# PHP DBTreeView

User manual

Version 1.0

# 1 Table of contents

**1 TABLE OF CONTENTS 2**

**2 CHANGES 3**

**3 INTRODUCTION 4**

**4 FEATURES 4**

**5 LICENSE 5**

**6 REQUIREMENTS 5**

**7 INSTALLATION 5**

**8 MY FIRST TREEVIEW 6**

8.1 OVERVIEW 6
8.2 DEFINE THE SOURCE OF DATA 6
8.3 DEFINE THE MANNER TO IDENTIFY THE NODES, AND TO RETRIEVE HIS CHILDREN 6
8.4 CREATE THE REQUEST HANDLER 6
8.5 PREPARE YOUR SCRIPT TO RECEIVE THE REQUESTS 7
8.6 INSERT THE TREEVIEW ELEMENT IN THE HTML PAGE 8

**9 A COMPLETE EXAMPLE 8**

**10 NODE PROPERTIES 10**

10.1 IDENTIFYING A NODE: NODE ATTRIBUTES 10
10.2 NODE TEXT: HTML CONTENT 11
10.3 NODE ICON 11
10.4 NODE URL LINK 11
10.5 NODE HASCHILDREN PROPERTY 12
10.6 NODE SETCHILDREN METHOD 12

**11 STYLE 12**

**12 ENCODING 12**

**13 ERROR HANDLING 13**

**14 API REFERENCE 13**

# 2 Changes

| *Date* | *Change description* |
|---|---|
| 17 august 2007 | First publication |
| | |
| | |
| | |
| | |
| | |

# 3 Introduction

PHP DBTreeView is a useful web control to display information in a tree like windows explorer. This component is designed to extract dynamic content from database, so that the tree content is updated step by step.

This project uses the latest web technologies like Ajax (JavaScript, DHTML, XMLHttpRequest) and PHP 5.

The use of this tool doesn't require JavaScript knowledge. You only have to be familiar with PHP 5.

You should be familiar with the concept of object oriented programming (see ref[1]). You should understand what's an interface (see ref [2]). If you aren't familiar with that, I suggest you to read the references given on page foot.

# 4 Features

- *Freeware and open source.*
- *Ajax technology*: each branch is loaded on demand without page refresh.
- *Node Icon*: control for each node icon. Closed, and open icons for directories.
- *Database*: works with every kind of database, or other data source.
- *Useful node identification*: uses an array of strings to easily identify each node.
- *HTML node text*: each node receives text in HTML. You can create nodes with pictures, tables, links, etc.
- *Multiple treeviews on the same page.*
- *Multiple levels loading at once*: you can customize your trees by creating trees with several levels opened at startup. You can choice between: not to preload nodes, to load but not to display nodes, or to load and to display nodes.

[1] http://www.php.net/manual/en/language.oop5.php
[2] http://www.php.net/manual/en/language.oop5.interfaces.php

## 5 License

PHP DBTreeView is free and open source software (GNU Lesser GPL).

The author of this library is Rodolphe CARDON DE LICHTBUER, from Brussels (Belgium). You can contact him at rcardon@free.fr.

For the PHP and JavaScript codes, and the documentation, see LICENSE.txt included in the library.

The pictures in the 'media' directory are NOT part of this library and are used for demonstration only. Some pictures (directory icons, '+' icon, '-' icon) are copyrighted (Microsoft Windows). Don't use these pictures without having the appropriate licenses.

## 6 Requirements

- Web server with PHP 5 (PHP 4 is not supported)
- Browser with JavaScript support (IE, Firefox, …)

This library has been tested on IE 7 and Firefox. It should work on other browsers supporting JavaScript like Opera. You can send your experience with other browsers to the author.

The browser doesn’t need to support Java (Java Sun or MS VM). Java and JavaScript are completely different and independent programming languages.

## 7 Installation

Download the PHP DBTreeView library on http://rcardon.free.fr/dbtreeview.

Copy the ‘lib/dbtreeview’ directory anywhere in your web server. This directory contains

- the PHP file ‘dbtreeview.php’ and ‘dbtreeview_config.php’,
- the JavaScript file ‘treeview.js’,
- the default cascading style sheet ‘treeview.css’,
- and the ‘media’ directory with demo pictures.

# 8 My first treeview

## 8.1 Overview

The creation of a treeview has several steps:

- Define the source of data
- Define the manner to identify each node, and to retrieve his children
- Create the request handler: an object used by the processor to answer ‘what are the children of this node ?’
- Insert the treeview in the HTML page.

