Tree Applet

Parameters and interface functions description

Tree applet can load content in two modes:

- initialize it from PARAM tags, present in HTML that contain the applet (these tags are described below in section "Tree parameters")
- load items located on sub-levels dynamically from the server where the applet is loaded from

In both cases it is possible to manage items of the tree applet using public methods of it. This can be done as from another Java applet that uses Tree applet, so from browser using JavaScript. The public methods are described below in the section "Tree methods".

1. Tree applets parameters.

Initializing of tree items.

To populate tree with data you should specify for every item set of required parameters, listed below.

Name	Required	Possible values	Description
ITEMi	Yes	according to IDC parameter	Item's identifier. Used as reference to item while accessing from script functions.
NAMEi	Yes	Any string value	Visible name of tree item, which will be displayed on screen.
ACTIONi	Yes	Valid parameters for script function	List of parameters, transmitted to script function, described by EVAL parameter. Can accept: 1. Integer value 2. String value in single or double quotation marks. 3. Collection of (1) and (2), separated by commas.
LEVELi	Yes	Integer	Level of current node. Must be from 0 till previous item's level + 1.
SELECTi	No	Any value	If specified, then current item will be selected.(Cursor will be placed on last selected item.)
IM0i, IM1i	No	Integer	Indexes of images (given by IMAGE parameters), associated with current item. Ignored if no ICONS_PERMANENT parameter specified. It is possible to specify a file name directly in this parameter. In this case the parameter should have value in format f<file_name>
CHILDi	No	Any value	Use in dynamical tree only. Not null value means non-terminal node - children for it will be loaded from server. (See also GET_DOC parameter)

COLORi	No	HTML color format	The color of item's text
SELCOLORi	No	HTML color format	The color of text for selected items
CHECKEDi	No	Any value	If specified, then current item will be checked (using with CHECKBOXES parameter)

Note: 'i' indicates the sequential number of an item in initialization list. Starts from 0.

Example 1. Possible item definition.

```
<PARAM NANE='ITEM0' VALUE='0'>
<PARAM NANE='NAME0' VALUE='Root item'>
<PARAM NANE='ACTION0' VALUE='r13'>
<PARAM NANE='LEVEL0' VALUE='0'>
<PARAM NANE='SELECT0' VALUE='1'>
```

Color parameters:

This set of parameters describes the color model of the tree applet.

Name	Description
BG_COLOR	Background color.
FG_COLOR	Color of non-selected item's label.
BGSEL_COLOR	Background of selected item.
FGSEL_COLOR	Color of selected item's label.

Other parameters.

CROSSES

If this parameter has non-empty value, non-terminal items will be marked with "+" on the left side

CHECKBOXES

Creates tree with checkboxes for all nodes. Images for checkboxes has size 11x11 pt and can be specified with followed parameters:

Name	Description	
IMAGE_CHECKED	Image for checked item	icons
IMAGE_UNCHECKED	Image for unchecked item	icons
IMAGE_SHADEN	Image for unchecked item, who has checked children on next levels	icons

This parameter can be signed with AutoCheckBoxes and SHOW_IMAGES flags:

AutoCheckBoxes

Processing of checks is automatic - if item checked, all subnodes becomes to checked state too and all parents came into shadow mode. Default value is "1".

SHOW_IMAGES

Shows checkboxes with images. Default value is "1".

IMAGE

Specifies the list of images for the tree items. Has structure IMAGE0, IMAGE1, ... ,IMAGEn. This numbers can be passed in IM0 and IM1 parameters of an item (see Item initializing tree items.).

If ICONS_PERMANENT parameter is not specified, they will have next meaning:
IMAGE0 - Image for root nodes.
IMAGE1 - Image for terminal nodes.
IMAGE2 - Image for non-terminal opened nodes.
IMAGE3 - Image for non-terminal closed nodes.

The path for image starts from TreeApplet class folder.

ICONS_PERMANENT

If this parameter has non-empty value, then images associated with the item should be given in IM0,IM1 parameters, otherwise, default images from IMAGE0...IMAGE3 parameters will be used instead.

Important: If this parameter specified, minimum one of images (IM0 or IM1) must be specified for each node. Otherwise the tree will not be initialized.

EVAL

The name of script function, which will be called when any item in tree is selected. Values specified in ACTION parameter will be passed to this function.

Example 2. Function with single argument.

```
function foo( s ) { }

<PARAM name="EVAL" VALUE="f">
<PARAM name="ACTION0" VALUE="0">  // foo(0) will be called
<PARAM name="ACTION1" VALUE="'test'"> // foo('test') will be called
```

Example 3. Function with two arguments.

```
function foo( s1, s2 ) { }

<PARAM name="EVAL" VALUE="f">
<PARAM name="ACTION0" VALUE="0,0">  // foo(0,0) will be called
<PARAM name="ACTION0" VALUE="'test',28"> // foo('test',28) will be called
<PARAM name="ACTION0" VALUE="28"> // Invalid, function does't take one argument.
```

HIER_SEL

Hierarchical selection flag. By default you cannot select node item and its child simultaneously. To enable child selection you should set this parameter to any nonempty value.

ROOT_CLOSEABLE

Allows you to collapse root nodes with LEVEL=0. Disabled by default.

GET_DOC

Used in dynamical trees. It means the name of document, which will load the data from server. Loading happens when:

1. This parameter is not null,
2. Node hasn't child subnodes
3. Parameter CHILD for this node is not null.

The document can get identifier of called node through **id** parameter and must return the list of strings, ended by empty string. Every string contains information about single child node in the next format:

```
"Name","id","action",hasChildren,im0,im1
```

Where id - identifier of the node, name - its name, action - the same meaning as ACTION parameter, hasChildren - CHILD parameter, im0,im1 - IM0,IM1 parameters.

