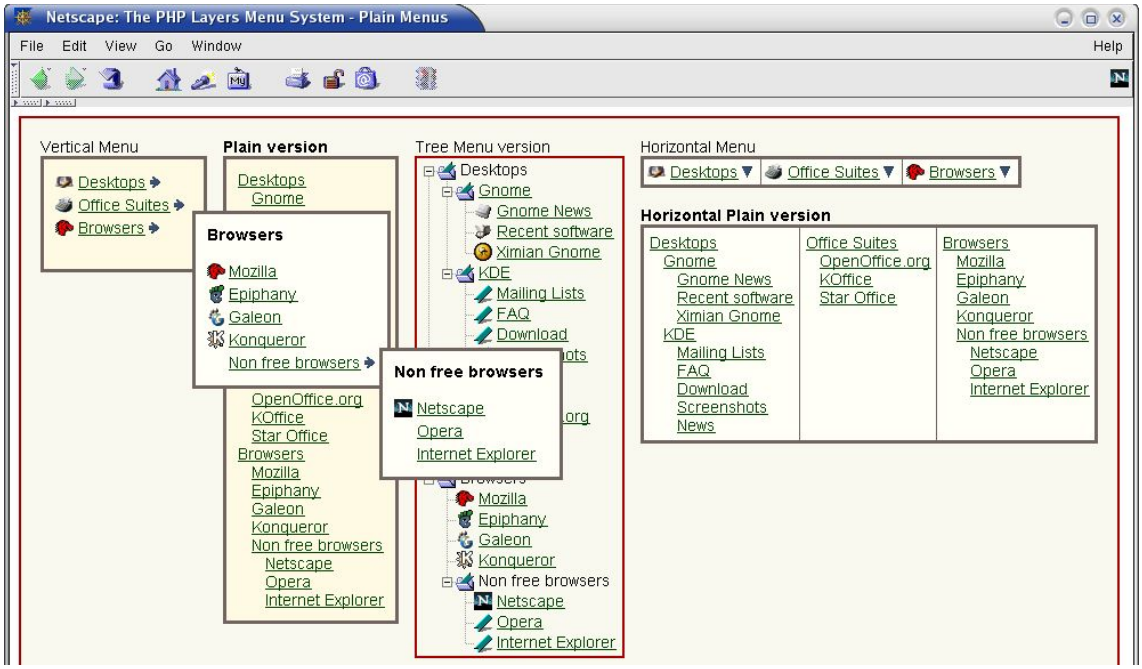
3 But there are many significant differences: as an example, I have completely redesigned the HTML code used for the menu, to use DIVs rather than tables and contextually overcome some formatting problems that are evidenced in some particular cases and that seem not fixable using tables.

Note that, if you put more than one PHP tree menu on the page (as in the `index.php` demo script), the same query string is used for all that menus and the only one `$phpTreeMenuDefaultExpansion` is used for all that menus the query string, i.e. the `setPHPTreeMenuDefaultExpansion()` method should be called only once, to specify all items to be expanded for all that menus.

I have noted that some people use the PHP Tree Menu as the main menu for their site. Well, all sincerely, I consider it as a an accessibility solution to be sent to some browsers, but I suggest to prefer the JS Tree Menu, as it does not require to submit a request to the server to expand/collapse an item and, hence, it saves bandwidth and it is much faster, especially on slow connections. However, if you use frames for your pages and you put the PHP Tree Menu in the navigation frame, if your menu tree is not big, then updating times to expand/collapse an item are not a big problem, and you may prefer the PHP Tree Menu to the JS Tree Menu to use something that is browser independent (functionality is not reduced for old browsers) and does not require JavaScript.

9.2 - The Plain Menu

The Plain Menu does not require a server supporting PHP and does not require JavaScript to work: it is based of an indented and completely expanded list of menu items; both a vertical version and a horizontal version are available, as you can see in the example provided in the following screenshot, that has been prepared with Netscape 4.



Let us highlight the few differences w.r.t. the use of the PHP Tree Menu. First of all, `layerstreemenu.css` is needed only for the tree menus, hence the following lines are not needed for plain menus:

```
<style type="text/css">
<!--
@import url("layerstreemenu.css");
//-->
</style>
```

The plain menus use templates, hence the template class has to be included:

```
include ("lib/template.inc.php");
```

Finally, we have to use the `newPlainMenu()` method instead of `newPHPTreeMenu()`:

```
print $mid->newPlainMenu("treemenu1");
```

Use `newHorizontalPlainMenu()` instead of `newPlainMenu()` method to obtain a horizontal plain menu instead of a vertical plain menu (look also at the previous screenshot).