## 8.2 Define the source of data

Where do the data come from? Typically, data are extracted from a database.

Assume we have a table ‘MY_TABLE’ with the fields ‘id’, ‘description’, ‘parent’, where ‘id’ is a unique identification number, ‘description’ is a string, and ‘parent’ is the identification number of the parent entry. For example:

| *id* | *description* | *parent* |
|---|---|---|
| 1 | Service A | 0 |
| 2 | Service B | 0 |
| 3 | Service A1 | 1 |
| 4 | Mister M | 2 |
| 5 | Mister J | 1 |

The hierarchical representation is:

- Service A (id=1)
  - Service A1 (id=3)
    - Mister M (id=4)
- Service B (id=2)
  - Mister J (id=5)

## 8.3 Define the manner to identify the nodes, and to retrieve his children

Each node in DBTreeView is identified by an associative array of attributes. Each attribute has a key and a value. Both key and value are strings.

In our example, each node may be identified with only one attribute: his ‘id’ number.

In the case of ‘Service A’, we have

```
$attributes = array(”id” => ”1”);
```

## 8.4 Create the request handler

The request handler is an object used by the processor to answer ‘what are the children of this node?’.

This object must implements the ‘RequestHandler’ interface. This interface contains only one function `handleChildrenRequest(ChildrenRequest $req)`. This function receives the request containing the attributes of the parent node and must return an object of type ‘ChildrenResponse’ containing the children nodes.

For our example, we have:

```
class MyHandler implements RequestHandler{

   public function handleChildrenRequest(ChildrenRequest $req){

      $attributes = $req->getAttributes();

      if(!isset($attributes['id'])){
         die("error: attribute id not given");
      }
      $parentID = $attributes['id'];

      $link = mysql_connect(HOST, USERNAME, PASSWORD)
         or die("Unable to connect.");
      mysql_select_db(MY_DATABASE) or die("Could not select database");

      $query = sprintf("SELECT * FROM mytable WHERE parent='%s'",
               mysql_escape_string($parentID));

      $result = mysql_query($query) or die("Query failed");

      $nodes=array();

      while ($line = mysql_fetch_assoc($result)) {
         $id = $line["id"];
         $nodeId = array("id"=>$id);
         $nodeDescription = $line["description"];
         $node = DBTreeView::createTreeNode($nodeID, htmlspecialchars($nodeDescription));

         //Set the URL link
         $node->setURL(sprintf("./getDetails.php?id=%s");", $id), "targetFrame");

         //add the node to the children node list
         $nodes[] = $node;
      }

      mysql_free_result($result);
      mysql_close($link);

      //We create the response message that must be returned by this function
      $response = DBTreeView::createChildrenResponse($nodes);
      return $response;
   }
}
```

## 8.5 Prepare your script to receive the requests

By default, the same script is used both to display the HTML page containing the treeview and to answer the Ajax requests from the treeview (XML messages).

Your script must begin with a call to the DBTreeView processor. This processor will automatically determine if an Ajax request is done or not. If not, the HTML page will be sent to the client.

The script should be structured like this:

```
<?php
//the path to the DBTreeView library without the ‘/’ character at the end
```

```
define("TREEVIEW_LIB_PATH","./lib/dbtreeview");

//include the DBTreeView classes
require_once(TREEVIEW_LIB_PATH . '/dbtreeview.php');

class MyHandler implements RequestHandler{
  //content of MyHandler
}

try{
   DBTreeView::processRequest(new MyHandler());
}catch(Exception $e){
   echo("Error:". $e->getMessage());
}
// print here the HTML page
?>
```

Your script may not begin with white spaces. The first characters of the file must be the “<?php” tag and you may not print anything before the call to `processRequest`. Otherwise, the script will return malformed XML responses.

## 8.6 Insert the treeview element in the HTML page

The insertion of the HTML requires two steps.

