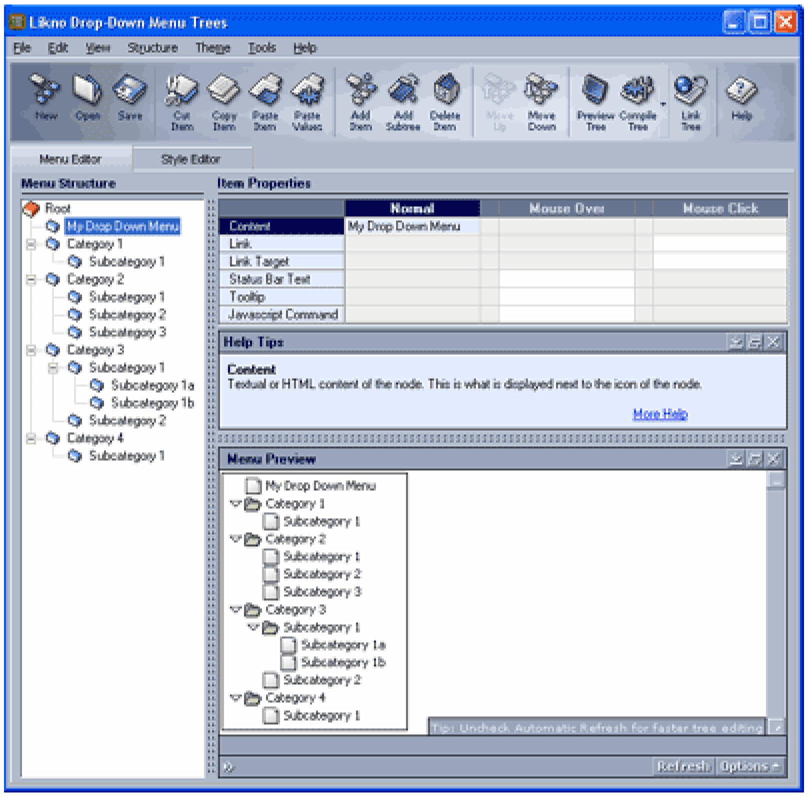
http://www.likno.com

Likno Drop-Down Menu Trees



	Normal	Mouse Over	Mouse Click
Content	My Drop Down Menu		
Link			
Link Target			
Status Bar Text			
Tooltip			
Javascript Command			

Content
Textual or HTML content of the node. This is what is displayed next to the icon of the node.

More Help

User's Guide

Likno Drop-Down Menu Trees Help

The Complete Tool for Creating Powerful Javascript Tree Controls

by Likno Software

Likno Drop Down Menu Trees is a user-friendly interface (GUI) that allows you to create stylish, feature-rich html javascript tree controls for your web pages, with minimal effort and coding.

The present manual will assist you in taking the most out of this software.

The Likno Software team

Likno Drop-Down Menu Trees Help User's Guide

Printed: 7/8/2012

Special thanks to:

We would like to thank all of our users and customers that have contributed their comments and suggestions for improving Likno Drop-Down Menu Trees Help and the services we offer. It is our aim to listen to all of them carefully in order to bring the best possible result for them. We couldn't have done without you!

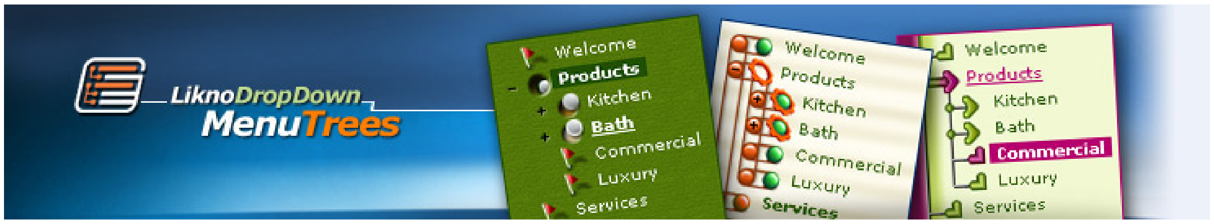
Table of Contents

Introduction

Part I

1 Introduction

1.1 Likno Drop-Down Menu Trees Overview



Likno Drop-Down Menu Trees is more than a cross-browser, platform-independent hierarchical navigation control for your web pages. It's an intuitive and fun way of showing structured information and providing your visitors with the means to manipulate this information in a user-friendly manner.

Menu trees lend themselves as the natural user interface for allowing your visitors to navigate quickly through categories and subcategories of links in your site, while at the same time exposing your sitemap.

Forget the overused, dull styles you've all seen before, now you can select from one of our many original themes comprising of custom icons and styles to suit your individual requirements and dazzle your visitors!

Likno Drop-Down Menu Trees is built using Javascript and DHTML technologies and does not require any plug-ins or extensions to work on your clients' browsers.

You can choose one of the many predefined themes ranging in style from the standard Microsoft Windows tree interface to more interesting and even funny styles which are sure to catch your visitors' attention and keep them coming for more!

You can change the style for the whole tree and even customize each and every item using a different icon and style for the default, mouseover and selected states. The possibilities are endless!

Once your menu tree is ready you can publish it on your website and allow it to...grow! By using our custom javascript API you can allow your visitors' choices to change or rearrange your tree dynamically on their browser. You can add or delete items, and you can sort the items under any branch. You can even generate the menu tree from a database and associate an array of values (e.g. from a recordset) with each leaf.

Major Features Include:

Likno Drop-Down Menu Trees was designed with the end user in mind. Users are accustomed to using tree controls on their every day computer work and require no less than the same functionality from their web-based counterparts.

Simply put, Likno Drop-Down Menu Trees is a complete cross-browser, platform-independent hierarchical navigation control for your web pages (check out product overview).

Some of the main features of Likno Drop-Down Menu Trees are:

   * Cross-browser, tested on all major modern browsers and operating systems (MS Internet Explorer, Firefox, Opera, Konqueror & Safari) (*)
   * A wide variety of different themes comprising of different icons, CSS styles for the nodes, and

background properties. Check out our extensive Theme Gallery!
* Styles can be defined individually for three different states, normal, mouse over and selected
* Icons can be defined for each of the states mentioned above, and for each node, closed or open branch
* Different icons can be individually set for specific nodes (that is, not following the tree theme)
* Any number of CSS classes can be dynamically added and used or changed for different nodes or branches of the tree
* Unlimited number of trees in the same page, each with different styles and properties
* Unlimited nodes and/or levels (depth)
* Use the keyboard arrows to navigate through the tree and expand/collapse branches
* The tree participates in the TAB order of your web page. If you have multiple trees in the same page, you can also use Alt+1, Alt+2, etc. to navigate between them
* Double click or single click to expand/collapse a branch
* Support for either fixed box size trees with scrollbars, or freely expandable
* Trees can be attached to any HTML element on the page
* Support for multiple selection trees (can be used as check boxes), optional use of the CTRL modifier key to add to selection
* Custom context menu with user-defined items, following the tree theme
* Visit link in a target frame, and/or execute a Javascript command when clicking on a node
* Optional tree state persistence across sessions. The state of the tree is saved in a cookie
* Support for tool tips and custom status bar messages on mouse over for each node
* Ability to associate custom user-defined data with each node (and perform sorting and other operations on them)
* Optional auto-collapse property (Microsoft Explorer style)
* Comprehensive client-side API, including functions for manipulating all aspects of the tree, sorting, searching for nodes, dynamically inserting or deleting nodes, selecting a node etc.

(*) Some features may not be available to certain browsers, due to limitations and/or bugs of these browsers.

1.2 About Likno Software

Likno Software is a software development company dedicated to creating quality software for demanding web professionals. Our development team is 100% Internet-oriented, focused on creating tools that minimize your web development efforts.

For detailed **technical support** you can access the following URL:
http://www.likno.com/support_page.html

For **sales** questions you may contact us at **sales@likno.com**.

Our web site is located at http://www.likno.com

User Interface

Part

II

2 User Interface

2.1 Main window layout

The Main window uses two tabbed sections:

The Menu Editor Tab

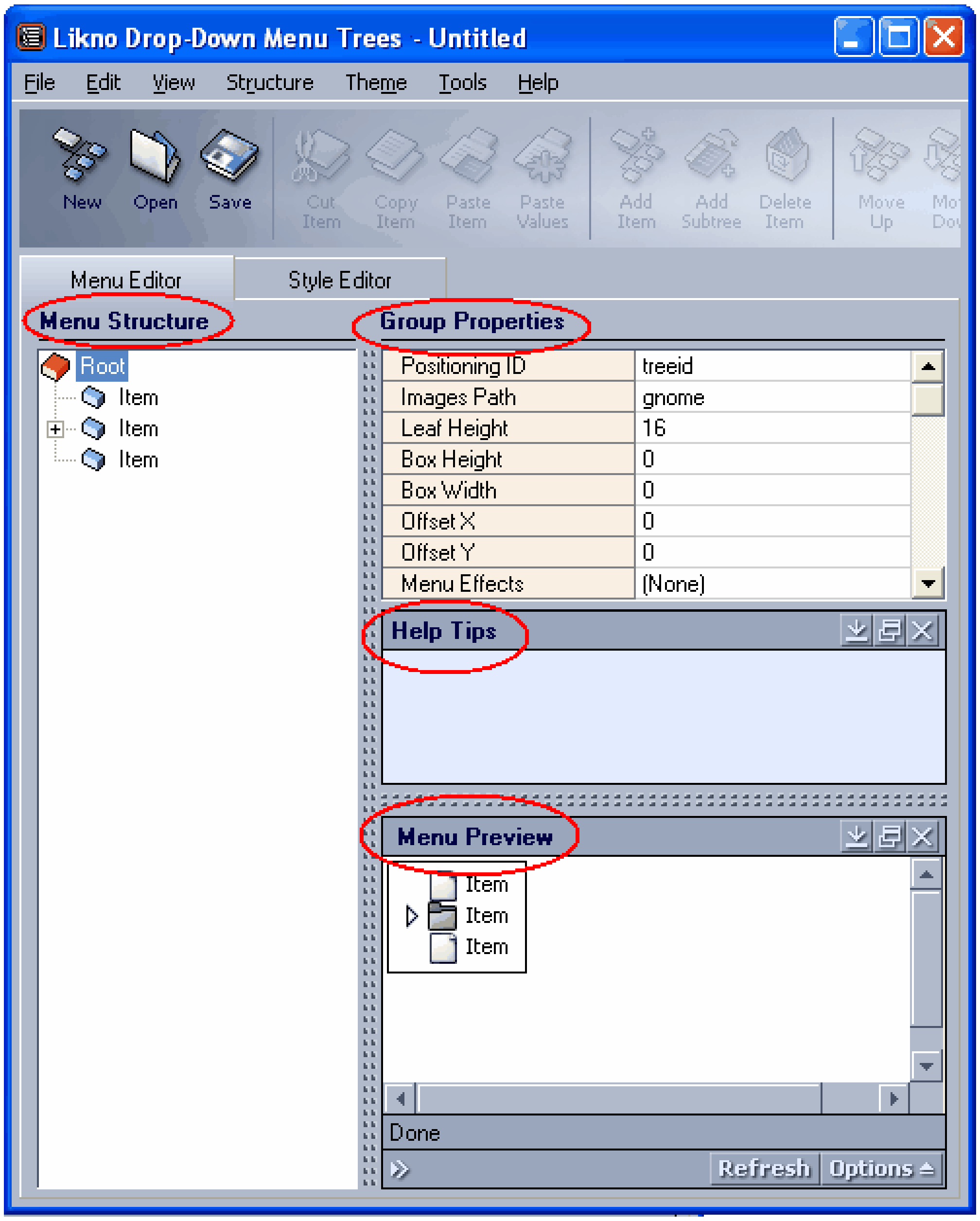
This section is split into four panes:

The Menu Structure Pane
This is where the menu is structured by specifying its elements and its parent-child relationships.

The Properties Pane
This is where the properties of each element are specified.

The Help Tips Pane
This is where a short description is displayed to provide quick help on the selected property.

The Menu Preview Pane
This is where your menu can be previewed in a WYSIWYG IE browser.



Positioning ID	treeid
Images Path	gnome
Leaf Height	16
Box Height	0
Box Width	0
Offset X	0
Offset Y	0
Menu Effects	(None)

The Style Editor Tab

This section is split into four panes:

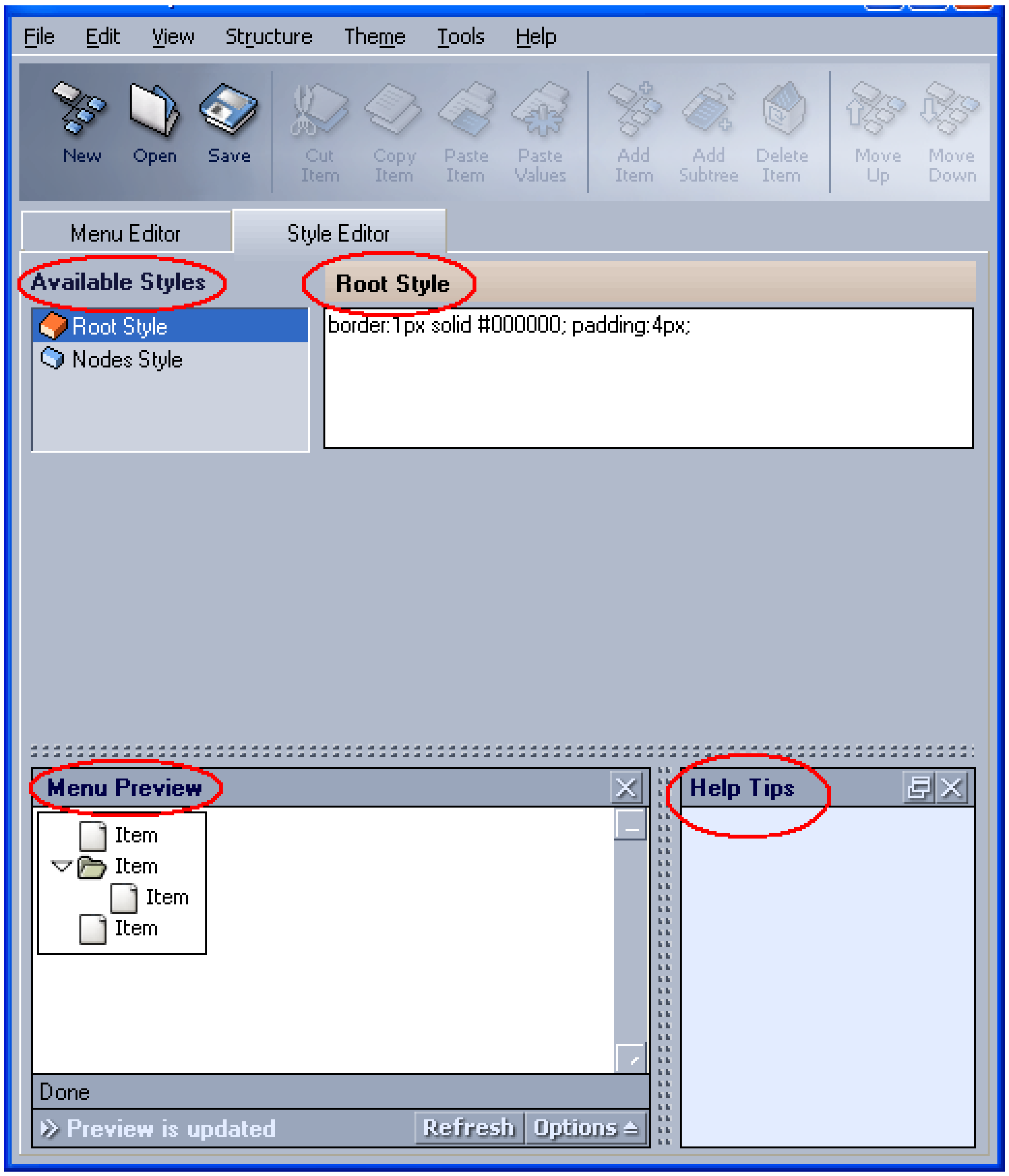
The Available Styles Pane
This is where the available styles can be selected for further editing.