Probably, the JS Tree Menu is preferable to the vertical version in many cases, as it provides a completely expanded tree to non supported browsers, and, in my opinion, an expanded tree is more pleasant than an expanded and indented list; however, you can consider also that the Plain Menu is more compact, i.e. it covers a smaller area on the page w.r.t. the JS Tree Menu, as you can see in the previous screenshot.

10 - Conversions between the text format and the DB

The `lib/layersmenu-process.inc.php` code provides the `ProcessLayersMenu` class, that extends `LayersMenu`. The `ProcessLayersMenu` class provides two methods, `getMenuStructure()` and `getSQLDump()`, that convert menus data from the database to a text menu structure and vice versa.

If your menu data are available as a text file, you can easily obtain a corresponding SQL dump, using the `getSQLDump()` method, that outputs an SQL dump corresponding to the items of a menu. The output SQL commands should not imply compatibility issues with a DBMS different from PostgreSQL and MySQL, but, if you have problems with another DBMS, please let me know. An example of use is provided in the `example-filetodb.php` script.

If your menu data are stored in a database, you can easily obtain a corresponding menu structure text output, using the `getMenuStructure()` method, that outputs the menu structure text corresponding to the items of a menu. An example of use is provided in the `example-dbtofile.php` script.

The interfaces of both methods are documented in the previously mentioned hypertext documentation produced through phpDocumentor (and, of course, in the comments of the corresponding code).

11 - Help! My web server does not support PHP!

Doesn't your server support PHP? Sure? Why it doesn't, given that PHP is a cross-platform and widely used scripting engine? :-) However, PHPLM may be useful also if your pages will go on a server not supporting PHP. Load the bundled `index-static.html` in your browser and you will realize that, apart from the PHP Tree Menus, that inherently require a server supporting PHP, it works just the same way as the `index.php` demo script. In practice, if you have PHP installed on your development workstation, you can prepare the layers menu HTML+JavaScript related code on your workstation and then use this code without relying on the PHP engine: you can include that code with another scripting engine, using server side includes, frames (look at `example-frame.html`), and so on. Obviously, PHPLM is much more handy if your server supports PHP, but, I repeat, this is not a must.

12 - Customized versions: the contributed patches

In the PATCHES directory, the package bundles patches that provide interesting customizations of the menus behavior. Related documents are bundled in the same directory, hence this manual does not provide their documentation. Please consider that such patches are not tested as deeply as the “official” (unpatched) version and that they can support a lower amount of browsers w.r.t. the unpatched version.

13 - Hint: caching the header, the menubar, and the footer

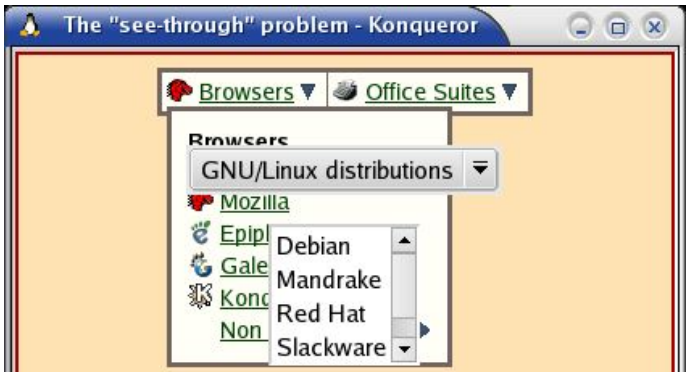
You may want to avoid to rebuild all the HTML+JS code of the menu(s) for each page request; to this end, I suggest you to cache the code of header, menubar, and footer, either using a DB table, as in PgMarket (look at the "`cache`" table), or using some files.

14 - Appendix A - FAQ

Many of the most frequently asked questions that I have received are covered by the previous sections of the manual. Here I reply to FAQs that do not fit well in the other sections.

