HTML_TreeMenu Documentation

Introduction

HTML_TreeMenu is a set of PHP classes to enable the easy creation of HTML based tree menus. It currently has the ability to create both DHTML, static and listbox menus. All have the prerequisite of Javascript, however the Javscript code could be used manually, without the PHP classes.

Authors

HTML_TreeMenu was written by Richard Heyes and Harald Radi. Contributions/patches have also been received and incorporated from various people.

Features

- Easy to learn OO based API
- DHTML (traditional) or listbox (<select>) output styles
- Multiple menus per page
- Cross browser DOM compatible DHTML
- Optional branch status persistence using cookies
- Optional static mode without DHTML (still requires Javascript support)
- Per node icon with alternate “expanded” icon
- Per node CSS class specification
- Per node link targets
- Per node Javascript event specification with custom onExpand, onCollapse and onToggle events
- Ability to specify from menu creation if a node is expanded by default, and if it should be made to be visible (ie its parents are expanded)

Structure Overview

There are five classes in total:

HTML_TreeMenu	The “top level” tree class.
HTML_TreeNode	The node class.
HTML_TreeMenu_Presentation	An abstract base class for the next two classes.
HTML_TreeMenu_DHTML	Produces a DHTML style menu.
HTML_TreeMenu_Listbox	Produces a listbox menu.

The HTML_TreeNode class is used to create the structure which is added to an instance of the HTML_TreeMenu class. This is in turn passed to an instance of either of the presentation classes (DHTML or Listbox). These two “presentation” classes have a *printMenu()* method which can be called to print the resulting menu.

Example

See the example.php file for the code. This code makes a simple menu with two root nodes, each with five nested nodes. The code goes about creating the initial HTML_TreeMenu object, and then creates the nodes to be added. The nodes are then added to the menu object, which itself is then passed to an instance of the HTML_TreeMenu_DHTML object (through the constructor) and also an HTML_TreeMenu_Listbox object. Then follows some HTML in which the *printMenu()* method of each presentation object is called to show the menus. Note the use of references when assigning the return of *addItem()*. Failure to assign by reference will cause problems.

API Reference

HTML_TreeMenu

Methods

&addItem(object &$node)
This method is used to add a HTML_TreeNode to the tree. It takes a HTML_TreeNode as its sole argument and returns a reference to the node inside the TreeMenu object.

createFromStructure(array $params)
This method is an extremely useful one if you already have a tree structure defined using one of the supported tree classes. It takes said tree structure and returns a tree menu based upon it. This takes the work out of traversing your tree and creating the structure yourself. The supported tree structures are Richard Heyes' Tree class (http://www.phpguru.org/tree.html) and Wolfram Kriesings' Tree class available through PEAR http://pear.php.net/Tree. The $params argument should be an associative array which can consist of the following:

structure	The tree structure
type	The type. Defaults to 'heyes'. Can also be 'kriesing'.
nodeOptions	Default HTML_TreeNode options which are used whilst building the menu. In the case of my own Tree class, these will be merged with the tag data.

The return value is the HTML_TreeMenu object.

createFromXML(mixed $xml)
This method will create an HTML_TreeMenu object from the supplied $xml argument. This argument can either be a string containing the XML, or a PEAR::XML_Tree object. If the argument is a string, the method will attempt to *require()* the XML_Tree class using standard PEAR techniques (ie: r*equire_once('XML/Tree.php')*), and then create an XML_Tree object based on the string. The method will then convert the XML_Tree object to a Tree class using my own Tree class (available here: http://phpguru.org/tree.html) so this file (Tree.php) **MUST** be include()ed or require()ed before calling this method. If the Tree class cannot be found, this method will *die()*. Once converted, the method will then use the *createFromStructure()* method described above to create an HTML_TreeMenu object and return it. For further information on using this method and the XML schema see the case study below.

HTML_TreeNode

Methods

Constructor([array $options [, array $events]])
The constructor handles setting up the node object based on the options supplied. The $options argument should be an associative array which can consist of the following:

text	Title of the node, defaults to blank.
link	HREF of the link, defaults to blank.
icon	Filename of the icon. Should be in the images directory as supplied to the presentation object.
expandedIcon	Filename of the icon to be used when the node is expanded.
class	CSS class for this node, defaults to blank.
expanded	Default expanded status of this node. Defaults to *false*, and has no effect on non dynamic presentations.
linkTarget	Target for the link. Defaults to linkTarget of the presentation class.
isDynamic	If this node is dynamic or not. Defaults to true.
ensureVisible	If true, this node will be made visible regardless of the expanded settings and clientside persistence. Defaults to false.