The Properties Pane
This is where the properties of the selected style are edited. The Pane changes its content according to the type of the selected style (Root, Nodes).

The Help Tips Pane

This is where a short description is displayed to provide quick help on the selected property.

The Menu Preview Pane
This is where your menu can be previewed in a WYSIWYG IE browser.



Root Style

border:1px solid #000000; padding:4px;

2.2 Keyboard shortcuts

There are several shortcuts defined for quicker access to frequently used functions:

- **F1**
Provides context sensitive help for each property

- **F12**

Opens the Preview Menu window

- **Ctrl + F12**
Opens the Compile Menu window

- **Shift + Ctrl F12**
Opens the "Link Compiled Menu to Web Pages" window

- **Ctrl + N**
Creates a new menu tree project

- **Ctrl + O**
Opens an .awt menu tree project

- **Ctrl + S**
Saves the current menu tree project

- **Ctrl + X**
Cuts the selected element

- **Ctrl + C**
Copies the selected element

- **Ctrl + V**
Pastes the selected element

- **Ctrl + A**
Pastes the values of the selected element only

Commands

Part III

3 Commands

3.1 Command: File

New
Creates a new menu tree project.
The new menu consists of three items and all properties are set to the default style.

Open
Opens a menu tree project.
The menu tree files are found under the **.awt** extension.

Save
Saves the current menu tree project into an ***.awt** file.

Save As
Saves the current menu tree project with a different filename.

Exit
Closes the application.

3.2 Command: Edit

Cut
Copies the selected item (node) and then deletes it from the menu structure.

Copy
Copies the selected item (node) to the clipboard.

Paste
Inserts the copied item (node) from the clipboard into the menu structure.

Paste Values
Lets you paste the values of the copied item (node) to the selected item (does not create a new item in the menu structure). You can always select a subset of the copied values to paste to your target items.

3.3 Command: View

This command lets you hide or show certain panels that may be useful but not required for the use of the program.

While we do recommend that you keep these panels visible you may hide them if your monitor space is limited.

The panels that can be hidden are the following:

- Help Tips panel
- Menu Editor Preview panel
- Style Editor Preview panel

3.4 Structure

3.4.1 Command: Add Item

Adds a new item in the menu structure.

The new item can always be placed somewhere else by using the **Move Item Up/Down** or the **Cut/Paste** functions.

See also
Delete Item

3.4.2 Command: Add Item Child

Adds a new item as a **child** to the selected item.

See also
Delete Item

3.4.3 Command: Move Item Up

Moves the selected item up by one position.

This action will not move the item outside its parent item. This can be easily done by using the Cut and Paste functions instead.

See also
Move Item Down

3.4.4 Command: Move Item Down

Moves the selected item down by one position.

This action will not move the item outside its parent item. This can be easily done by using the Cut and Paste functions instead.

See also
Move Item Up

3.4.5 Command: Delete Item

Deletes the selected item from the menu structure.

If the selected item has any child items, the child items are deleted as well.

See also
Add Item

3.5 Theme

3.5.1 Command: Apply Theme to Tree

Opens the **Theme Gallery** to choose from a collection of predefined themes and apply it to your tree.

This action replaces all styles used in your menu with the styles contained in the theme.

Therefore, your menu changes appearance but it does not lose its content information.

See also
Export New Theme

3.5.2 Command: Export New Theme

Creates a new theme based on the appearance of your available styles found in the Style Editor.

You can now view and apply your new theme to existing menus by using the Apply Theme to Menu command.

See also
Apply Theme to Menu

3.6 Tools

3.6.1 Command: Refresh Tree Preview

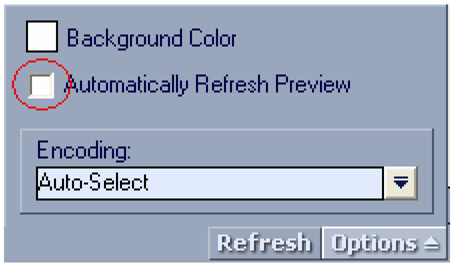
Displays the tree design on a WYSIWYG IE browser pane.

This is a fast way to view the changes made during the tree design phase. It is a **WYSIWYG** window that will help you experiment with your tree properties, observe its behavior and implement its design exactly the way you like.

If the preview window is open and a change is made at the properties or in the menu structure, the preview window signals its non-updated status by the bottom message colored in red.

To show the latest changes in your menu design, press F12 or click the icon in the application toolbar or the Refresh button in the Menu Preview window. The preview will then be updated.

Note that you can have your preview automatically updated every time you make a change to the menu by checking the related check box at the Options



Just have in mind that this is not recommended for very large trees as the automatic refresh may slow down the application response to your actions.

3.6.2 Command: Compile Tree

Compiles the .awt file into a .js file that has to be linked to your web pages.

To use a tree menu in your web pages you must take the following four steps:
Step 1. **Compile** the tree menu (note: spaces are not allowed in filename)
Step 2. **Link** the compiled tree to your web pages using the related application command
Step 3. **Upload** the necessary files and folders (see a description of these elements below)
Step 4. **Upload** the web pages that contain the tree link

How it works:

When compiling a tree menu as ***tree.js***, the following elements are created inside the same directory:

- The **tree.js** file that contains the tree contents.
- The domain-specific **ldmt.js** file, which is the javascript engine (library) that builds the tree on the page for the chosen domain (along with localhost).

The **images** subfolder (or your custom directory as defined in Project Properties > Compile Properties > Folders) which contains the images used in the tree.

Example
Let's say that the local root folder of your site is the **c:\mysite** folder and that the contents of your web site are:

```
| ------------------------------------------- |
| c:\mysite\index.html             |
| c:\mysite\companyinfo.html    |
|          ...                     |
| c:\mysite\faq.html               |
| --------------------------------------------|
```

You now design a tree, called **tree.awt**, and you compile it into a file called **c:\mysite\tree.js**.

Assuming that you used the **gnome** theme, your local web site now contains:
| -- |
| c:\mysite\index.html |
| c:\mysite\companyinfo.html |
| ... |
| c:\mysite\faq.html |
| **c:\mysite\tree.js** |
| **c:\mysite\ldmt.js** |
c:\mysite\images\gnome\*.*

The elements in **bold** should be **uploaded** on the server in the same directory structure where they are locally created.

If you don't upload them or you misplace them on the server, the tree will produce a run-time error and will fail to appear.

Notes:

You can compile the tree anywhere you like **within** your site's directory structure. This decision should not be affected by the location of the pages you want to link the tree to.

For example, if you want to link the tree into a Dreamweaver template, you are **not** required to compile the tree into the Templates directory. On the contrary, we do not advise you to do so as you will have to upload the Templates directory on the server too.
In general, simply compile your tree into your **root** directory or into a manually-created subdirectory and you will be able to link it to any page, shared border, template or library of your web site directory structure.

See also
Link Compiled Tree to Web Page(s)
Compile Properties: Folders
Compile Properties: Domain

3.6.3 Command: API: Extract ldmt.js

How to create and access a programmable tree using the API

Step 1: Use the application to **extract** the library file (**ldmt.js**), which is the engine that generates the trees on-the-fly.

To extract the library, go to **"Tools -> API: Extract ldmt.js**" menu option of the application. Choose a folder where your page containing the tree also resides, as you will later insert a link into this page for the ldmt.js file.

The extracted library will be licensed (and functional) for the particular domain specified at your "Project Properties". Note that for testing purposes your trees will always be functional in your local environment (localhost), regardless of the domain chosen.

Step 2: Edit the page where you want the programmable tree to appear, insert a link to the ldmt.js file, and start programming the tree based on the information found at:

- The API Reference contained herein.

- The sample pages located at the subfolder **"API Examples"** of your installation folder. The sample pages are the ones that start with the word "API".

- The examples contained herein at the "Examples" chapter. View first example.

3.6.4 Command: Project Properties

Allows you to specify the following properties and options for your loaded menu project:

Preview Panel Settings

Project Info Settings

Compile Properties

Folders

Domain

Advanced Scripting

3.6.4.1 Compile Properties

To read more info on a "Compile Properties" topic, please click on its respective link below:

Folders

Domain

Advanced Scripting

3.6.4.1.1 Compile Properties: Folders

Here you specify the output folders and files to be created when compiling your menu tree project.

The **Site_Root folder** is the location where your web site (your main index page) is locally stored. It is also referred as the "Site Root" folder in other applications. It is very important that you specify the correct folder here, as it is used in both the compile process and in the <Link> item property for browsing the site pages.

The **Compiled tree file name** is the name of the javascript file that is generated when you compile the tree. It defaults to *tree.js* but you may as well change it to something else. Note that you can use multiple trees in the same website by having different projects compiling trees under different "Compiled tree file name" properties.

Once you setup your "Site_Root folder", you are able to edit the "**Compiled subfolders**" which include the following:

Tree file and library (ldmt.js) subfolder

Here you can specify a custom subfolder in your web site where your tree will be compiled. The default folder where the tree will be compiled is the Site Root of your web site. Note that a subfolder needs to exist prior to selecting it. When compiling your tree, a file called **ldmt.js** is also created under the same directory. This file is the **"library"** file that contains the code for generating the tree on the page.

Tree images subfolder

Here you can specify the subfolder where all images used by the tree are generated. If not uploaded, the tree will still display all items but without images (the missing image sign ("**x**") may also appear in place of those images).

Note: Compiled subfolders may also contain **relative** paths (eg: ../). If you do so, the subfolder location will be relative to the location of the compiled **tree.js** file.

3.6.4.1.2 Compile Properties: Domain

Here you choose the **domain** where the compiled tree will be used. Initially you can only create trees for **localhost** (i.e. for the local environment). You need to have purchased licenses for online domains and apply the related Activation Code (found in your Likno User Area account) so that these purchased domains appear in the list.

Examples of domains:

http://www.likno.com

https://www.mysite.com

3.6.4.1.3 Compile Properties: Advanced Scripting

In this Compile Property you can specify additional javascript code to be inserted into the tree.js file upon compilation of the tree.

This additional code will be executed every time the tree appears on page (i.e. every time the tree srcipt is running).

An experienced developer can integrate several javascript effects inside the tree.js file. You can imagine that this extends the functionality dramatically, since the tree.js may be running in all the pages of your web site.

Your scripts may produce javascript errors in the Preview pane if they rely on information located at your actual web pages. In this case it is recommended that you check the "Ignore script in Preview pane" so that they do not run at preview mode. The scripts ***do*** run on your actual web pages.

Note 1: Do not use the <script> & </script> tags. The tree.js is already a javascript file.

Note 2: This feature is for advanced users only! You may produce Javascript Errors if you do not include valid code here.

3.6.4.2 Preview Panel Settings

You can specify options for the Preview Panel, such as the Background Color, the text Encoding and if the menu is automatically refreshed when a change is made to a property.

Note: The "Automatically Refresh Preview" is not recommended if you have very large trees as it may slow you down when editing the tree properties.

3.6.4.3 Project Info Settings

Here you can specify settings for your tree project, such as:

Author Name
Author URL
Project Description
Design Notes

3.6.5 Command: Link Compiled Tree to Web Pages

Links a compiled tree (.js file) to a selection of web pages.

At this stage, the application inserts the necessary tree link into the selected web pages. No DHTML or Javascript knowledge is required since the whole process is conducted by the application. The tree link is a few lines of code and can be easily identified inside the source code by a related comment.

The linking process requires only two pieces of information:
1. The **compiled tree file** (eg: tree.js)
2. The **web page(s)** where the tree link will be created.

Obviously, you need to **upload** the updated web pages on the server so that the tree can be viewed on-line. However, it is important that you also upload...
1. the compiled tree file (eg: ***tree.js***).
2. the tree library **ldmt.js**
3. the **images** folder

...at the **same** directory structure where they reside locally.
See the Compile Tree command for more info on this issue.

To update the design of an already compiled tree you should first make the changes to its respective **.awt** file and then recompile it. Provided that you did not change the name or the directory of the compiled tree, you do not need to re-link the tree to the pages that already contain that link.

How does the tree-linking process work:
Let's say that you want to link a menu tree called "tree3.js" to a page. The process asks the following question and takes the respective action:

Does tree-linking code already exist for "tree3.js"?
Yes **->** **Replace** the old tree-linking code with the new one
No **->** Insert a **new** tree-linking code for "tree3.js" into the page

Notes:

Note about file extensions:
The link can be inserted in any kind of page, regardless of its extension. Just use the All Files (*.*) filter to select web pages that have an extension other than the typical .html or .htm one.

Note for Dreamweaver users:
If you like, you can also link the compiled menu to a template (.dwt) or a library (.lbi). Just select the related directory (Templates or Library) and use the All Files (*.*) filter so that you can view the template or library file.
The tree **does not have** to be compiled inside the Templates directory. On the contrary, we do not advise you to do so as you will have to upload the Templates directory on the server too.

Note for FrontPage users:
Similarly to Dreamweaver, you can link your menu into an **include** so that it can be used by all other pages.

See also
Compile Tree

3.6.6 Command: Check for Updates

Checks if newer versions have been released since the last time you installed the Likno Drop-Down Menu Trees application.

It is recommended that you check for updates frequently as we regularly release improved cross-browser support for your tree implementations.

3.6.7 Command: Customize

Allows you to customize AllWebMenus on the following settings:

General

Defaults

Preview Panel Settings

Project Info Settings

3.6.7.1 General

Here you can customize the general look and feel of your interface.

Show text on toolbar buttons

You can choose not to show text on the toolbar buttons (default behavior) so that you save some space for your design panels.

3.6.7.2 Defaults

Allows you to customize your AllWebMenus preferences on the following properties and options:

Preview Panel Settings

Project Info Settings

Note that the above settings refer to the initial values (defaults) that new tree projects should start with. If you want to update the values of your loaded project you should edit its Project Properties instead.

3.6.7.2.1 Preview Panel Settings

Here you can specify **default** settings for **new** trees concerning their Preview Panel.

For more info on these settings, please refer to the related project properties.

3.6.7.2.2 Project Info Settings

Here you can specify **default** settings for **new** trees concerning their Project Info.

For more info on these settings, please refer to the related project properties.

3.7 Command: Help

Contents
Displays the contents of the help file. It also supports both keyword and full-text search.

How to Register
Displays information about the registration process.

Enter Activation Code
This is where you enter your Activation Code. Your Activation Code activates the domains that you have purchased so that you can compile your trees for those domains.

Visit Likno on the Web
Visit our web site to contact us and get the latest information on our products.

About
Displays the About box.

Part IV

4 Properties

4.1 Properties Overview

Properties are classified according to the **element** (the whole tree or a specific item) they refer to:

- Tree Properties
- Item Properties

4.2 Properties Description

4.2.1 Tree Properties

Tree Properties deal with the behavior and appearance of the whole tree.

4.2.1.1 Positioning ID

You will use this ID inside your HTML pages to mark where the tree appears.

The tree will appear attached to the HTML element (a DIV) that uses this specified ID.

You may leave this value to default ("treeid"). You only need to change this value if you use more than one tree in your page OR there is another HTML element that uses the same value as ID.

When linking your tree to your pages, apart from the **linking code** the following code is also added to the HTML:

```
<div id="treeid" style="position:absolute; width:100%;"></div>
```

The above code is the **positioning code** which marks where the tree with <Positioning ID> = "treeid" will appear.

4.2.1.2 Images Path

The path/directory that contains the icons used by the tree.