14.1 - The “see-through” problem

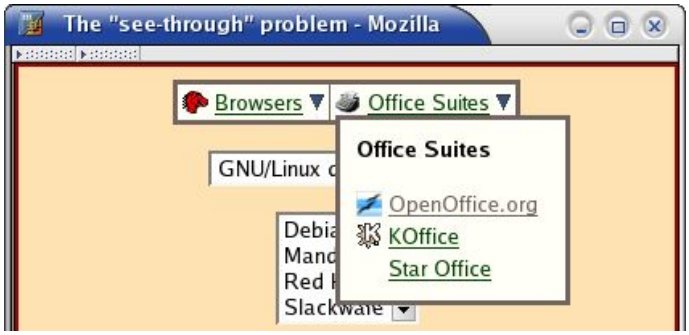
Many people have issued this bogus bug report, in many different ways... well, look at the following screenshot.



Please, look at me and read my lips: this is ***not*** a bug of the PHP Layers Menu system. Some interesting references about this topic:

http://www.geocrawler.com/mail/msg.php3?msg_id=4426606&list=119
http://www.webreference.com/dhtml/diner/seethru/
http://www.intranetjournal.com/ix/msg/36271.html

Briefly, this problem is not fixable and is ***not*** z-index related. It has been completely solved in Mozilla 1.4 (and, hence, also in Netscape 7.1) and in Opera 7.x, and, according to users' reports, Apple Safari is not affected by it.



On the contrary, Konqueror is affected, but it is a “young” browser and I suppose that it's only a matter of time, as I feel that its developers are clever guys. Internet Explorer has not made any progress since from the past century: apart from the `iframe` object, IE 6.0 is affected by this problem just at the same extent as IE 4.0 was. If you are angry due to this problem, then complain of it to Microsoft, not to me: I use Mozilla, and Mozilla developers have satisfied this claim. In the next development branch, probably I will add workarounds to this problem for Konqueror and IE, but, please, do not report me this problem anymore: it is useless to stress me further about this issue. Please also note that such workarounds can only consist of hiding widgets that are ***not*** part of the PHPLM system: in other terms, to workaround this problem, I have to do something that would not be exactly up to me and/or to the PHPLM system.

14.2 - Frames support

Question: *Does PHPLM support frames? If I put the menu in the navigation frame, submenus are hidden by the main frame.*

Answer: You can use the (JS or PHP) tree menu system also with frames; an example is provided in `example-frame.html`. However, if you want to use frames with a horizontal/vertical layers menu, you have to consider that a browser window with two frames is just like two distinct browser windows: the only difference is that, with frames, you have the two browsing canvases grouped inside the same window frame (but each one has its own scrollbar, as an example). If you put the menu bar in one canvas, its submenus stay in the same canvas. Until now, I have never tried to implement a layers menu having the menu bar in one frame and the submenus in another frame, also because I do not consider this as a really necessary feature: including a few lines of PHP code (that prepare the HTML+JS code of the menu) on all pages, you obtain the same effect and you also avoid using frames; the same thing can be done also using server side includes instead of the `include`/`require` PHP instruction.

14.3 - The [link] field used only as a query string

Some people have asked why the menu system does not work correctly if the `[link]` field consists of something like "`?foo=bar`". The `[link]` field is **not** appended to the URL of the current page; if you want to use the `[link]` field as a (query) string to be appended to the current URL (or to another URL/string), use the `setPrependedUrl()` method, as already explained in this manual.

15 - Appendix B - Some implementation details

In the following, I provide some insights about the internals of the menu system; they are basically some personal notes of mine; some of them are not necessarily immediately understandable :-). However, in my opinion, such notes can be useful if you want to dive into the code to understand it and maybe to customize it... happy reading :-).

15.1 - Browser detection

Basically, the following cases are considered:

- `DOM` - Document Object Model [DOM]
- `NS4` - Netscape 4.x
- `IE4` - Internet Explorer 4

Apart from some details and workarounds, it is made the simplifying assumption that sufficiently recent browsers (Mozilla, NS 6.1+, Konqueror 2.1+, Opera 5+, IE 5+) can be considered compliant to the DOM specifications if only not-so-much-advanced features are employed, while Netscape 4.x and Internet Explorer 4 have to be handled as distinct cases.

Here and there it is necessary to use specific code also for Konqueror, Opera, and Internet Explorer.

15.2 - Working principle

The generation of layers is based on a depth-first search over the tree, as it can be easily understood examining `layersmenu*.txt`. The depth-first search allows to logically arrange in a one-dimension sequence the tree's nodes, just as the directories in file managers providing a tree view (for example, the Gnome's “gmc” and Nautilus, and the MS Windows' file browser) when all branches are expanded, to show all subdirectories.