First, include the following code in the <head> HTML section:

```
printf('<script src="%s/treeview.js" type="text/javascript"></script>',
         TREEVIEW_LIB_PATH);
printf('<link href="%s/treeview.css" rel="stylesheet" type="text/css" media="screen"/>',
         TREEVIEW_LIB_PATH);
```

Then, include the treeview itself in the HTML <body> section. You must give the attributes of the root node (in our example, the root node is ‘id’=0), the path to the library, and a treeview ID. If you have several treeviews on the same page, you must use different treeview IDs.

```
$rootAttributes = array("id"=>"0");
$treeID = "treeview1";
$tv = DBTreeView::createTreeView(
      $rootAttributes,
      TREEVIEW_LIB_PATH,
      $treeID);
$tv->setRootHTMLText("Organization");
$tv->printTreeViewScript();
```

# 9 A complete example

We assume that the script below is ‘http://www.mysite.com/mytree.php’ and that the PHP DBTreeView library is installed in ./lib/dbtreeview/ (relative path to mytree.php)

```
<?php

//the path to the DBTreeView library without the ‘/’ character at the end
define("TREEVIEW_LIB_PATH","./lib/dbtreeview");

//include the DBTreeView classes
require_once(TREEVIEW_LIB_PATH . '/dbtreeview.php');

//define the handler that responses to the client requests. This class must implements
the RequestHandler interface. This interface contains only one function
‘handleChildrenRequest’
```

```
class MyHandler implements RequestHandler{

   public function handleChildrenRequest(ChildrenRequest $req){

      $attributes = $req->getAttributes();

      if(!isset($attributes['code'])){
         die("error: attribute code not given");
      }
      $parentCode = $attributes['code'];

      $link = mysql_connect(HOST, USERNAME, PASSWORD)
         or die("Unable to connect.");
      mysql_select_db(MY_DATABASE) or die("Could not select database");

      // Set the character encoding to UTF-8.
      if(!mysql_query("SET CHARACTER SET utf8")){
       throw new Exception('Could not set character set UTF-8.');
      }

      //In our example, the table ‘mytable’ contains records with the fields ‘code’,
‘description’, and ‘parent’.
      $query = sprintf("SELECT * FROM mytable WHERE parent='%s'",
               mysql_escape_string($parentCode));

      $result = mysql_query($query) or die("Query failed");

      $nodes=array();

      while ($line = mysql_fetch_assoc($result)) {
         $code = $line["code"];
         //Prepare HTML node text
         $text = "<b>$code</b> : ".$line["description"];

         //Create the node with given text, and given identification (attributes)
         $node = DBTreeView::createTreeNode(
            $text, array("code"=>$code));

         //Set the URL link
         $node->setURL(sprintf("./getDetails.php?code=%s");", $code));

         //add the node to the children node list
         $nodes[] = $node;
      }

      mysql_free_result($result);
      mysql_close($link);

      //We create the response message that must be returned by this function
      $response = DBTreeView::createChildrenResponse($nodes);
      return $response;
   }
} //class MyHandler

try{
   //if the HTTP command is ‘POST’ with XML data, process the treeview request, otherwise
this function does nothing and the script continues
   DBTreeView::processRequest(new MyHandler());
}catch(Exception $e){
   echo("Error:". $e->getMessage());
}

//HTML content
print("<html>\n<head>\n");
//include the javascript reference in the head section
$scriptHead = sprintf("<script src=\"%s/treeview.js\"
type=\"text/javascript\"></script>",
         TREEVIEW_LIB_PATH);
//include the CSS for treeview file in the head section
$scriptHead = $scriptHead . "\n".
         sprintf('<link href="%s/treeview.css" rel="stylesheet" type="text/css"
media="screen"/>',
         TREEVIEW_LIB_PATH);
print($scriptHead."\n");
print("</head>\n");
?>
<body>
```

```
<h1>Demonstration</h1>
<p>This tree download nodes on demand. When you click on a '+' to see a branch, a request
is sent to the server, without page refresh.</p>
<?php
$rootAttributes = array("code"=>"0");
$treeID = "treev1";
$tv = DBTreeView::createTreeView(
      $rootAttributes,
      TREEVIEW_LIB_PATH,
      $treeID);
$tv->setRootHTMLText("NACEBEL codes");
$tv->setRootIcon("star.gif");
$tv->printTreeViewScript();
?>

</body>
</html>
```

What does it happen when I open the ‘http://www.mysite.com/mytree.php’ in my browser ? The server sends the HTML page with JavaScript code. The JavaScript receives parameters like the attributes of the hidden root node (in our example: ‘code’ = ‘0’).

Then, the JavaScript calls ‘mytree.php’ again with an XML request: what are the children of the node {‘code’ = ‘0’}. The server retrieves information from MySQL, and returns the answer to the browser in XML format.

When the user clicks on a ‘+’ in the tree to expand a branch, the JavaScript calls again the PHP script and so one. The tree is updated without whole page refresh.

In this example, the responder script is the same script as the page containing the treeview. The ‘processRequest’ function automatically detects if the script is called by the treeview, or if the page is called by the browser to display the HTML page.