This is how you can access these images in your hard drive:

- Open the folder where Likno Drop Down Menus is installed...
  (for example: **C:\Program Files\Likno Drop-Down Menu Trees**)

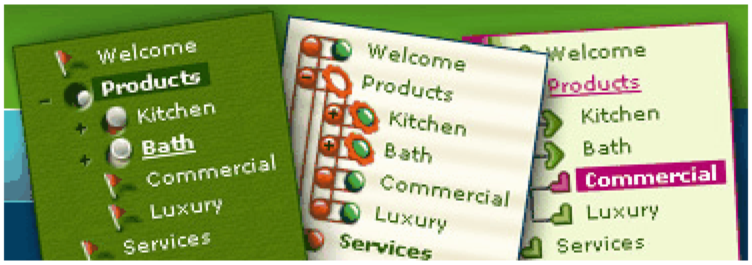
- ...then open the following subfolder:
  ...**Themes\images\ldmt\16x16\**

- There you will find subfolders which contain a set of icons (images) that are used by the tree. For example, the subfolder "gnome" contains images that can be used for a gnome-like tree.

- You can create your own subfolder (for example: "**\myicons**") and add a set of your own images there. Note that the icons must have specific filenames (shown on table below) and they all have to be of the same image size (width & height) to be used by the tree. In most cases your images must be 16x16 pixels, which is the standard size of icons used for tree controls.

- Now that you have created your custom "Images Path" subfolder (e.g. **myicons**), you only need to select it from the drop-down box of the property <Images Path>.

Note: You can access many more interesting images by **purchasing/downloading additional Themes** through the following URL:



http://www.likno.com/drop-down-menu-trees/gallery.php

Naming specification table:

icon	filename	description
	empty.gif	Just an empty 16x16 image
📄	leaf.gif	Default icon for all nodes with no children
📄	leaf_mo.gif	Default icon for all nodes with no children (mouse over state)
📄	leaf_s.gif	Default icon for all nodes with no children (selected state)
┌	line_es.gif	Connecting line (East-South)
└	line_ne.gif	Connecting line (North-East)
├	line_nes.gif	Connecting line (North-East-South)
│	line_ns.gif	Connecting line (North-South)
⊟	minus_e.gif	Collapse icon (East)
⊟	minus_es.gif	Collapse icon (East-South)
⊟	minus_ne.gif	Collapse icon (North-East)
⊟	minus_nes.gif	Collapse icon (North-East-South)
⊞	plus_e.gif	Expand icon (East)
⊞	plus_es.gif	Expand icon (East-South)
⊞	plus_ne.gif	Expand icon (North-East)
⊞	plus_nes.gif	Expand icon (North-East-South)
📁	subtree_closed.	Default branch collapsed icon

	gif	
📁	subtree_closed_mo.gif	Default branch collapsed icon (mouse over state)
📁	subtree_closed_s.gif	Default branch collapsed icon (selected state)
📂	subtree_open.gif	Default branch expanded icon
📂	subtree_open_mo.gif	Default branch expanded icon (mouse over state)
📂	subtree_open_s.gif	Default branch expanded icon (selected state)

Note: You may use either the .gif or .png format, as long as you do not change the required specific names of the files (for example: "**plus_e**.png" or "**plus_e**.gif") and all images use the same format within the subfolder.

4.2.1.3 Leaf Height

Height of each leaf on the tree.

4.2.1.4 Box Height

Height of the box of the tree to be used (if 0, the tree will expand dynamically to the bottom), in pixels.

4.2.1.5 Box Width

Width of the box of the tree to be used (if 0, the tree will expand dynamically to the right), in pixels.

4.2.1.6 Offset X

Horizontal distance from the top left corner of the HTML element to which this tree is attached (in pixels).

4.2.1.7 Offset Y

Vertical distance from the top left corner of the HTML element to which this tree is attached (in pixels).

4.2.1.8 Menu Effects

Indicates whether this tree will use animated branch collapse/expand effects (visible only in Microsoft Internet Explorer).

4.2.1.9 Horizontal Scrollbar

Indicates whether horizontal scrollbars should be used if some node exceeds the width of the tree box.

4.2.1.10 Click to Expand

Indicates whether a single click on a branch will expand it.
By default, branches are expanded with a double click.

4.2.1.11 Allow Folder Selection

Indicates whether the user can select folders (otherwise only leaves will be selectable).

4.2.1.12 Auto Collapse

Indicates whether clicking on a branch at the same level will collapse all other expanded branches at the same level (in the standard Microsoft Windows XP Explorer fashion).

4.2.1.13 Allow Multiple Selections

Indicates whether this tree allows multiple selections.
A tree which allows multiple selections can be used as an options control with check boxes. When the user clicks on a node, the previously selected node is not cleared.

Note: Using the API, you can use the `selection` property to get an array of all the selected nodes.

4.2.1.13.1 Use [Ctrl] key

Indicates whether the Ctrl key should be used for multiple selections.
If true, the user must keep the Ctrl key pressed while clicking on a node to add it to the selection (in the standard Microsoft Windows Explorer fashion). If the user clicks on a node without pressing the Ctrl key, the previously selected nodes will be cleared.

4.2.1.13.2 Toggle Selected

Indicates whether clicking again on a selected node de-selects it.

4.2.1.14 Remember Tree State

Saves and remembers the tree state.

Select this property if you want the tree to keep its "state" when the visitor browses the website from page to page. The tree state is stored in a cookie (the visitor needs to allow cookies in her browser) and remembers which branches are open or closed.

4.2.2 Item Properties

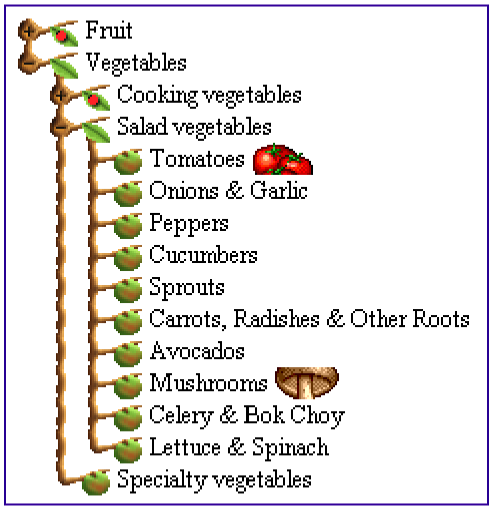
Item Properties deal with the behavior and appearance of an individual item.

*Item properties do **not** change when a new Theme is applied.*

4.2.2.1 Content

Textual or HTML content of the node. This is what is displayed next to the icon of the node.

In the tree below, the content of the "Fruit" node is simply "`Fruit`" while the content of the "Tomatoes" node is "`Tomatoes <img src='images/tomatoes.gif' style='border:0; vertical-align:top;'>`"



4.2.2.2 Link

Specifies the URL which will load when the user clicks on this node. Can be relative to the page hosting the tree control (e.g. "examples/example1.htm") or absolute (e.g. "http://www.likno.com).

4.2.2.3 Link Target

Specifies the frame or window where the page specified by the <Link> property opens.

Values:
Leave it blank (none) to open the <Link> into the same frame or window where the tree is located (**default** for non-framed pages).

Choose **_blank** to open the <Link> into a new browser window.

Choose **_top** to open the <Link> in the whole area of the current browser window (not to a specific frame).

Choose the **frame name** where you want the <Link> to open in.

Example with Frames:
Let's assume that we have the following Frameset Definition inside the source code of the home page:

```
<frameset framespacing="0" border="0" cols="171,*" frameborder="0">
  <frame name="left" target="main" src="left_page.htm" scrolling="auto">
  <frame name="main" src="main_page.htm" target="_self">
  <noframes>
  <body>
  <p>You need a browser that supports frames to see this site </p>
  </body>
  </noframes>
</frameset>
```

Let's also assume that we have placed our tree at the left_page.htm page (opening at the **"left"** frame). We want the tree items to open their pages specified by their <Link> property inside the **main** frame.

As we can see from the <frameset> tag, the frame where the main_page.htm opens has been assigned the following name:

main

This is exactly what you need to type into the **<Link Target> property** of the menu items. Do not get confused and use the content of the **src** attribute (eg: main_page); this will not work. You need to use the content of the **name** attribute (eg: main).

A wizard may also help you choose the correct value by pressing the [...] button next to the field.

Note! If you use the **frame** option, the frame name will have no effect inside the Menu Preview window, so that it does not produce any errors related to a frame name that does not exist. The <Link Target> property **does** work when your tree is compiled and linked to your actual pages.

4.2.2.4 Status Bar Text

Text to show on the browser status bar whenever user moves mouse over this node. *

*Note: Some browsers disable changing the status bar text in script by default.

4.2.2.5 Tooltip

A tooltip to be shown whenever the user moves the mouse over this node.

4.2.2.6 Javascript Command

A Javascript function to be called whenever this node is clicked.

The Javascript Command will be executed when the user **clicks on** the item's area. This does not cancel the *<Link>* property. You can set both a <Link> to open **and** a Javascript command to be executed when clicking on the Item.

You can use more than one javascript commands, provided that you separate them with a semicolon (;).

Notes! You may use only quotes (") or only apostrophes (') inside the text of the property but **not both**! Using both quotes and apostrophes will cause a javascript error.

Javascript Commands are deactivated when previewing your tree inside the Preview pane so that they do not produce any javascript errors. Javascript Commands ***do*** run on your actual web pages.

FAQ and Tips

Part

V

5 FAQ and Tips

5.1 Troubleshooting

This is a collection of quick tips to avoid trouble.

Compile Tree - Link Tree process:

- It is critical that you do not forget to upload the **ldmt.js** file along with the ***tree*.js** file after you compile your tree.

- You can compile the tree anywhere you like **within** your site's directory structure. This decision should not be affected by the location of the pages you want to link the tree to.
  For example, if you want to link the tree into a Dreamweaver template, you are **not** required to compile the tree into the Templates directory. On the contrary, we do not advise you to do so as you will have to upload the Templates directory on the server too.
  In general, simply compile your tree into your root directory or into a manually-created subdirectory and you will be able to link it to any page, shared border, template or library of that web site directory structure you want.

- If you use **FrontPage**, the tree should not be compiled inside the *_borders* folder or any other special folders used by the authoring tool to manipulate common objects. The obvious reason is that these folders are not uploaded on the server when publishing the site. Instead, compile the tree in the root or in any other manually-created subdirectory.

Font selection:

For cross-platform compatibility, it is advisable to use the generic font families instead of any system-specific font. For example, if you use "Arial" and a specific platform does not have the Arial font, then the tree will use the browser's **default** font. But if you use "Sans Serif" instead of "Arial", the browser will use the appropriate Sans Serif font (in most cases it will be "Arial", but in platforms that do not have Arial it will be a **similar** font; not the default font)

Filenames:

- It is known that **Netscape cannot read images that contain spaces in their filenames**! Therefore, always use "my_image.gif" or "myimage.gif" instead of "my image.gif".

- **Spaces in the compiled tree name are not allowed** as they are not parsed properly by some browsers, such as Netscape. As an example, use "main_tree.js" or "maintree.js" instead of "main tree.js".

- Avoid using filenames that contain special characters, like quotes ("), apostrophes ('), etc. In most cases these characters will produce an error.

Note on <*Javascript Command*> syntax:

You may use only quotes (") or only apostrophes (') inside the text of the property but **not both**! Using both quotes and apostrophes will cause a javascript error.

Note on <*Link*> syntax:

Always use **slashes** instead of backslashes
(example: ../users/~user10/user10.html)

5.2 Frequently Asked Questions

My tree works fine when viewed locally but does not appear when viewed on the server. What have I done wrong?
You have forgotten to upload the tree-building library (**ldmt.js** file) along with the compiled tree (eg: **tree.js**) or you have uploaded them in the wrong directory.
Click here to read more about this procedure
(Note: Make sure that you have selected the Domain to compile for.)

Can I use HTML tags inside the Content property?
Yes, you can use any HTML tag you like inside the <Content> property.

Can a tree item link to a paragraph within a web page?
Yes, just enter **http://www.mysite.com/info.html#anchor** in the *<Link>* property of the items and the tree can be used to navigate inside the **info.html** web page.

Can I have a tree item preselected on page load?
Yes, you can use the selectBranch() function in your HTML pages to achieve this.

API: Create and manage your trees programmatically

Part

VI

6 API: Create and manage your trees programmatically

6.1 Introduction

Apart from creating trees using the application, you also have the option of creating your trees programmatically through the documented API. This option requires some programming skills, but gives you more flexibility as it includes functions for manipulating all aspects of the trees, sorting, searching for nodes, dynamically inserting or deleting nodes, etc.

Using the API you have the following capabilities:

- Generate your trees from a database and associate an array of values (e.g. from a recordset) with each leaf.
- Add or Delete items on-the-fly based on conditions or events.
- Allow your visitors to change or rearrange your tree dynamically on their browser.
- Sort the items under any branch based on conditions.
- Define different styles for different groups of items and for three different states: normal, mouse over and selected.
- Define different icons for each of the states mentioned above, and for each node, closed or open branch.
- Any number of CSS classes can be dynamically added and used or changed for different nodes or branches of the tree.
- Optional tree state persistence across sessions. The state of the tree is saved in a cookie.
- Ability to associate custom user-defined data with each node (see the Stock market portfolio and Shopping cart management examples)

...and many more.

How to create and access a programmable tree using the API

Step 1: Use the application to **extract** the library file (**ldmt.js**), which is the engine that generates the trees on-the-fly.

To extract the library, go to **"Tools -> API: Extract ldmt.js**" menu option of the application. Choose a folder where your page containing the tree also resides, as you will later insert a link into this page for the ldmt.js file.

The extracted library will be licensed (and functional) for the particular domain specified at your "Project Properties". Note that for testing purposes your trees will always be functional in your local environment (localhost), regardless of the domain chosen.

Step 2: Edit the page where you want the programmable tree to appear, insert a link to the ldmt.js file, and start programming the tree based on the information found at:

- The API Reference contained herein.
- The sample pages located at the subfolder **"API Examples"** of your installation folder. The sample pages are the ones that start with the word "API".
- The examples contained herein at the "Examples" chapter. View first example.

6.2 API reference

6.2.1 Overview

The Likno Drop Down Menu Trees API consists of two objects, `LiknoTreeElement` and `LiknoTreeRoot`.
`LiknoTreeElement` represents a generic node in the tree, while `LiknoTreeRoot` represents a complete tree, is derived from `LiknoTreeElement` and provides an additional set of properties and functions.

6.2.2 The LiknoTrees array

Whenever you create a new LiknoTreeRoot object, an entry is added to the global array `LiknoTrees`. You can then have access to all the trees in your webpage using this array in your Javascript code.
See the Stock market portfolio and the Shopping cart management examples.

6.2.3 Objects

6.2.3.1 LiknoTreeElement

6.2.3.1.1 LiknoTreeElement()

You do not need to construct a LiknoTreeElement object directly in code. Instead, once you have a LiknoTreeRoot object, you use the addLeafWithParams and insertLeafWithParamsAfter functions to create and add nodes to your tree.

6.2.3.1.2 Properties

All properties can be set/get using the appropriate function, e.g. you can set the `isSelectable` property of the `LiknoTreeElement` using the `setIsSelectable()` function and you can get the current value of this property for a leaf using the `getIsSelectable()` function.
Although Javascript is not a strongly typed language, a generic property type is indicated under every property name.
If the property type is an array, a number in brackets next to it will indicate the required number of elements in the array (e.g. array(3) means that the property is required to be an array of exactly 3 elements). An ellipsis ("...") means that the array can have a variable number of elements.

6.2.3.1.2.1 classnames

This is an array of 3 strings specifying the suffix of the CSS class name used for the normal, mouseover and selected state of this node.