15.3 - Nodes-layers correspondence

In the categories tree, the “root node” is represented by the “Top” category, that does not appear in the layers menu. Then we have to consider the children of the root node, i.e. the “first-level” subcategories (of “Top”), that have to be handled differently from the other nodes, as they have to be always visible on the page. Finally, we have to distinguish leaf nodes - i.e. the ones without children - (categories without corresponding subcategories) from all the other nodes; such distinction is significant also for the children of the root node (“first-level” subcategories); in the subsequent part of the present document it will be clear why leaf nodes have to be handled a bit differently from the other nodes.

The algorithm employed to generate the needed layers performs a scan of nodes analogous to the one foreseen in the PHP Tree Menu 1.1, authored by Bjorge Dijkstra `<bjorge@gmx.net>` and published at `http://www.zinc.f2s.com/scriptorium/index.php/treemenu` . In fact, the Layers Menu scans the nodes reading a menu structure whose format is analogous to the one used by the PHP Tree Menu 1.1.

15.4 - Implementation driving rules

Each layer lists all the children of a not-leaf node; each of these children has a link to the corresponding URL associated with.

The root node's children (i.e. the subcategories of “Top”) can be considered as belonging to a “special” layer, as they have to be always visible on the page, while, in the general case, all other layers are hidden.

Each layer is defined when it is involved by the depth-first search; its geometrical placement is identified by its top-left corner coordinates and it depends by the route leading to it. In particular, it depends on the position and width of the “father” layer and on size and scroll of the browsing window; the vertical coordinate has to be related to the vertical coordinate of the link corresponding to the “father node”.

A `LMPopUpL([layername])` function is defined, that is used to set as visible the layer labeled as `[layername]`; for a given layer, the label (name) is obtained as `'L[N]'`, where `[N]` is a progressive integer number, identifying the position (line) of the father node inside the depth-first search result (starting from 0 for the root node).

A `shutdown()` function is defined, that is used to set as hidden all the layers (only links related the root node's children are always visible on the page).

A `LMPopUp([menuName])` function is defined, that is used to set as visible all layers corresponding to the route leading to the corresponding layer, starting from the link related to a root node's child. All other layers have to be hidden, hence such a function starts calling the `shutdown ()` and the set as visible all the layers on the name route.

It is imposed that the Layers Menu behavior is analogous to the behavior of the Gnome and KDE main menus and of the “Start” menu of MS Windows. This last rule implies the following sub rules.

- If `shutdownOnClick` is set to `1` in `layersmenu.js`, a click event outside of visible layers has to trigger the hiding of all the layers (`shutdown()`).
- The positioning of the mouse pointer over a link corresponding to a non-leaf node has to trigger the visibility of all the layers on the route to the layer containing the “children of the link”.
- The positioning of the mouse pointer over a link corresponding to a leaf node (category without subcategories) has to trigger the visibility of all the layers on the route to the layer under the mouse pointer. This rule implies also that, if the mouse pointer exits from a link corresponding to a non-leaf node and goes over a link corresponding to a leaf node, then the “highest level” layer among the visible ones will become hidden. For a leaf node which is a root node's child, this rule simply implies a `shutdown()`.

15.5 - Useful hints

The geometry and the appearance of the menu is determined by parameters chosen invoking suited methods (refer to the class implementation in `layersmenu.inc.php`) and by the HTML templates, provided in the `templates` directory.

In `lib/layersmenu.inc.php`, at the end of the `_parseCommon()` method, there is a commented out “`if`” block to handle the root node's children differently from children with hierarchical level larger than 1. This is intended as an alternative that is useful if you want to associate links only to leaf nodes, i.e. when non-leaf subcategories are intended to be empty (you are associating products only to leaf subcategories) and, hence, it is useless to visit them.

I have decided to add to each layer a title, corresponding to the name of the father node, because there is not a strict visual correspondence between a given layer and its “father link”, but this correspondence may become more evident if the layer has a title recalling the “father”.

When the mouse passes over a link corresponding to a non leaf node, the starting ordinate of the layer to be popped up is set to the vertical coordinate of the link minus an offset (unless the size of the browsing window imposes some “wrapping”); such offset is chosen when invoking the `new*Menu()` method.

Alas, Netscape 4 is not able to return the vertical coordinate of the link by relying on the DOM tree; hence, for Netscape 4, such coordinate is estimated (with very, very good accuracy) through the

vertical coordinate that the pointer has when it triggers the `onmouseover` event; that's why the vertical coordinate of the mouse pointer position is continuously detected through a portion of JavaScript code specifically devoted to Netscape 4. A threshold for the triggering of the layer position is used too, to avoid flickering of the layer to continuous repositioning if the mouse is moved around the link.