The second argument is an associative array of Javascript events and associated handler code. This can also include three custom events: onexpand, oncollapse, and ontoggle, which should be self explanatory. This argument could also be supplied as the *events* key in the first argument.

setOption(string $option, mixed $value)
Use this method to set any of the options after the node has been created. The option names are the same as those in the constructor, and take similar values.

&addItem(object &$node)
This is similar in every respect to the *addItem()* method of the HTML_TreeMenu class and is used to add child nodes.

HTML_TreeMenu_DHTML

Methods

Constructor(&$structure [, array $options [, $isDynamic]])
This sets up the presentation object with the given structure and options. The structure should be an HTML_TreeMenu object. The options argument should be an associative array, and can consist of the following:

images	The folder to look in for images. Defaults to “images”.
linkTarget	Target for any links. Can be set here instead of for every node. Defaults to “_self”.
defaultClass	Default CSS class to use. Defaults to blank.
usePersistence	Whether to use clientside persistence or not (with cookies). Defaults to true.
noTopLevelImages	Whether to skip the display of the first level of branch images if there are multiple root nodes. Defaults to false.

The thrid argument *$isDynamic* can be used to specify Whether the entire tree is dynamic or not. Defaults to true.

printMenu([$options])
This is a method inherited from the abstract HTML_TreeMenu_Presentation class and is used to print the menu. The optional argument *$options* should be an associative array which can consist of the same options as the *$options* argument of the constructor.

HTML_TreeMenu_Listbox

Methods

Constructor(&$structure [, array $options])

This sets up the presentation object with the given structure and options. The structure should be an HTML_TreeMenu object. The options argument should be an associative array, and can consist of the following:

promoText	The text that appears at the top of the listbox. Defaults to “Select...”
indentChar	The character used to indent the nodes. Defaults to “ ”.
indentNum	How many *indentChars* to use per indentation level.
linkTarget	Target for any links. Can be set here instead of for every node. Defaults to “_self”.
submitText	Text for the submit button. Defaults to “Go”.

printMenu([$options])

This is a method inherited from the abstract HTML_TreeMenu_Presentation class and is used to print the menu. The optional argument *$options* should be an associative array which can consist of the same options as the *$options* argument of the constructor.

Case study: Creating a tree menu using XML

The HTML_TreeMenu::createFromXML() method is an extremely useful one if you have a treemenu to maintain and wish to do so in simpler fashion than updating PHP code every time a change is need. The XML can be kept in a file and read in to create the menu, with little more than a few lines of PHP code necessary.

Pros:

- Very easy to maintain
- Little coding required by you
- No loss of control over presentation by way of XML tag attributes

Cons:

- Slower

As a result of:

- Increased amount of included code (Tree class, XML_Tree class, XML_Tree_Node class, XML_Parser class)
- XML parsing

the decreased speed is significant enough to warrant caching of the resulting HTML_TreeMenu object, either in a users' session or perhaps a file based cache. (PEAR has a couple of candidates – Cache & Cache_Lite). If you do cache the treeMenu, then it's entirely concievable that it may end up being faster than building it in PHP every time, so the speed drawback becomes a non-issue.

The XML schema you should use consists of two tags, <treemenu> and <node>. The <treemenu> tag is the root element and there should only ever be one of these. The <node> tag defines a node in the tree and can be nested as much as want. The <node> tag can take as attributes any of the options which can be passed to the HTML_TreeNode constructor (eg. text, link, icon etc). Some example XML:

```
<?xml version="1.0"?>
<treemenu>
      <node text="Node 1" icon="folder.gif" />
      <node text="Node 2" icon="folder.gif" />
      <node text="Node 3" icon="folder.gif" />
      <node text="Node 4" icon="folder.gif">
            <node text="Node 4_1" icon="folder.gif" />
            <node text="Node 4_2" icon="folder.gif" />
            <node text="Node 4_3" icon="folder.gif" />
            <node text="Node 4_4" icon="folder.gif" />
      </node>
      <node text="Node 5" icon="folder.gif" />
</treemenu>
```

As you can see it's perfectly OK to use the XML shortcut <node ... /> instead of <node ... ></node> if a node has no child nodes. This XML will create a treeMenu with five root nodes, with the fourth having four child nodes. The code you would need to use to create the HTML_TreeMenu object is as follows:

```
<?php
      require_once('HTML/TreeMenu.php');
      require_once('XML/Tree.php');
      require_once('Tree.php'); // Tree class from phpguru.org

      $xml = file_get_contents('treemenu.xml');
      $treeMenu = HTML_TreeMenu::createFromXML($xml);
?>
```

You're then free to use the $treeMenu structure with either of the presentation classes as normal.

License

The package is distributed under the BSD license. Wishlist fulfilment is always appreciated of course:

Richard Heyes	http://phpguru.org/wishlist
Harald Radi	http://www.amazon.com/...