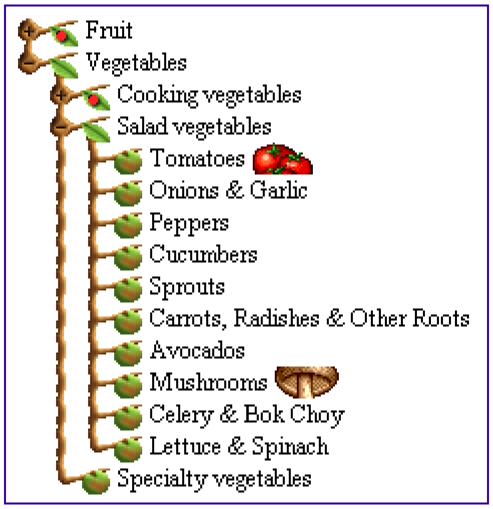
Example:

```
LiknoTrees[0].addClass("default", "font-family:Tahoma; font-size:11px; color:#FFFFFF;
background-color:; text-decoration:none; font-weight:normal; padding-left:4px;
padding-right:4px;");
LiknoTrees[0].addClass("mouseover", "font-family:Tahoma; font-size:11px; font-weight:
normal; text-decoration:underline; color:#d9f6ab; padding-left:4px; padding-right:4px;");
LiknoTrees[0].addClass("selected", "font-family:Tahoma; font-size:11px; font-weight: bold;
color:#b1efff; background-color:; padding-left:4px; padding-right:4px;");
var leaf = LiknoTrees[0].findLeafByText("examples", false, false, true);
leaf.setClassnames("default", "mouseover", "selected");
```

6.2.3.1.2.2 content

Textual or HTML content of the node. This is what is displayed next to the icon of the node.
Although you can use the `setContent` function to change the content property for an item at runtime, most of the time you will set the content of a node using the addLeafWithParams function.

In the tree below, the content of the "Fruit" node is simply "`Fruit`" while the content of the "Tomatoes" node is "`Tomatoes <img src='images/tomatoes.gif' style='border:0; vertical-align:top;'>`"



6.2.3.1.2.3 data

An array of optional user-defined values which can be associated with this node. Each node can hold an arbitrary number of values in this array, and these values can be set/retrieved and used for sorting the nodes of the tree. For example, you could have a tree showing Wall Street market stock names, and have their associated values stored internally in the tree using this property.
See the Stock market portfolio example.

6.2.3.1.2.4 depth

Indicates the depth of this node, starting from 0 for the root of the tree.

6.2.3.1.2.5 iconFormat

This is the image format of the icons used by this node and can be either of two string values, "gif" or "png". You do not need to set this property manually for each node, unless you want to use a custom icon in a node using a different format than the format specified initially by the tree.

6.2.3.1.2.6 iid

Unique internal id of node, is calculated automatically using the tree id and should not be set or changed manually.
The format of the iid is: TreeName_ItemNo[_SubItemNo]*

The following example assumes that the tree is named "myTree".

Tree Structure	**iid**
Item1	myTree_0
SubItem1	myTree_0_0
SubItem2	myTree_0_1
SubSubItem1	myTree_0_1_0
SubSubItem2	myTree_0_1_1
SubItem3	myTree_0_2
Item2	myTree_1

Item3	myTree_2

6.2.3.1.2.7 isExpanded

Indicates whether this node (folder) is expanded (assuming this node has children).

6.2.3.1.2.8 isMouseOver

Indicates whether the mouse is over this node.

6.2.3.1.2.9 isSelectable

Indicates whether this node can be selected (clicked) or not.

6.2.3.1.2.10 isSelected

Indicates whether this node is currently selected.

6.2.3.1.2.11 jscriptCommand

A Javascript function to be called whenever this node is clicked.
See the Shopping cart management example.

6.2.3.1.2.12 leafIcon

An array of up to 3 strings specifying the filenames of the icons for the normal, mouseover and selected states of this node respectively. These icons will be used if the node has no children.

6.2.3.1.2.13 leaves

An array of all the children nodes of this node (if any).

6.2.3.1.2.14 link

Specifies the URL which will load when the user clicks on this node. Can be relative to the page hosting the tree control (e.g. "examples/example1.htm") or absolute (e.g. "http://www.likno.com).

6.2.3.1.2.15 nextLeaf

Points to the next sibling of this node, or null if this is the last child.

6.2.3.1.2.16 parent

Points to the parent node of this child.

6.2.3.1.2.17 prevLeaf

Points to the previous sibling of this node, or null if this is the first child.

6.2.3.1.2.18 statusbarText

Text to show on the browser status bar whenever user moves mouse over this node. *
*Note: Some browsers disable changing the status bar text in script by default.

6.2.3.1.2.19 subtreeClosedIcon

An array of up to 3 strings specifying the filenames of the closed subtree icons for the normal, mouseover and selected states respectively. These icons will be used if the node has children.

6.2.3.1.2.20 subtreeOpenIcon

An array of up to 3 strings specifying the filenames of the open subtree icons for the normal, mouseover and selected states respectively. These icons will be used if the node has children.

6.2.3.1.2.21 target

Specifies the name of the frame in which the URL specified by this node's link property will be loaded.

6.2.3.1.2.22 tooltip

A tooltip to be shown whenever the user moves the mouse over this node.
See the Stock market portfolio and the Shopping cart management examples.

6.2.3.1.2.23 vsbl

Indicates whether this node is visible or not. The leaves of a closed folder have this set to false.

6.2.3.1.3 Functions

These are the functions supported by the LiknoTreeElement object.
Note that all these functions are available to the LiknoTreeRoot object too, since it is inherited from LiknoTreeElement.
The return type of each function, if any, is indicated underneath each function name.
If the returned variable is restricted to specific values, those will be indicated in brackets next to the return type.

6.2.3.1.3.1 addLeafWithParams

Parameters

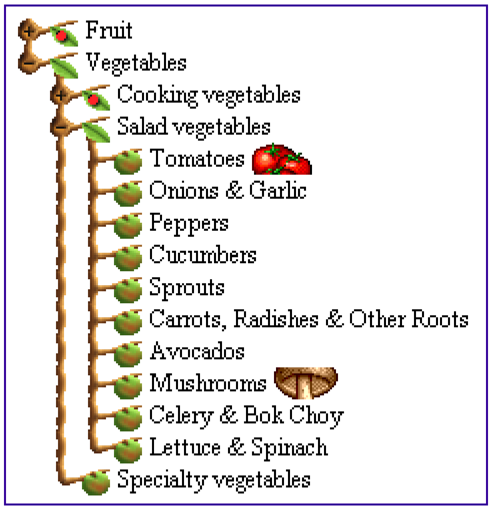
#	name	type	description
1	`content`	string	The content of the node. It can be either plain text or html code including complex formatting and/or images. In the latter case, care must be taken that the resulting html code does not break the tree code.
2	`link`	string	The URL which will load when the user clicks on this node.
3	`target`	string	The name of the frame in which the preceding URL will be loaded.
4	`vsbl`	boolean	Specifies whether this node will be initially visible or not. A parent node with all its children invisible will be shown as collapsed.
5	`icon`	string	The file path to the image file for a custom icon to be used by this node. If not specified, the default icon(s) for this tree will be used.

Adds a new node to the tree, as a child of the node which is calling it, and returns this new node.
The new node will be added at the end of the list of children of its parent.
The following example will build the tree shown below (in the first image, the tree is shown in its initial state, while in the second image, the tree is shown fully expanded):

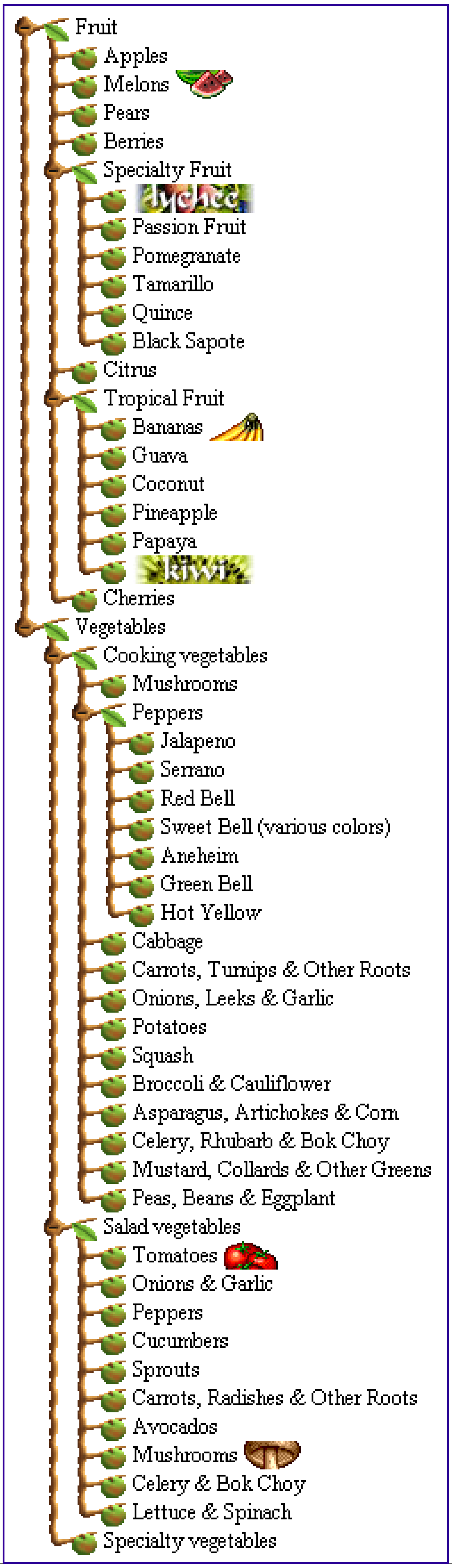
Example:

```
treeD = new LiknoTreeRoot("treeD", "example", 0, 0, 0, 0, 16, true, "background-color:white;
border:1px solid #330099; padding:4px;", "gif");
treeD.setImagesDir("images/ldmt/16x16/tree/");
treeD.addClass("de", "font-family:Times New Roman; font-size:12px; color:black;
background-color:transparent; text-decoration:none; font-weight:normal; padding-left:4px;
padding-right:4px;");
treeD.addClass("mo", "font-family:Times New Roman; font-size:12px;
text-decoration:underline; color:#669900; padding-left:4px; padding-right:4px;");
treeD.addClass("se", "font-family:Times New Roman; font-size:12px; color:white;
background-color:#CC6600; padding-left:4px; padding-right:4px;");
```

```
t2_a = treeD.addLeafWithParams("Fruit", "", "", true, "");
t2_a0 = t2_a.addLeafWithParams("Apples", "", "", false, "");
t2_a1 = t2_a.addLeafWithParams("Melons <img src='images/melons.gif' style='border:0;
vertical-align:top;'>", "", "", false, "");
t2_a2 = t2_a.addLeafWithParams("Pears", "", "", false, "");
t2_a3 = t2_a.addLeafWithParams("Berries", "", "", false, "");
t2_a4 = t2_a.addLeafWithParams("Specialty Fruit", "", "", false, "");
t2_a40 = t2_a4.addLeafWithParams("<img src='images/lychee.jpg' style='border:0;
vertical-align:top;'>", "", "", false, "");
t2_a41 = t2_a4.addLeafWithParams("Passion Fruit", "", "", false, "");
t2_a42 = t2_a4.addLeafWithParams("Pomegranate", "", "", false, "");
t2_a43 = t2_a4.addLeafWithParams("Tamarillo", "", "", false, "");
t2_a44 = t2_a4.addLeafWithParams("Quince", "", "", false, "");
t2_a45 = t2_a4.addLeafWithParams("Black Sapote", "", "", false, "");
t2_a5 = t2_a.addLeafWithParams("Citrus", "", "", false, "");
t2_a6 = t2_a.addLeafWithParams("Tropical Fruit", "", "", false, "");
t2_a60 = t2_a6.addLeafWithParams("Bananas <img src='images/bananas.gif' style='border:0;
vertical-align:top;'>", "", "", false, "");
t2_a61 = t2_a6.addLeafWithParams("Guava", "", "", false, "");
t2_a62 = t2_a6.addLeafWithParams("Coconut", "", "", false, "");
t2_a63 = t2_a6.addLeafWithParams("Pineapple", "", "", false, "");
t2_a64 = t2_a6.addLeafWithParams("Papaya", "", "", false, "");
t2_a65 = t2_a6.addLeafWithParams("<img src='images/kiwi.jpg' style='border:0;
vertical-align:top;'>", "", "", false, "");
t2_a7 = t2_a.addLeafWithParams("Cherries", "", "", false, "");
t2_b = treeD.addLeafWithParams("Vegetables", "", "", true, "");
t2_b0 = t2_b.addLeafWithParams("Cooking vegetables", "", "", true, "");
t2_b1 = t2_b.addLeafWithParams("Salad vegetables", "", "", true, "");
t2_b2 = t2_b.addLeafWithParams("Specialty vegetables", "", "", true, "");
t2_b00 = t2_b0.addLeafWithParams("Mushrooms", "", "", false, "");
t2_b01 = t2_b0.addLeafWithParams("Peppers", "", "", false, "");
t2_b02 = t2_b0.addLeafWithParams("Cabbage", "", "", false, "");
t2_b03 = t2_b0.addLeafWithParams("Carrots, Turnips & Other Roots", "", "", false, "");
t2_b04 = t2_b0.addLeafWithParams("Onions, Leeks & Garlic", "", "", false, "");
t2_b05 = t2_b0.addLeafWithParams("Potatoes", "", "", false, "");
t2_b06 = t2_b0.addLeafWithParams("Squash", "", "", false, "");
t2_b07 = t2_b0.addLeafWithParams("Broccoli & Cauliflower", "", "", false, "");
t2_b08 = t2_b0.addLeafWithParams("Asparagus, Artichokes & Corn", "", "", false, "");
t2_b09 = t2_b0.addLeafWithParams("Celery, Rhubarb & Bok Choy", "", "", false, "");
t2_b10 = t2_b0.addLeafWithParams("Mustard, Collards & Other Greens", "", "", false, "");
t2_b11 = t2_b0.addLeafWithParams("Peas, Beans & Eggplant", "", "", false, "");
t2_b010 = t2_b01.addLeafWithParams("Jalapeno", "", "", false, "");
t2_b011 = t2_b01.addLeafWithParams("Serrano", "", "", false, "");
t2_b012 = t2_b01.addLeafWithParams("Red Bell", "", "", false, "");
t2_b013 = t2_b01.addLeafWithParams("Sweet Bell (various colors)", "", "", false, "");
t2_b014 = t2_b01.addLeafWithParams("Aneheim", "", "", false, "");
t2_b015 = t2_b01.addLeafWithParams("Green Bell", "", "", false, "");
t2_b016 = t2_b01.addLeafWithParams("Hot Yellow", "", "", false, "");
t2_b10 = t2_b1.addLeafWithParams("Tomatoes <img src='images/tomatoes.gif' style='border:0;
vertical-align:top;'>", "", "", true, "");
t2_b11 = t2_b1.addLeafWithParams("Onions & Garlic", "", "", true, "");
t2_b12 = t2_b1.addLeafWithParams("Peppers", "", "", true, "");
t2_b13 = t2_b1.addLeafWithParams("Cucumbers", "", "", true, "");
t2_b14 = t2_b1.addLeafWithParams("Sprouts", "", "", true, "");
t2_b15 = t2_b1.addLeafWithParams("Carrots, Radishes & Other Roots", "", "", true, "");
t2_b16 = t2_b1.addLeafWithParams("Avocados", "", "", true, "");
t2_b17 = t2_b1.addLeafWithParams("Mushrooms <img src='images/mushrooms.gif' style='border:0;
vertical-align:top;'>", "", "", true, "");
t2_b18 = t2_b1.addLeafWithParams("Celery & Bok Choy", "", "", true, "");
t2_b19 = t2_b1.addLeafWithParams("Lettuce & Spinach", "", "", true, "");
treeD.drawTree();
```



- Fruit
- Vegetables
  - Cooking vegetables
  - Salad vegetables
    - Tomatoes
    - Onions & Garlic
    - Peppers
    - Cucumbers
    - Sprouts
    - Carrots, Radishes & Other Roots
    - Avocados
    - Mushrooms
    - Celery & Bok Choy
    - Lettuce & Spinach
  - Specialty vegetables



- Fruit
  - Apples
  - Melons
  - Pears
  - Berries
  - Specialty Fruit
    - lychee
    - Passion Fruit
    - Pomegranate
    - Tamarillo
    - Quince
    - Black Sapote
  - Citrus
  - Tropical Fruit
    - Bananas
    - Guava
    - Coconut
    - Pineapple
    - Papaya
    - kiwi
  - Cherries
- Vegetables
  - Cooking vegetables
    - Mushrooms
    - Peppers
      - Jalapeno
      - Serrano
      - Red Bell
      - Sweet Bell (various colors)
      - Aneheim
      - Green Bell
      - Hot Yellow
    - Cabbage
    - Carrots, Turnips & Other Roots
    - Onions, Leeks & Garlic
    - Potatoes
    - Squash
    - Broccoli & Cauliflower
    - Asparagus, Artichokes & Corn
    - Celery, Rhubarb & Bok Choy
    - Mustard, Collards & Other Greens
    - Peas, Beans & Eggplant
  - Salad vegetables
    - Tomatoes
    - Onions & Garlic
    - Peppers
    - Cucumbers
    - Sprouts
    - Carrots, Radishes & Other Roots
    - Avocados
    - Mushrooms
    - Celery & Bok Choy
    - Lettuce & Spinach
  - Specialty vegetables

6.2.3.1.3.2 clearSelection

Clears (resets) the selected state (isSelected property) for this node and its children recursively.