How do you detect if a node is a leaf?

Well, with the tree items ordered according to a depth-first search, a node is a leaf if one of the following two conditions is true;

- it is the last node of the tree;
- it is not the last node of the tree, and the hierarchical level associated to the subsequent node is larger or equal to the current hierarchical level.

It should be rather easy to become convinced of this statement considering the view corresponding to a completely expanded tree menu.

16 - Appendix C - The GNU Free Documentation License

GNU Free Documentation License
Version 1.1, March 2000

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself,

plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section entitled "History", and its title, and add to

```
   it an item stating at least the title, year, new authors, and
   publisher of the Modified Version as given on the Title Page.  If
   there is no section entitled "History" in the Document, create one
   stating the title, year, authors, and publisher of the Document as
   given on its Title Page, then add an item describing the Modified
   Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for
   public access to a Transparent copy of the Document, and likewise
   the network locations given in the Document for previous versions
   it was based on.  These may be placed in the "History" section.
   You may omit a network location for a work that was published at
   least four years before the Document itself, or if the original
   publisher of the version it refers to gives permission.
K. In any section entitled "Acknowledgements" or "Dedications",
   preserve the section's title, and preserve in the section all the
   substance and tone of each of the contributor acknowledgements
   and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document,
   unaltered in their text and in their titles.  Section numbers
   or the equivalent are not considered part of the section titles.
M. Delete any section entitled "Endorsements".  Such a section
   may not be included in the Modified Version.
N. Do not retitle any existing section as "Endorsements"
   or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or
appendices that qualify as Secondary Sections and contain no material
copied from the Document, you may at your option designate some or all
of these sections as invariant.  To do this, add their titles to the
list of Invariant Sections in the Modified Version's license notice.
These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains
nothing but endorsements of your Modified Version by various
parties--for example, statements of peer review or that the text has
been approved by an organization as the authoritative definition of a
standard.

You may add a passage of up to five words as a Front-Cover Text, and a
passage of up to 25 words as a Back-Cover Text, to the end of the list
of Cover Texts in the Modified Version.  Only one passage of
Front-Cover Text and one of Back-Cover Text may be added by (or
through arrangements made by) any one entity.  If the Document already
includes a cover text for the same cover, previously added by you or
by arrangement made by the same entity you are acting on behalf of,
you may not add another; but you may replace the old one, on explicit
permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License
give permission to use their names for publicity for or to assert or
imply endorsement of any Modified Version.


5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this
License, under the terms defined in section 4 above for modified
versions, provided that you include in the combination all of the
Invariant Sections of all of the original documents, unmodified, and
list them all as Invariant Sections of your combined work in its
license notice.

The combined work need only contain one copy of this License, and
multiple identical Invariant Sections may be replaced with a single
copy.  If there are multiple Invariant Sections with the same name but
different contents, make the title of each such section unique by
adding at the end of it, in parentheses, the name of the original
author or publisher of that section if known, or else a unique number.
Make the same adjustment to the section titles in the list of
Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History"
in the various original documents, forming one section entitled
"History"; likewise combine any sections entitled "Acknowledgements",
and any sections entitled "Dedications".  You must delete all sections
entitled "Endorsements."


6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents
released under this License, and replace the individual copies of this
License in the various documents with a single copy that is included in
the collection, provided that you follow the rules of this License for
verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute
it individually under this License, provided you insert a copy of this
License into the extracted document, and follow this License in all
other respects regarding verbatim copying of that document.
```

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (c) YEAR YOUR NAME.
> Permission is granted to copy, distribute and/or modify this document
> under the terms of the GNU Free Documentation License, Version 1.1
> or any later version published by the Free Software Foundation;
> with the Invariant Sections being LIST THEIR TITLES, with the
> Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
> A copy of the license is included in the section entitled "GNU
> Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

17 - References

[PHPDOC] phpDocumentor - http://phpdoc.org/

[DIJKSTRA] PHP Tree Menu, authored by Bjorge Dijkstra, whose home site was on http://www.zinc.f2s.com/scriptorium/index.php/treemenu; nowadays, this is a broken link, and, alas, I have not found elsewhere that menu system.

[DOM] Document Object Model (DOM) - http://www.w3.org/DOM/