You also can define another script as responder. If many pages contains the same treeview, you can use the same script responder using `$tv->setScript($URL)`. See API documentation.

# 10 Node properties

This section gives an overview of the node properties. See the API reference for complete documentation.

## 10.1 Identifying a node: node attributes

Each node is identified by an associative array of strings: pairs of keys and values. The key and the value string cannot contain tabulation or line breaks.

You could identify each node with only one attribute, for example ‘id’. But, in many cases, the use of several attributes is useful (ex: filtering). Suppose we have the following case:

- Organization 1
    - Men
    - Women
- Organization 2
    - Men
    - Women

We could identify each node like that:

- “org” => “1”
    - “org” => “1”, “sex” => “m”
    - “org” => “1”, “sex” => “f”
- “org” => “2”
    - “org” => “2”, “sex” => “m”
    - “org” => “2”, “sex” => “f”

For the second level nodes, the SQL request should look like :

```
SELECT INTO mytable ... WHERE org = ... AND sex = ...
```

With this example, you can see the utility to have an array of attributes to identify each node. When the user sends a request to open a node, the parent node attributes are sent to the server and are given as parameters to the function `handleChildrenRequest`.

## 10.2 Node text: HTML content

The ‘DBTreeView::createNode’ function receives HTML for the node text. So you can easily creates nodes containing table, pictures, hyperlink, etc. as content.

If the node text must contains special HTML character like ‘<’, ‘>’, ‘&’, ‘”’, you must first protect the string with the ‘htmlspecialchars’ PHP function.

## 10.3 Node icon

You can specify the ‘opened’ and ‘closed’ icons of each node. If not specified, the default icons are used.

A node without child is always considered as ‘closed’.

## 10.4 Node URL link

You have two possibilities to add a hyperlink to a node: using self made HTML code (<a href=’…’>…</a>) as text node, or using the ‘$node=>setURL’ function.

The setURL function receives one or two arguments. The first argument is the URL. The second optional argument is the target frame.

## 10.5 Node hasChildren property

You can specify if a node doesn’t have any child by setting the ‘hasChildren’ node property to ‘false’. By default, this property is ‘true’.

If the property is true, the node *may* have children (maybe not).

When you click on the ‘+’ button of a node with the ‘hasChildren’=’true’, but without any child, the ‘+’ will simply disappear after the request didn’t return any node.

## 10.6 Node setChildren method

In most cases, you **don’t need** to use this method.

The ‘setChildren’ method is **only** used when multiple levels must be loaded at once. For example, it’s possible to display 2 levels of nodes so that all nodes of the first level are opened. See the ‘examples/multiple-levels.php’ file.

The use of this method has effect on the performance of the loading. Be careful using this method.

# 11 Style

The style of the treeview elements is controlled with CSS (cascading style sheets). You can change the font, the margin between the nodes, the style of the links, etc.

By default, your must include the ‘treeview.css’ CSS file. You can include your own style definition.

# 12 Encoding

The character encoding is a problem that concerns every programmer. Each programmer should be familiar with this issue and should known what is UTF-8, Unicode, etc.

The best choice is to use a Unicode type encoding like UTF-8. This section gives some tips to control the character encoding in HTML page and database transactions.

First, if you web page are encoded in UTF-8, the <head> element of the HTML page MUST contain the following meta tag (see ref[3]):

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
```

If your page contains form with text fields, this tag causes the browser to send the fields content in UTF-8.

Secondly, PHP doesn’t have native support for multi-byte encoding like UTF-8, but it doesn’t matter. A string in PHP is just an array of bytes. If the meta tag above is present, the retrieved fields will already be in UTF-8.

Thirdly, it’s important that you configure the encoding between your server and your database. With MySQL, do the following when the connection is established, before the first SQL query:

```
if(!mysql_query("SET CHARACTER SET utf8")){
    throw new Exception('Could not set character set UTF-8.');
   }
```

For PostgreSQL, you can use (see ref[4]):

```
pg_set_client_encoding($conn, "UTF8");
```

## 13 Error handling

If the server returns invalid XML data (for example if there is an error in your PHP script), the JavaScript will pop-up a window with the server message. You can easily debug your code with that functionality.

## 14 API reference

The full description of the library is contained in the API (Application programmable interface) reference on http://rcardon.free.fr/dbtreeview/, or in the ‘apidoc’ directory.

[3] http://www.w3.org/TR/html401/charset.html
[4] http://www.postgresql.org/docs/8.2/static/multibyte.html