6.2.3.1.3.3 compareTo

Parameters

#	name	type	description
1	leaf	LiknoTreeElement	The node to compare this node to.

If sortOnData is a valid index in the array of data associated with both nodes, compares the element of the data array specified by sortOnData with the corresponding element of `leaf`, and returns -1, 0, or 1 if this node's data is less, equal or greater than the specified node's data.
If sortOnData is invalid or -1, a lexicographical comparison of the contents (`content` property, including any html tags, if present) of both nodes is performed.
The results are inverted if sortAscending is false.

6.2.3.1.3.4 enableFx

Enables or disables the animated effects shown when the user collapses or expands this folder node (in Microsoft Internet Explorer only).

6.2.3.1.3.5 findLeafByText

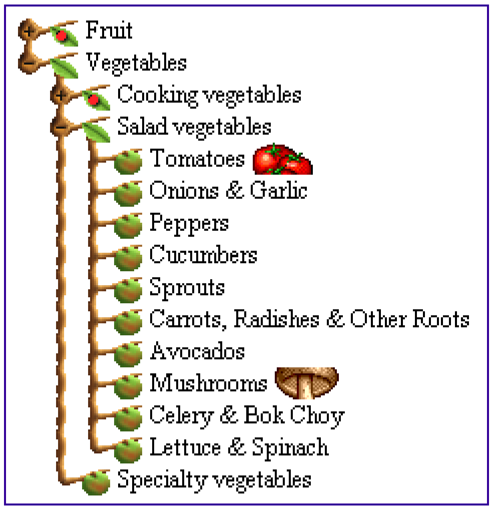
Parameters

#	name	type	description
1	text	string	The text to search for.
2	exactmatch	boolean	Specifies whether an exact match is required (if false, a substring search will be performed).
3	casesensitive	boolean	Specifies whether the search will be case sensitive.
4	vsblonly	boolean	Specifies whether to search only currently visible nodes (if false, nodes under collapsed branches will not be searched).

Returns the first node under this node and its children (recursively) whose contents match the string text, according to the parameters specified.
In the tree below, the following code will select the node "Tomatoes".

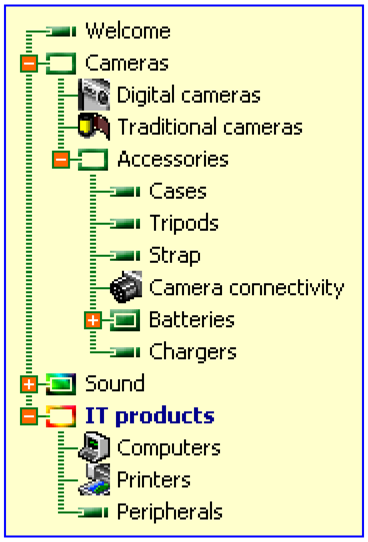
Example:

```
var leaf = LiknoTrees[0].findLeafByText("tomato", false, false, true);
if(leaf)
	LiknoTrees[0].selectBranch(leaf.getIid());
```



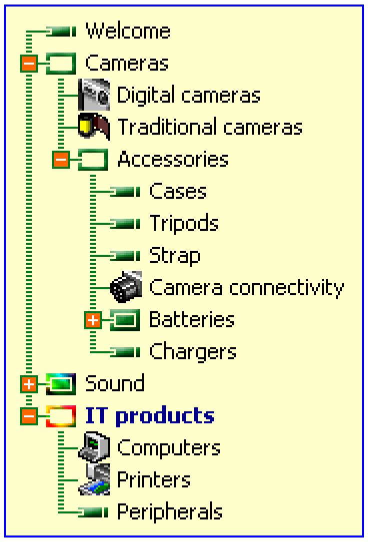
6.2.3.1.3.6 findNextVisibleNode

Returns the next visible node below this one (not necessarily a sibling node).
In the tree below, calling `findNextVisibleNode` on the "Chargers" node will return the "Sound" node.



6.2.3.1.3.7 findPrevVisibleNode

Returns the previous visible node above this one (not necessarily a sibling node).
In the tree below, calling `findPrevVisibleNode` on the "Sound" node will return the "Chargers" node.



6.2.3.1.3.8 getLeaf

Parameters			
#	**name**	**type**	**description**
1	`nbr`	number	The zero-based index of the node to get.

Returns the child leaf indicated by the zero-based `nbr` index (e.g. if nbr = 2, returns the 3rd child of this node, or null if it doesn't exist).

6.2.3.1.3.9 getLeafPos

Returns the zero-based position of this node in the array of siblings.

6.2.3.1.3.10 getRoot

Returns the root node of the tree to which this node belongs.

6.2.3.1.3.11 getVisibleSiblingAbove

Returns the first visible sibling above this node, or null if none.

6.2.3.1.3.12 getVisibleSiblingBelow

Returns the first visible sibling below this node, or null if none.

6.2.3.1.3.13 hasChildren

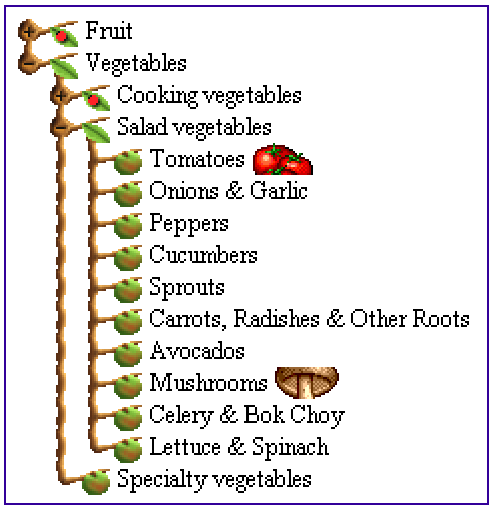
Indicates whether this node has children.

6.2.3.1.3.14 hasLink

Indicates whether this node has a link or not.

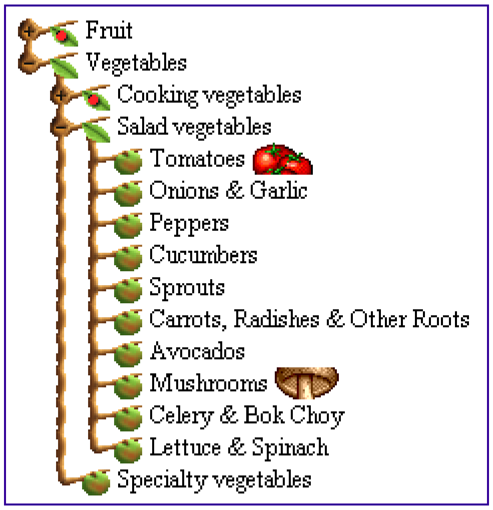
6.2.3.1.3.15 hasSiblingAbove

Indicates whether this node has any siblings above it (i.e. the first child will return false).
In the tree below, the "Specialty vegetables" node has a sibling above it ("Salad vegetables"), while the "Cooking vegetables" node has none.



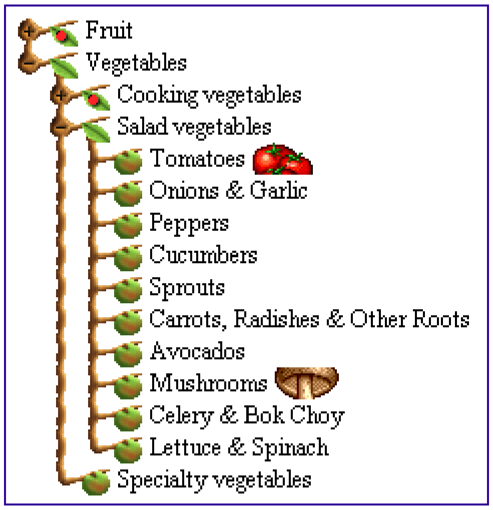
6.2.3.1.3.16 hasSiblingBelow

Indicates whether this node has any siblings below it (i.e. the last child will return false).
In the tree below, the "Salad vegetables" node has a sibling below it ("Specialty vegetables"), while the "Vegetables" node has none.



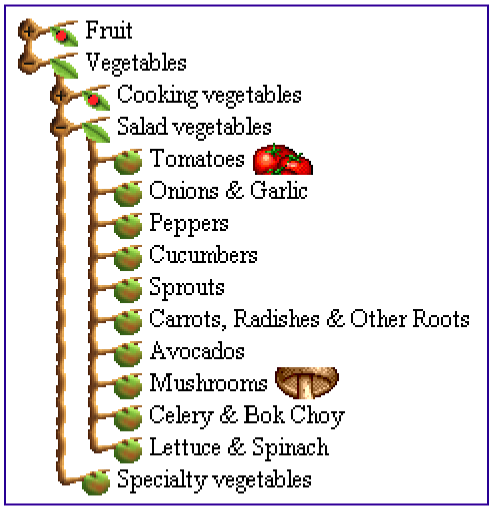
6.2.3.1.3.17 hasVisibleChildren

Indicates whether this node has visible children. A node with no visible children will be shown as collapsed.
In the tree below, the "Fruit" and "Cooking vegetables" nodes have no visible children.



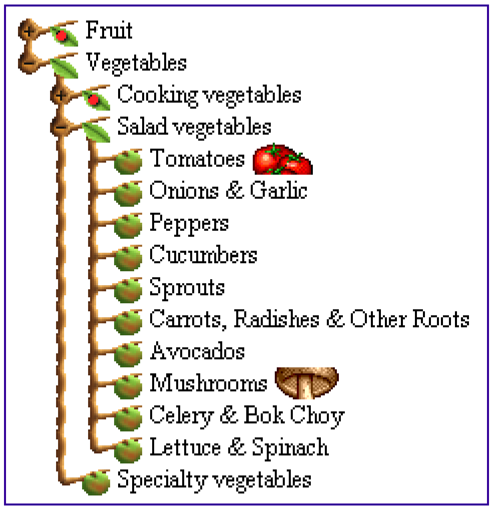
6.2.3.1.3.18 hasVisibleSiblingAbove

Indicates whether this node has any visible siblings above it.
In the tree below, the "Tomatoes" node has no visible siblings above it, while the "Specialty vegetables" node's first visible sibling above is the "Salad vegetables" node.



6.2.3.1.3.19 hasVisibleSiblingBelow

Indicates whether this node has any visible siblings below it.
In the tree below, the "Vegetables" node has no visible siblings below it, while the "Salad vegetables" node's first visible sibling above is the "Specialty vegetables" node.



6.2.3.1.3.20 insertLeafWithParamsAfter

Parameters			
#	**name**	**type**	**description**
1	`content`	string	The content of the node. It can be either plain text or html code including complex formatting and/or images. In the latter case, care must be taken that the resulting html code does not break the tree code.
2	`link`	string	The URL which will load when the user clicks on this node.
3	`target`	string	The name of the frame in which the preceding URL will be loaded.
4	`vsbl`	boolean	Specifies whether this node will be initially visible or not. A parent node with all its children invisible will be shown as collapsed.
5	`icon`	string	The file path to the image file for a custom icon to be used by this node. If not specified, the default icon(s) for this tree will be used.

Inserts a new node to the tree right after this node (as a sibling), and returns this new node.

6.2.3.1.3.21 isSibling

Parameters			
#	**name**	**type**	**description**
1	`leaf`	LiknoTreeElement	The node to check whether this node is a sibling with.

Indicates whether this node is a sibling to the specified node.

6.2.3.1.3.22 resetMouseOver

Resets (clears) the mouse over state (isMouseOver property) for this node.

6.2.3.1.3.23 resetSelected

Resets (clears) the selected state (isSelected property) for this node.
If this is a multiple selection tree (multiSelect property), it will also remove this node from the tree's selection property.

6.2.3.1.3.24 setClassnamesR

Parameters			
#	**name**	**type**	**description**
1	classnameDe	string	CSS class name to be used for the normal state.
2	classnameMo	string	CSS class name to be used for the mouse over state.
3	classnameSe	string	CSS class name to be used for the selected state.

Recursively sets the CSS classes to be used by this node and its children.
Note that this function takes 3 parameters, in contrast to the `setClassnames` function which sets the classnames property for this node only and takes an array of these 3 parameters.

6.2.3.1.3.25 setCustomBranchIcons

Parameters			
#	**name**	**type**	**description**
1	expandedicon	string	The file path to the icon image to be used for the normal state of this branch when it is expanded.
2	closedicon	string	The file path to the icon image to be used for the normal state of this branch when it is collapsed.
3	expandedicon_mo	string	The file path to the icon image to be used for the mouse over state of this branch when it is expanded.
4	closedicon_mo	string	The file path to the icon image to be used for the mouse over state of this branch when it is collapsed.
5	expandedicon_s	string	The file path to the icon image to be used for the selected state of this branch when it is expanded.
6	closedicon_s	string	The file path to the icon image to be used for the selected state of this branch when it is collapsed.

Sets custom icons to be used for the normal, mouse over and selected states of this node.
Note that these icons will be used as long as this node has children. Use setCustomLeafIcons to specify custom icons to be used when this node has no children.
If a parameter is not passed, the default icon for this state (based on the tree properties) will be used.

6.2.3.1.3.26 setCustomLeafIcons

Parameters			
#	**name**	**type**	**description**
1	leaficon	string	The file path to the icon image to be used for the normal state of this node.
2	leaficon_mo	string	The file path to the icon image to be used for the mouse over state of this node.
3	leaficon_s	string	The file path to the icon image to be used for the selected state of this node.

Sets custom icons to be used for the normal, mouse over and selected states of this node.
Note that these icons will be used as long as this node has no children. Use setCustomBranchIcons to specify custom icons to be used when this node has children.
If a parameter is not passed, the default icon for this state (based on the tree properties) will be used.

6.2.3.1.3.27 sortBranch

Parameters			
#	**name**	**type**	**description**

1	`onData`	number	A valid index in the data array or -1.
2	`recursively`	boolean	Specifies whether sorting will be performed recursively on the children of this node.

Sorts the nodes under this branch (recursively if `recursively` is true) based on user-defined data specified by the index `onData` (if any) or lexicographically if the index is -1 or invalid. Returns false if branch has no children.
You will usually not call this function directly, but rather the sortBranchId function.

6.2.3.2 LiknoTreeRoot

6.2.3.2.1 LiknoTreeRoot()

Parameters

#	**name**	**type**	**description**
1	`treeid`	string	The id of the tree (iid property).
2	`objid`	string	The id of the HTML element in the webpage where this tree will be attached to.
3	`x`	number	The horizontal offset from the top-left corner of the above HTML element where the tree will be placed (in pixels).
4	`y`	number	The vertical offset from the top-left corner of the above HTML element where the tree will be placed (in pixels).
5	`width`	number	The width of the bounding box of the tree in pixels. If 0, then the tree will expand/collapse freely horizontally. You can set the horzScroll property to define whether horizontal scrollbars will appear automatically if the tree contents do not fit in the specified width.
6	`height`	number	The height of the bounding box of the tree in pixels. If 0, then the tree will expand/collapse freely vertically.
7	`leafheight`	number	The default leaf height for each node in the tree in pixels. All nodes will have this height.
8	`vsbl`	boolean	Specifies whether the root of the tree will be visible or not. Currently not used, set to false.
9	`style`	string	The CSS style for the background of the tree. See the Using themes and Designing your own themes topics for more details.
10	`imgformat`	string	The format of the images used for the tree icons, can be either "gif" or "png".

This is the constructor for the LiknoTreeRoot object. You use this to put a new tree in your web page. For more details of use, see the example for the addClass function, or any of the examples at the end.

6.2.3.2.2 Properties

These are the properties of the LiknoTreeRoot object.
Note that the LiknoTreeRoot object also inherits all the properties of the LiknoTreeElement object.

6.2.3.2.2.1 allowFolderSelect

Indicates whether the user can select folders (otherwise only leaves will be selectable).

6.2.3.2.2.2 attachedTo

ID of the HTML element to which this tree is attached (usually a DIV element).

6.2.3.2.2.3 autoCollapse

Indicates whether clicking on a branch at the same level (depth) will collapse all other expanded branches at the same level (in the standard Microsoft Windows XP Explorer fashion).

6.2.3.2.2.4 boxHeight

Height of the box of the tree to be used (if 0, the tree will expand dynamically to the bottom).

6.2.3.2.2.5 boxWidth

Width of the box of the tree to be used (if 0, the tree will expand dynamically to the right).

6.2.3.2.2.6 clickToExpand

Indicates whether a single click on a branch will expand it.
By default, branches are expanded with a double click.

6.2.3.2.2.7 contextmenuId

ID of the HTML element of the node where the context menu of the tree is displayed over.

6.2.3.2.2.8 cssClasses

An array of an arbitrary number of pairs (arrays) of strings, each pair specifying a CSS class, where the first value is the name of the class and the second value is the set of styles used by this class.
This is where all the CSS classes used by the tree are kept. The names specified for each class will be modified in the final CSS definition to be unique for this tree globally in the HTML document. This means that you can specify the same names for the classes in different trees and they will not interfere with each other.
The names for the default classes must be "`de`", "`mo`" and "`se`" for the default, mouse over and selected states respectively.
See the example for the addClass function.

6.2.3.2.2.9 cssCreated

Indicates whether the CSS classes for this tree are already created.
If you add a new CSS class at runtime, set this to false before calling drawTree again, in order for the new CSS classes to take effect.

6.2.3.2.2.10 hasContextMenu

Indicates whether this tree has a custom context (right-click) menu.

6.2.3.2.2.11 horzScroll

Indicates whether horizontal scrollbars should be used if some node exceeds the width of the tree box.

6.2.3.2.2.12 imagesDir

The path/directory with the icons to be used by the tree.
Note that the icons must have specific filenames and they all have to be of the same image size (width & height) to be used by the tree.
In most cases your images must be 16x16 pixels, which is the standard size of icons used for tree controls.

Example:

```
treeD = new LiknoTreeRoot("treeD", "test4", 0, 0, 0, 0, 16, true, "background-color:white;
border:1px solid #330099; padding:4px;", "gif");
treeD.setImagesDir("images/ldmt/16x16/tree/");
```

ico	filename	description

n		
	empty.gif	Just an empty 16x16 image
📄	leaf.gif	Default icon for all nodes with no children
📄	leaf_mo.gif	Default icon for all nodes with no children (mouse over state)
📄	leaf_s.gif	Default icon for all nodes with no children (selected state)
┌	line_es.gif	Connecting line (East-South)
└	line_ne.gif	Connecting line (North-East)
├	line_nes.gif	Connecting line (North-East-South)
│	line_ns.gif	Connecting line (North-South)
⊟	minus_e.gif	Collapse icon (East)
⊟	minus_es.gif	Collapse icon (East-South)
⊟	minus_ne.gif	Collapse icon (North-East)
⊟	minus_nes.gif	Collapse icon (North-East-South)
⊞	plus_e.gif	Expand icon (East)
⊞	plus_es.gif	Expand icon (East-South)
⊞	plus_ne.gif	Expand icon (North-East)
⊞	plus_nes.gif	Expand icon (North-East-South)
📁	subtree_closed.gif	Default branch collapsed icon
📁	subtree_closed_mo.gif	Default branch collapsed icon (mouse over state)
📁	subtree_closed_s.gif	Default branch collapsed icon (selected state)
📂	subtree_open.gif	Default branch expanded icon
📂	subtree_open_mo.gif	Default branch expanded icon (mouse over state)
📂	subtree_open_s.gif	Default branch expanded icon (selected state)

6.2.3.2.2.13 isStatusbarDirty

Indicates whether the browser status bar text has been changed by this tree.

6.2.3.2.2.14 menuFx

Indicates whether this tree will use animated branch collapse/expand effects (visible only in Microsoft Internet Explorer).

6.2.3.2.2.15 mouseoverId

ID of the HTML element of the node where the mouse is currently over in the tree.

6.2.3.2.2.16 multiSelect

Indicates whether this tree allows multiple selections.
A tree which allows multiple selections can be used as an options control with check boxes. When the user clicks on a node, the previously selected node is not cleared.
You can use the selection property to get an array of all the selected nodes.

6.2.3.2.2.17 offsetX

Horizontal distance from the top left corner of the HTML element to which this tree is attached (in pixels).

6.2.3.2.2.18 offsetY

Vertical distance from the top left corner of the HTML element to which this tree is attached (in pixels).

6.2.3.2.2.19 prevStatusbarText

The text displayed in the browser status bar before this tree changed it.

6.2.3.2.2.20 selectedId

ID of the HTML element of the node currently selected in the tree.

6.2.3.2.2.21 selection

For trees which allow multiple selections, this is an array of the nodes which are currently selected.

6.2.3.2.2.22 sortAscending

Indicates whether sorting will be performed in ascending or descending order.
See the Stock market portfolio example.

6.2.3.2.2.23 sortOnData

A zero-based index in the data array to specify whether sorting in this tree will be performed on some user-defined data associated with the nodes, or -1 if sorting will be performed with a lexicographical comparison on the content property of the nodes.
See the Stock market portfolio example.

6.2.3.2.2.24 toggleSelected

Indicates whether clicking again on a selected node de-selects it.

6.2.3.2.2.25 treeNbr

Internal number used by all Likno trees, a zero-based index in the global LiknoTrees array.
You must not change this, you may only want to retrieve it at runtime in your Javascript code.

6.2.3.2.2.26 useCtrl

Indicates whether the Ctrl key should be used for multiple selections.
If true, the user must keep the Ctrl key pressed while clicking on a node to add it to the selection (in the standard Microsoft Windows Explorer fashion). If the user clicks on a node without pressing the Ctrl key, the previously selected nodes will be cleared.

6.2.3.2.3 Functions

These are the functions supported by the LiknoTreeRoot object.
The return type of each function, if any, is indicated underneath each function name.
If the returned variable is restricted to specific values, those will be indicated in brackets next to the return type.

6.2.3.2.3.1 addClass

Parameters

#	name	type	description
1	name	string	The name of this class to be used internally by the tree. Note that this will not be the name by which you can refer to this class in the rest of your HTML document.
2	styl	string	The set of CSS styles comprising this class.

Adds a new class in the set of classes used by this tree. If a class with the same name already exists in this tree, it will be replaced.
See cssClasses for more information.

Example:

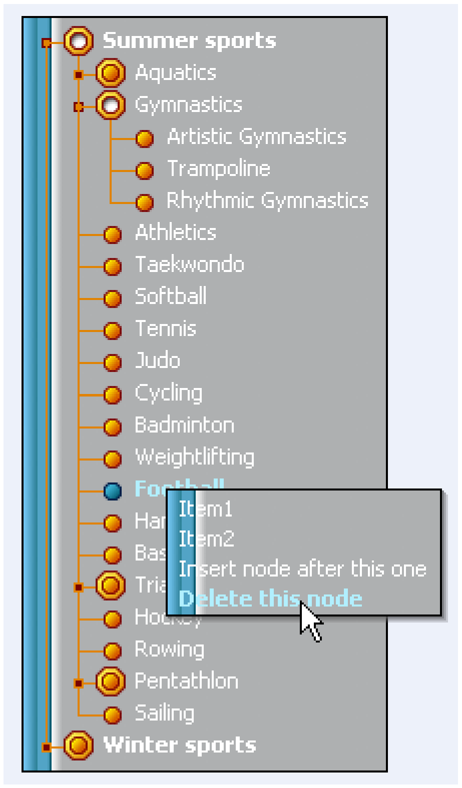
```
treeD = new LiknoTreeRoot("treeD", "test4", 0, 0, 0, 0, 16, true, "background-color:white;
border:1px solid #330099; padding:4px;", "gif");
treeD.setImagesDir("images/ldmt/16x16/tree/");
treeD.addClass("de", "font-family:Times New Roman; font-size:12px; color:black;
background-color:transparent; text-decoration:none; font-weight:normal; padding-left:4px;
padding-right:4px;");
treeD.addClass("mo", "font-family:Times New Roman; font-size:12px;
text-decoration:underline; color:#669900; padding-left:4px; padding-right:4px;");
treeD.addClass("se", "font-family:Times New Roman; font-size:12px; color:white;
background-color:#CC6600; padding-left:4px; padding-right:4px;");
```

6.2.3.2.3.2 addContextMenu

Parameters

#	name	type	description
1	menuitems	array	An array of even dimension, specifying an arbitrary number of pairs of values, where even indices hold the captions for each menu item, and odd indices hold a javascript function to be called whenever the user clicks on the corresponding menu item.

Adds a context (right-click) menu to the items in the tree.
The context menu will inherit the styles used by the tree by default.
See the Shopping cart management example for more details on how to use this function.



6.2.3.2.3.3 compareLeaves

Parameters			
#	name	type	description
1	`leaf1`	LiknoTreeElement	The first leaf to compare.
2	`leaf2`	LiknoTreeElement	The second leaf to compare.

Compares two leaves, based on this tree's sortOnData and sortAscending properties (if -1, a lexicographical comparison is performed).
See also compareTo.

6.2.3.2.3.4 createStylesheet

Creates the CSS <style> tag with the classes to be used by the tree. Most of the time you will not need to call this function directly, since it is called by drawTree if cssCreated is `false`.

6.2.3.2.3.5 deleteNode

Parameters			
#	name	type	description
1	`leaf`	LiknoTreeElement	The node we want to delete.

Deletes a node and all its children (recursively) from the tree.

6.2.3.2.3.6 drawTree

Writes the HTML code which draws the tree on your webpage. If cssCreated is `false` (which is by default the first time the tree is drawn), it will also create the necessary CSS style sheet for the classes used by your tree.
For details of usage, see the Examples topic at the end.

6.2.3.2.3.7 getLeafById

Parameters			
#	**name**	**type**	**description**
1	`leaf`	string	Specifies the name of this class to be used internally by the tree. Note that this will not be the name by which you can refer to this class in the rest of your HTML document.
2	`leafid`	string	Specifies the set of CSS styles comprising this class.

Searches under `leaf` for a node with iid = `leafid` and returns it if found, otherwise returns null.

6.2.3.2.3.8 loadTreeState

Loads the tree state (expanded/collapsed branches) from the associated cookie (if any).
Returns false if no cookie was found.

6.2.3.2.3.9 saveTreeState

Saves the tree state (expanded/collapsed branches) in a cookie for persistence between sessions.
You will probably want to call this function when the user leaves your webpage (in your `onunload` event handler).
See also the loadTreeState function.

6.2.3.2.3.10 selectBranch

Parameters			
#	**name**	**type**	**description**
1	`branchid`	string	Specifies the iid of the branch on which the action will be performed.

Selects the branch specified by `branchid`. For more information also check on the iid format.

The following examples assume your tree is named "myTree":

- select the second main item	myTree.selectBranch("myTree_1");
- select the third sub item of the first main item	myTree.selectBranch("myTree_0_2");

6.2.3.2.3.11 sortBranchId

Parameters			
#	**name**	**type**	**description**
1	`branchid`	string	Specifies the iid of the branch on which the action will be performed.
2	`onData`	number	A valid index in the data array or -1.
3	`recursively`	boolean	Indicates whether sorting will be performed on all children of this branch recursively.
4	`ascending`	boolean	Indicates the sorting order.

Sorts the children of the branch specified by `branchid` recursively if `recursively` is `true`. If `onData` is

a valid index in the array of associated data, sorting is performed using these values, otherwise a lexicographical sorting on the content of the nodes is performed.
The `onData` and `ascending` values set the sortOnData and sortAscending properties of this tree.
See the Stock market portfolio example.

6.2.3.2.3.12 toggleBranch

Parameters			
#	**name**	**type**	**description**
1	`branchid`	string	Specifies the iid of the branch on which the action will be performed.

Toggles the expanded/collapsed state of the branch specified by `branchid`.

6.3 Examples

6.3.1 Building a calendar tree

The following code creates a calendar tree for the current month, categorizing days into weeks starting on Monday and marking Sundays with a different color. It also marks the current date in green color.
Note: We use two functions, `y2k()` and `getWeek()` (found on the internet) to get the week number.

Example:

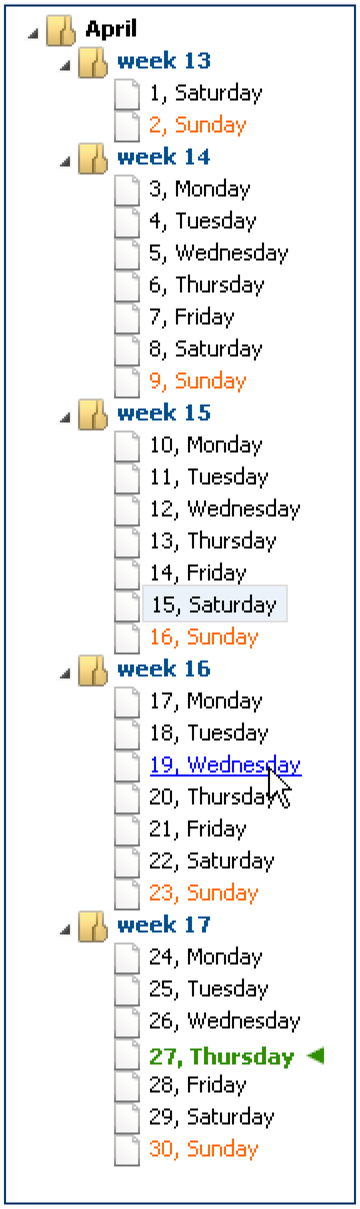
```
function createDatesTree()
{
    var today = new Date();
    var year = today.getFullYear();

    treeDates = new LiknoTreeRoot("treeDates", "example", 0, 0, 176, 600, 16, true,
"border:0; padding:4px; background-color:#FFFFFF; border:1px solid #003366;", "png");
    treeDates.setHorzScroll = false;
    treeDates.setImagesDir("images/ldmt/16x16/winvista/");
    treeDates.addClass("de", "font-family:Tahoma, Arial, sans-serif; font-size:11px;
color:black; background-color:transparent; text-decoration:none; font-weight:normal;
padding:2px 4px;");
    treeDates.addClass("mo", "font-family:Tahoma, Arial, sans-serif; font-size:11px;
text-decoration:underline; color:blue; padding:2px 4px;");
    treeDates.addClass("se", "font-family:Tahoma, Arial, sans-serif; font-size:11px;
color:black; background-color:#EBF2FA; padding:2px 4px; border:1px solid #DFDFE1;");
    var months = new Array("January","February","March","April","May","June","July","August"
,"September","October","November","December");
    var dayspermonth = new Array(31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31);
    var days = new Array("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
"Saturday", "Sunday");
    var day = 0;
    var week = -1;
    var month = today.getMonth();
    var t5w = new Array(6);
    t5d = new Array(31);
    var t5m = treeDates.addLeafWithParams("<b>"+months[month]+"</b>", "", "", true, "");
    var dayofmonth = 1; 22: var nodedate = new Date(year, month, dayofmonth);
    while(dayofmonth <= dayspermonth[month])
    {
         if(nodedate.getDay() == 1 || week == -1)
         {
              week++;
              t5w[week] = t5m.addLeafWithParams("<span style='color:#004f8f;'><b>week "+
getWeek(nodedate)+"</b></span>", "", "", true, "");
         }
         if(dayofmonth == today.getDate())
              t5d[day] = t5w[week].addLeafWithParams("<span style='color:#308f00;'><b>"+
dayofmonth+", "+days[nodedate.getDay()]+" <span style='font-family:arial;
font-size:13px;'>&#9668;</span></b></span>", "", "", true, "");
         else
         if(days[nodedate.getDay()] == "Sunday")
              t5d[day] = t5w[week].addLeafWithParams("<span style='color:#ff6600;'>"+
dayofmonth+", "+days[nodedate.getDay()]+"</span>", "", "", true, "");
         else
              t5d[day] = t5w[week].addLeafWithParams(dayofmonth+", "+days[nodedate.getDay
()], "", "", true, "");
         dayofmonth++;
         day++;
         nodedate = new Date(year, month, dayofmonth);
    }
    treeDates.drawTree();
}
```

In the example above, when the user clicks on a node in the tree, nothing happens.
If we wanted for example a PHP page ("showdate.php") with the date as a parameter to load inside a frame with the name "main" when the user clicks on a certain day, we could modify the code as follows:

Example:

```
...
            var link = "showdate.php?date="+nodedate.getFullYear()+"-"+month+"-"+dayofmonth;
            if(dayofmonth == today.getDate())
                t5d[day] = t5w[week].addLeafWithParams("<span style='color:#308f00;'><b>"+
dayofmonth+", "+days[nodedate.getDay()]+" <span style='font-family:arial;
font-size:13px;'>&#9668;</span></b></span>", link, "main", true, "");
            else
            if(days[nodedate.getDay()] == "Sunday")
                t5d[day] = t5w[week].addLeafWithParams("<span style='color:#ff6600;'>"+
dayofmonth+", "+days[nodedate.getDay()]+"</span>", link, "main", true, "");
            else
                t5d[day] = t5w[week].addLeafWithParams(dayofmonth+", "+days[nodedate.getDay
()], link, "main", true, "");
...
```

The code above will open the page "showdate.php?date=2006-3-10" when the user clicks on the "10, Monday" node.
This is how our example looks like:



6.3.2 Stock market portfolio

Let's say we would like to design a PHP/MySQL dynamic web application for users to manage their personal stock portfolios. The stock data are kept in a table like the following:

name	symbol	last_trade	volume	change	change_percent
Lucent Technologies Inc.	LU	$3.20	166,922,900	+0.03	0.95
Nortel Networks Corporation	NT	$3.06	37,914,900	+0.13	4.44
Time Warner Inc.	TWX	$16.69	27,816,900	-0.11	0.65
Pfizer Inc.	PFE	$25.09	24,630,700	-0.08	0.32
Motorola Inc.	MOT	$23.77	22,934,600	+0.48	2.06
Ford Motor Company	F	$7.65	22,898,400	+0.08	1.06
Merck & Co Inc.	MRK	$34.84	21,049,200	-1.15	3.20
General Electric Company	GE	$34.51	20,222,500	+0.09	0.26

Boston Scientific Corporation	BSX	$21.93	19,487,400	+0.24	1.11
AT&T Inc.	T	$26.45	17,838,200	-0.30	1.12
EMC Corporation	EMC	$13.62	17,452,900	-0.18	1.30
iShares MSCI Japan Index Fund	EWJ	$14.78	16,649,500	+0.07	0.48
Exxon Mobil Corporation	XOM	$62.09	15,409,100	-0.07	0.11
Goodman Global, Inc.	GGL	$20.05	14,329,400	+2.05	11.39
Rite Aid Corporation	RAD	$4.07	14,100,300	+0.06	1.50
Micron Technology Inc.	MU	$15.58	13,618,200	+0.02	0.13
Advanced Micro Devices, Inc.	AMD	$34.70	13,344,300	-0.08	0.23
Texas Instruments Inc.	TXN	$33.63	13,332,700	+0.41	1.23
BellSouth Corporation	BLS	$34.30	12,475,300	-0.51	1.47
UnitedHealth Group Inc.	UNH	$54.51	12,067,200	-0.79	1.43
St. Jude Medical, Inc.	STJ	$37.06	11,930,500	+0.81	2.23
Citigroup Inc.	C	$47.85	11,834,500	-0.41	0.85
Sprint Nextel Corporation	S	$26.40	11,607,900	-0.04	0.15
ConocoPhillips	COP	$67.25	11,314,700	+0.69	1.04
General Motors Corporation	GM	$19.55	10,492,900	-0.36	1.81
3M Company	MMM	$81.38	10,471,100	+3.92	5.06
Valero Energy Corporation	VLO	$62.80	10,096,400	+0.30	0.48
Verizon Communications	VZ	$33.68	9,662,900	-0.35	1.03
Spectrum Brands, Inc.	SPC	$15.51	9,366,000	-5.99	27.86
Corning Inc.	GLW	$26.35	9,164,500	+0.09	0.34

Assuming our table name is "stock", we can write some PHP code which will build our tree dynamically, using the stock names as captions for the nodes, and associating the rest information in the table with each node using the data property. This will allow us to perform actions like sorting the tree based on the actual stock values or the stock value changes. The user will probably want to have her stocks organized in her own categories, but for this example we will organize them in only 3 broad categories, those who went up, those who remained the same and those who went down. We will also set the tool tip for each node to the values in the table.
In the example below, the code is displayed against different background colors to indicate the different languages used, HTML, Javascript and PHP.

Example:

```
<html>
<head>
<title>Likno Drop-Down Menu Trees - Stock market portfolio example</title>
<style>
.btn
{
    border:2px #83452b outset;
    background-image:url(images/ldmt/16x16/wood/back.gif);
    font-family:tahoma;
    font-size:10px;
    font-weight:bold;
    padding:4px;
    cursor:pointer;
    cursor:hand;
    color:#83452b;
}
</style>
<script language="JavaScript" type="text/javascript" src="ldmt.js"></script>
</head>
<body>
<div id="example" style="position:absolute; left:16px; top:16px; width:100%;"></div>
<script language="JavaScript" type="text/javascript">
```

```
function sortOn(idx, asc)
{
    var node = LiknoTrees[0].findLeafByText("going up", false, false, true);
    if(node)
        LiknoTrees[0].sortBranchId(node.getIid(), idx, true, asc);
    node = LiknoTrees[0].findLeafByText("going down", false, false, true);
    if(node)
        LiknoTrees[0].sortBranchId(node.getIid(), idx, true, asc);
}

treeStock = new LiknoTreeRoot("treeStock", "example", 0, 0, 0, 0, 16, false, "padding:4px;
background-color:#ebdcc7; background-image:url(images/ldmt/16x16/wood/back.gif); border:2px
solid #83452b;", "png");
treeStock.setImagesDir("images/ldmt/16x16/wood/");
treeStock.addClass("de", "font-family:Tahoma; font-size:10px; color:#83452b;
background-color:; text-decoration:none; font-weight:normal; padding-left:4px;
padding-right:4px;");
treeStock.addClass("mo", "font-family:Tahoma; font-size:10px; text-decoration:underline;
color:#83452b; padding-left:4px; padding-right:4px;");
treeStock.addClass("se", "font-family:Tahoma; font-size:10px; font-weight: bold;
color:#83452b; background-color:; padding-left:4px; padding-right:4px;");
var stocks_up = treeStock.addLeafWithParams("<span style='color:green'>Going up</span>", "",
 "", true, "");
var stocks_same = treeStock.addLeafWithParams("<span style='color:darkblue'>No
change</span>", "", "", true, "");
var stocks_down = treeStock.addLeafWithParams("<span style='color:red'>Going down</span>",
"", "", true, "");
var stock;
```

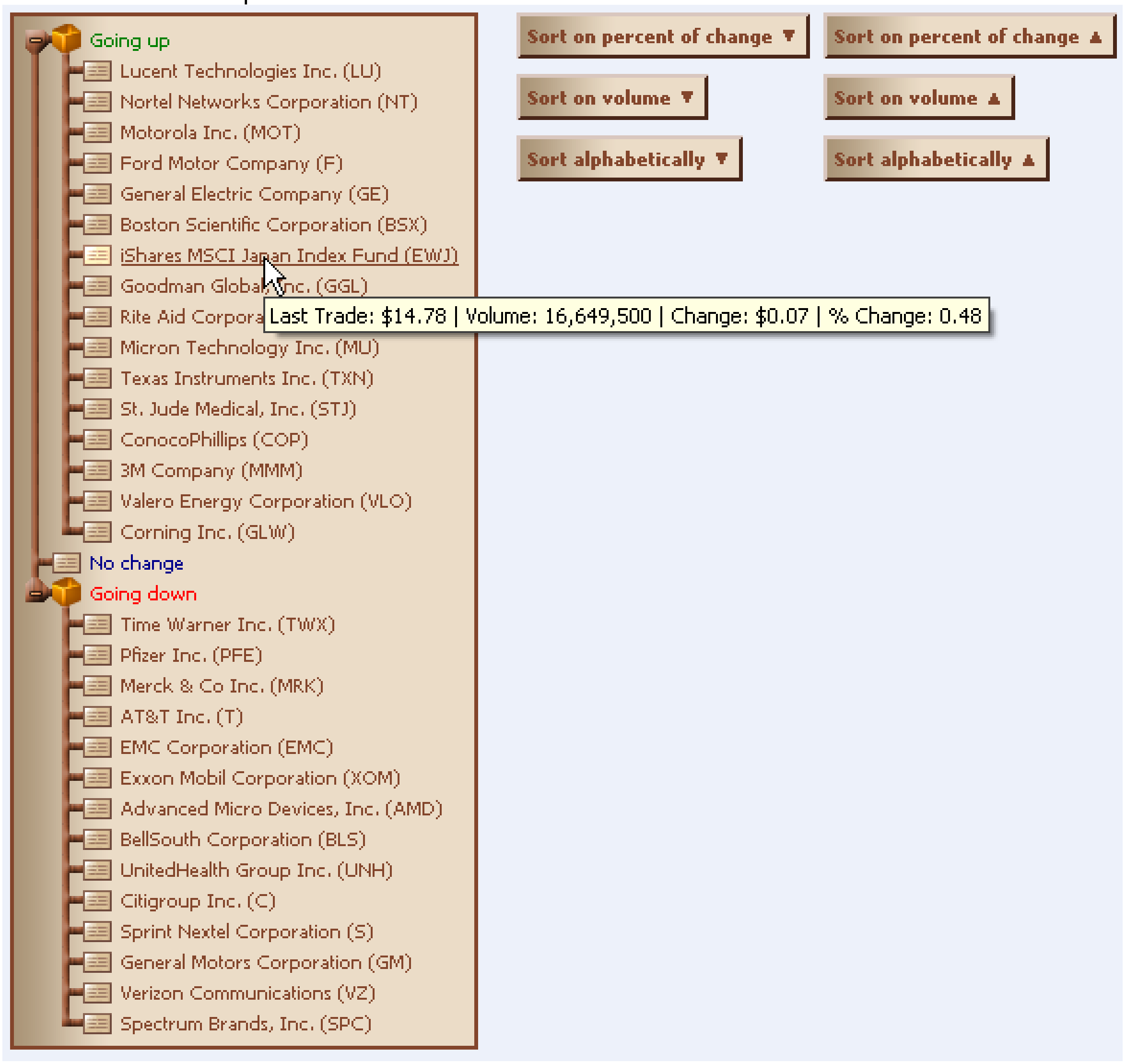
```
<?php
require_once('db.php');           // we keep the database connection details here
$query = "select * from stock";   // the actual query will probably be more complicated than
this
$result = mysql_query($query);
while($row = mysql_fetch_assoc($result))
{
    $cat = 'stocks_same';
    if($row['change'] > 0)
        $cat = 'stocks_up';
    else
    if($row['change'] < 0)
        $cat = 'stocks_down';
    echo "stock = $cat.addLeafWithParams('".$row['name']." (".$row['symbol'].")', '', '',
true, '');\n";
    echo "stock.setData(new Array('".$row['symbol']."', ".$row['last_trade'].", ".$row[
'volume'].", ".$row['change'].", ".$row['change_percent']."));\n";
    echo "stock.setTooltip('Last Trade: \$".$row['last_trade']." | Volume: ".strrev(substr(
chunk_split(strrev($row['volume']), 3, ','), 0, -1))." | Change: \$".$row['change']." | %
Change: ".$row['change_percent']."');\n";
}
?>
```

```
treeStock.drawTree();
```

```
</script>
<div class="btn" style="position:absolute; left:280px; top:16px;" onclick="sortOn(4, true);"
>Sort on percent of change &#9660;</div>
<div class="btn" style="position:absolute; left:440px; top:16px;" onclick="sortOn(4,
false);">Sort on percent of change &#9650;</div>
<div class="btn" style="position:absolute; left:280px; top:48px;" onclick="sortOn(2, true);"
>Sort on volume &#9660;</div>
<div class="btn" style="position:absolute; left:440px; top:48px;" onclick="sortOn(2,
false);">Sort on volume &#9650;</div>
<div class="btn" style="position:absolute; left:280px; top:80px;" onclick="sortOn(-1,
true);">Sort alphabetically &#9660;</div>
```

```
<div class="btn" style="position:absolute; left:440px; top:80px;" onclick="sortOn(-1,
false);">Sort alphabetically &#9650;</div>
</body>
</html>
```

This is how our example above looks like:


Going up
Lucent Technologies Inc. (LU)
Nortel Networks Corporation (NT)
Motorola Inc. (MOT)
Ford Motor Company (F)
General Electric Company (GE)
Boston Scientific Corporation (BSX)
iShares MSCI Japan Index Fund (EWJ)
Goodman Global Inc. (GGL)
Rite Aid Corpora
Micron Technology Inc. (MU)
Texas Instruments Inc. (TXN)
St. Jude Medical, Inc. (STJ)
ConocoPhillips (COP)
3M Company (MMM)
Valero Energy Corporation (VLO)
Corning Inc. (GLW)
No change
Going down
Time Warner Inc. (TWX)
Pfizer Inc. (PFE)
Merck & Co Inc. (MRK)
AT&T Inc. (T)
EMC Corporation (EMC)
Exxon Mobil Corporation (XOM)
Advanced Micro Devices, Inc. (AMD)
BellSouth Corporation (BLS)
UnitedHealth Group Inc. (UNH)
Citigroup Inc. (C)
Sprint Nextel Corporation (S)
General Motors Corporation (GM)
Verizon Communications (VZ)
Spectrum Brands, Inc. (SPC)
Sort on percent of change ▼
Sort on percent of change ▲
Sort on volume ▼
Sort on volume ▲
Sort alphabetically ▼
Sort alphabetically ▲

Last Trade: $14.78 | Volume: 16,649,500 | Change: $0.07 | % Change: 0.48

6.3.3 Shopping cart management

We would like to design a webpage which will list the items in a shopping cart organized and categorized in various ways and allow the buyer to add or remove items interactively. The buyer can remove or add items from her shopping cart dynamically, and she can check the total price or the price of individual items in the cart at all times.
Again, the code is displayed against different background colors to differentiate the HTML and Javascript parts.

Example:

```
<html>
```

```
<head>
<title>Likno Drop-Down Menu Trees - Shopping cart management example</title>

<script language="JavaScript1.2" type="text/javascript" src="ldmt.js"></script>
<script language="JavaScript1.2" type="text/javascript">

function calcPriceSelected()
{
    var items = LiknoTrees[0].getSelection().length;
    var price = 0;
    for(var i=0; i<items; i++)
    {
        var node = LiknoTrees[0].getSelection()[i];
        if(node.getData().length > 0 && node.getData()[0] > 0)
            price += node.getData()[0];
    }
    document.getElementById("priceselected").innerHTML = price + " €";
}

function calcPriceTotal()
{
    document.getElementById("pricetotal").innerHTML = calcPriceItems(LiknoTrees[0], 0) + " €
";
}

function calcPriceItems(node, price)
{
    var leaves = node.getLeaves();
    for(var i=0; i<leaves.length; i++)
        price = calcPriceItems(node.getLeaf(i), price);
    if(node.getData().length > 0 && node.getData()[0] > 0)
        price += node.getData()[0];
    return price;
}

function addSelected()
{
    var items = LiknoTrees[0].getSelection();
    if(items.length == 0)
        window.alert("You have not selected any items in your shopping cart!");
    else
    {
        for(var i=0; i<items.length; i++)
        {
            var content = items[i].getContent();
            if(content.substring(0, 44) == "<span style='text-decoration:line-through;'>")
            {
                 items[i].setContent(content.substring(44, content.lastIndexOf("</")));
                 var price = new Array();
                 price[0] = -1 * items[i].getData()[0];
                 items[i].setData(price);
            }
        }
    }
    LiknoTrees[0].drawTree();
    LiknoTrees[0].clearSelection();
    calcPriceSelected(); 51: calcPriceTotal();
}

function removeSelected()
{
    var items = LiknoTrees[0].getSelection();
    if(items.length == 0)
        window.alert("You have not selected any items in your shopping cart!");
    else
    {
        for(var i=0; i<items.length; i++)
        {
             items[i].setContent("<span style='text-decoration:line-through;'>"+items[i].
getContent()+"</span>");
             var price = new Array();
             price[0] = -1 * items[i].getData()[0];
             items[i].setData(price);
        }
    }
    LiknoTrees[0].drawTree();
    LiknoTrees[0].clearSelection();
    calcPriceSelected();
    calcPriceTotal();
}

function addNew()
{
    // Add new items in the shopping cart
}
```

```
</script>
</head>

<body style="background-color:#EDF1FA;">
<div style="position:absolute; left:16px; top:16px; font-family:tahoma; font-size:10px;
font-weight:bold;">Your shopping cart:</div>
<div id="cart" style="position:absolute; left:16px; top:32px; width:100%;"></div>
<div style="position:absolute; left:16px; top:292px; width:212px; border:1px solid black;
padding:4px;">
<table border='0' cellspacing='0' cellpadding='2' style="font-family:tahoma; font-size:10px;
font-weight:bold;">
<tr>
    <td align="right">Price of selected items:</td>
    <td id="priceselected"></td>
</tr>
<tr>
    <td align="right">Total price:</td>
    <td id="pricetotal"></td>
</tr>
</table>
</div>
<div style="position:absolute; left:16px; top:336px; font-family:tahoma; font-size:10px;
font-weight:bold;">Our products:</div>
<div id="shop" style="position:absolute; left:16px; top:352px; width:100%;"></div>
<script language="JavaScript1.2" type="text/javascript" src="treeCart.js"></script>
<script language="JavaScript1.2" type="text/javascript" src="treeShop.js"></script>
<script language="JavaScript1.2" type="text/javascript">
```

```
calcPriceTotal();
```

```
</script>
</body>

</html>
```

We use two trees in the same page, one to show the user's items in the cart, and one to show the available product categories. In a real situation, the code for each tree would be created on the web server, using a server-side language like PHP and querying a database. For simplicity, in our example we include only the generated Javascript code.
The code which builds the shopping cart tree is the following:

Example:

```
treeA = new LiknoTreeRoot("treeA", "cart", 0, 0, 212, 256, 16, false, "padding:4px;
background-color:#f7f7f7; border:1px solid #000066;", "png");
```

```
treeA.setImagesDir("images/ldmt/16x16/crystalballs/");
treeA.addClass("de", "font-family:Tahoma; font-size:11px; color:#000066; background-color:;
text-decoration:none; font-weight:normal; padding-left:4px; padding-right:4px;");
treeA.addClass("mo", "font-family:Tahoma; font-size:11px; text-decoration:underline;
color:#FF6D15; padding-left:4px; padding-right:4px;");
treeA.addClass("se", "font-family:Tahoma; font-size:11px; font-weight: bold; color:#ff4a00;
background-color:; padding-left:4px; padding-right:4px;");

treeA.setClickToExpand(true);
treeA.setAllowFolderSelect(false);
treeA.setMultiSelect(true);
treeA.setUseCtrl(true);

t2_a = treeA.addLeafWithParams("Cameras", "", "", true, "");
t2_a0 = t2_a.addLeafWithParams("Digital cameras", "", "", true, "");
t2_a00 = t2_a0.addLeafWithParams("PANASONIC Lumix DMC-FX01 black", "", "", true, "");
var price = new Array();
price[0] = 376;
t2_a00.setData(price);
t2_a00.setTooltip("376 €");
t2_a00.setJscriptCommand("calcPriceSelected();");

t2_a2 = t2_a.addLeafWithParams("Accessories", "", "", true, "");
t2_a20 = t2_a2.addLeafWithParams("Cases", "", "", true, "");
t2_a200 = t2_a20.addLeafWithParams("AQUAPAC Waterproof case - 410", "", "", true, "");
price = new Array();
price[0] = 25;
t2_a200.setData(price);
t2_a200.setTooltip("25 €");
t2_a200.setJscriptCommand("calcPriceSelected();");

t2_a21 = t2_a2.addLeafWithParams("Batteries", "", "", true, "");
t2_a210 = t2_a21.addLeafWithParams("Rechargeable", "", "", true, "");
t2_a2100 = t2_a210.addLeafWithParams("Panasonic CGA-S005E/1B battery", "", "", true, "");
price = new Array();
price[0] = 69;
t2_a2100.setData(price);
t2_a2100.setTooltip("69 €");
t2_a2100.setJscriptCommand("calcPriceSelected();");

t2_b = treeA.addLeafWithParams("Sound", "", "", true, "");
t2_b0 = t2_b.addLeafWithParams("MP3 Players", "", "", true, "");
t2_b00 = t2_b0.addLeafWithParams("SAMSUNG Multimedia player YP-T7FX 512 MB", "", "", true,
"");
price = new Array();
price[0] = 138;
t2_b00.setData(price);
t2_b00.setTooltip("138 €");
t2_b00.setJscriptCommand("calcPriceSelected();");

treeA.drawTree();
treeA.addContextMenu(new Array("Remove selected items", "removeSelected()", "Add selected
items", "addSelected()", "Add new items", "addNew()"));
```

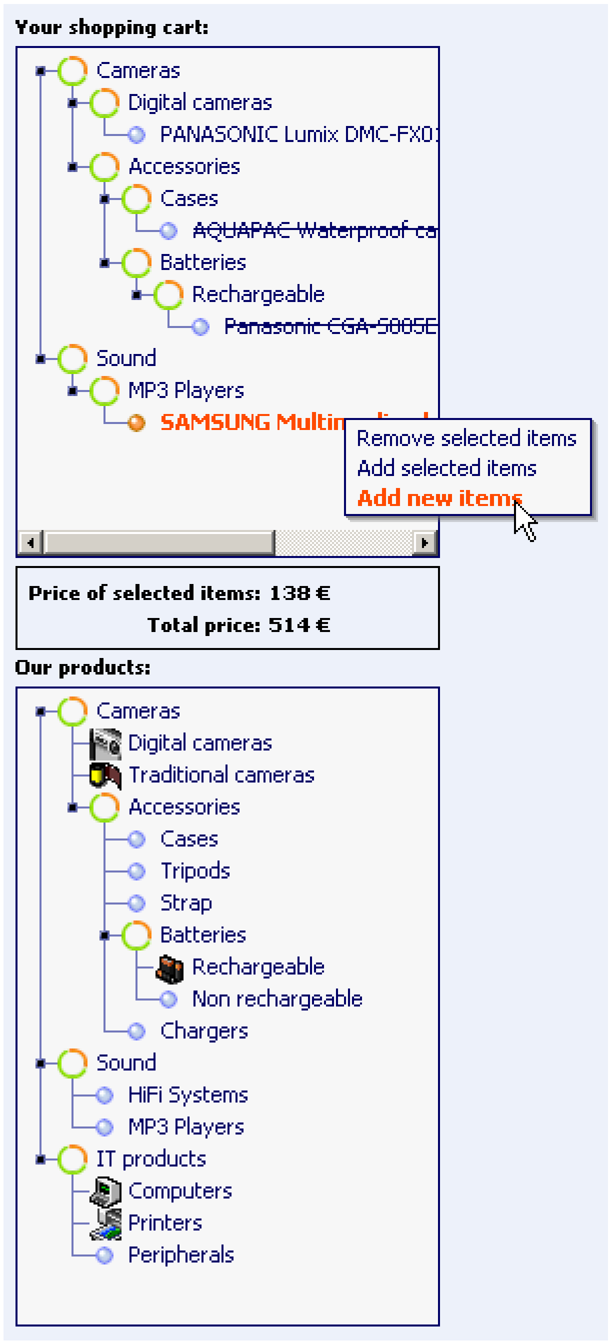
Again, we use the data property to hold the price of each item, and we also add a tool tip to allow the user to see the price for each item when she moves the mouse over that item. We also add a Javascript command (`calcPriceSelected`) to be executed each time the user clicks on one of the items in her shopping cart. The user can press and hold the Ctrl key to select multiple items. When `calcPriceSelected` is called, the price for all currently selected items in the cart is calculated.
We also added a custom context menu to allow the user to temporarily remove items from the cart, or add new items from the bottom tree.
The code for the product categories tree is much simpler, we only add some custom icons in some nodes and specify each node to load our `products.php` page with the category id as a parameter (e.g. `products.php?catid=7`) in the frame "`main`". The idea is that when the user selects a category from the bottom tree, the corresponding list of products will load in the frame "`main`" to allow her to select more products. Then she can use the "Add new items" menu option from the custom context

menu of the top tree to add the items she selected in her cart.
The code which builds the product categories tree is the following:

Example:

```
treeB = new LiknoTreeRoot("treeB", "shop", 0, 0, 212, 320, 16, false, "padding:4px;
background-color:#f7f7f7; border:1px solid #000066;", "png");
treeB.setImagesDir("images/ldmt/16x16/crystalballs/");
treeB.addClass("de", "font-family:Tahoma; font-size:11px; color:#000066; background-color:;
text-decoration:none; font-weight:normal; padding-left:4px; padding-right:4px;");
treeB.addClass("mo", "font-family:Tahoma; font-size:11px; text-decoration:underline;
color:#FF6D15; padding-left:4px; padding-right:4px;");
treeB.addClass("se", "font-family:Tahoma; font-size:11px; font-weight: bold; color:#ff4a00;
background-color:; padding-left:4px; padding-right:4px;");
treeB.setClickToExpand(true);

t2_a = treeB.addLeafWithParams("Cameras", "products.php?catid=1", "main", true, "");
t2_a0 = t2_a.addLeafWithParams("Digital cameras", "products.php?catid=2", "main", true,
"images/cameras_digital.gif");
t2_a1 = t2_a.addLeafWithParams("Traditional cameras", "products.php?catid=3", "main", true,
"images/cameras_traditional.gif");
t2_a2 = t2_a.addLeafWithParams("Accessories", "products.php?catid=4", "main", true, "");
t2_a20 = t2_a2.addLeafWithParams("Cases", "products.php?catid=5", "main", true, "");
t2_a21 = t2_a2.addLeafWithParams("Tripods", "products.php?catid=6", "main", true, "");
t2_a22 = t2_a2.addLeafWithParams("Strap", "products.php?catid=7", "main", true, "");
t2_a23 = t2_a2.addLeafWithParams("Camera connectivity", "products.php?catid=8", "main",
false, "images/connectivity.gif");
t2_a24 = t2_a2.addLeafWithParams("Batteries", "products.php?catid=9", "main", true, "");
t2_a240 = t2_a24.addLeafWithParams("Rechargeable", "products.php?catid=10", "main", true,
"images/batteries.gif");
t2_a241 = t2_a24.addLeafWithParams("Non rechargeable", "products.php?catid=11", "main", true
, "");
t2_a25 = t2_a2.addLeafWithParams("Chargers", "products.php?catid=12", "main", true, "");

t2_b = treeB.addLeafWithParams("Sound", "products.php?catid=13", "main", true, "");
t2_b0 = t2_b.addLeafWithParams("HiFi Systems", "products.php?catid=14", "main", true, "");
t2_b1 = t2_b.addLeafWithParams("MP3 Players", "products.php?catid=15", "main", true, "");

t2_c = treeB.addLeafWithParams("IT products", "products.php?catid=16", "main", true, "");
t2_c0 = t2_c.addLeafWithParams("Computers", "products.php?catid=17", "main", true,
"images/computers.gif");
t2_c1 = t2_c.addLeafWithParams("Printers", "products.php?catid=18", "main", true,
"images/printers.gif");
t2_c2 = t2_c.addLeafWithParams("Peripherals", "products.php?catid=19", "main", true, "");

treeB.drawTree();
```

This is how our example looks like:



6.4 Themes

6.4.1 Using themes

A Likno Drop-Down Menu Trees theme is a collection of elements which can give your tree a unique look.
A theme is comprised of the following:

- A unique set of icons (for more details see the imagesDir topic)
- A set of 3 CSS styles for the normal, mouse over and selected states respectively (for more details see the cssClasses topic)

- A style for the background of the tree, which may include a tileable background image

To use a theme in the API, all you have to do is use the `setImagesDir` and `addClass` functions, as you can see in the example for the addClass function.

6.4.2 Designing your own themes

To design your own theme, you have to follow these steps:

1. Design a custom set of icons, in most cases at 16x16 pixels size, and save them in the PNG or GIF format using the filenames described in the imagesDir topic.
   GIF icons can have 1-bit transparency (which basically means "hard" edges around the icons), while PNG icons can have per-pixel transparency, blending with the tree background and allowing for some nice shadow/glow effects. For PNG icons you do not need to worry about the lack of transparent PNGs by Microsoft Internet Explorer 6 or earlier, because it is handled automatically by the tree code internally.
   Great care must also be taken so that the icons are tileable. For example, the connecting lines (if any) must blend seamlessly at the edges.

2. Design a set of 3 CSS styles for the default normal, mouse over and selected states respectively. Although there are no limitations in the CSS styles which you can use, care must be taken not to "break" the tree look. For example, if you define a very big font size, the items will be drawn too high apart and the tree will become dysfunctional.
   See the example for the addClass function on how to use these classes in your trees.

3. Design a CSS style for the background of your tree, usually comprising of a border style, a padding and a background color or image. If you define a background image, care must be taken again that this image is tileable and does not obscure the icons or the text of the tree.

Index

http://www.likno.com