Example 4. ASP document for loading subnodes from database.

```
<%
id = request("id")
sqlStr ="select KeyField, NameField from DataTable where ParentKeyFied=" & id
set ResultSet = DBConnection.execute(sqlStr)
while not ResultSet.eof
          Key = ResultSet("KeyField")
          Name = ResultSet("NameField")
          response.writeln("""" & Name & """,""" & Key & """,""" & Key &
""",0,1,2" )
          ResultSet.moveNext
wend
response.writeln("")
%>
```

ON_LOAD

The name of script function, which will be called when all items have been loaded into the tree and you can call any tree applet's method.

ON_DBLCLICK

The name of script function, which will be called after doubleclicking on node. Function takes single parameter - the identifier of node.

ON_MENU

The name of script function, which will be called after changing of checked state of item. It has two parameters - identifier of item and boolean value of current checked state.

MULTISELECT

Enabling (1) or disabling (0) multiple selection mode of tree. Default it's enabled.

NODE_HEIGHT

The height of node. All nodes in tree has the same height. Default value is 18.

IMAGE_WIDTH / **IMAGE_HEIGHT**

Size of icons in tree. If real size of any image differs from this value, image will be resized. Default value is 16x16.

FONT_NAME / **FONT_SIZE** / **FONT_STYLE**

Font for nodes text. Default value is "Verdana", size 13, style PLAIN.

FONT_STYLE is a composition of the next string values: PLAIN, BOLD, ITALIC, separated by commas.

MENU

Enabling menu in tree applet. It will be called after right-button clicking on node. The value of this parameter is the name of script function, wich will be called after selection in menu. It must have two parameters - the identifier of node and number of selected item in menu (started from 0). If this parameter specified, next parameters will be processed:

MENUi

Specifies the name of menu item. Can has any string values or 'SEPARATOR' for inserting menu separator between menu items.

ON_MENU_PRE

Specifies menu initialization script function. This function should be used for customizing popup menu and is called directly before displaying menu and takes two parameters: identifier of node and TreeMenu object. Initially this object contains parameters defined with MENUi parameters. This object has next set of methods:

`void addString(String txt)`

Adds new item with specified label into menu.

`void addSeparator()`

Adds seperator into menu.

`void remove(int index)`

Removes item from menu.

void removeAll(int index)

Removes all items from menu.

String getLabel(int index)

Sets label of specified item.

void setLabel(int index, String txt)

Gets label of specified item.

void setEnable(int index, boolean isEnable)

Sets specified item to enabled or disabled state.

2. Tree applet methods.

Below is described a list of tree applet methods available from client side scripts to access and manipulate tree data.

Retrieving of selected items.

`boolean hasData()`

Returns true if there are some selected items in tree.

`String getSelectedItems()`

Returns the names of selected items, separated by commas. Empty string will be returned if zero items are selected.

`String getSelectedItemsId(<int sortOrder>)`

Returns the identifiers of selected items, separated by commas. Empty string will be returned if zero items are selected. If sortOrder=0 (default) identifiers will be returned in natural selection order, else in top-down order

`String getLastSelectedId()`

Returns the identifier of the last selected item. Empty string will be returned if zero items are selected.

`boolean isLastItemSelected()`

Returns true, if the last selected item is deselected by repeated pressing on it. (The selected item has color border around it)

Retrieving of item properties.

`String getLabel(String id)`

Returns the label of item. Null if item with such id not found.

`String getAction(String id)`

Returns the action string for item. Null if item not found.

`String getParentId(String id)`

Returns the parent ID for item. Null if item not found or if it is the root node.

`String getSubItems(String id)`

Returns the list of all child items from the next level of the tree, separated by commas.

`String getAllSubItems(String id)`

Returns the list of all child items from all next levels of the tree, separated by commas.

`String findRef(String act)`

Returns the identifier of node with action **act**. Null if act not found.

Selecting of items.

`void selectItem(String id [, boolean isCall])`

Moves selection to the item with specified ID. If isCall parameter is true (default), script function for the newly selected node will be called.

Tree manipulation functions.

`void deleteSelectedItem()`

Deletes selected item.

`void deleteItem(String id [, boolean isSel])`

Deletes item with 'id' identifier. If isSel parameter is true (default), the parent of deleted item becomes
selected, otherwise there will be no selected items in the tree.

`void deleteAllNext(String id)`

Deletes item with 'id' identifier and all items on the same level of the tree.

`void setLabel( String id, String label)`

Sets label for the item with specified ID.

```
void insertChild( String id, String label, String action
   [, int String, String im1] )
```

Inserts new child item for selected node.

```
void insertNewChild( String pId, String id, String label, String action
   [, int im0, int im1] [, String options])
```

Inserts new child item for node with 'pId' ID. 'id' - is the identifier for newly created item.
options is composition of the next values separated by commas:

SELECT - new item becomes selected
CALL - action function for new item will be called after insertion
CHILD - new item has children (for dynamical tree)
CHECKED - new item is checked (for tree with checkboxes)

TOP - inserts new node at first child position

Default value for options is "SELECT,CALL,CHILD"

```
void insertNewNext( String pId, String id, String label, String action
   [, int im0, int im1] [, String options])
```

Inserts new node at the position next to the 'pId' item on the same tree level.

```
void insertNewRoot( String id, String label, String action
   [, int im0, int im1] [, String options] )
```

Inserts new root node at LEVEL=0.

```
void reloadTree(String docName)
```

Forces a full reload of tree by reading of specified document from the server.
The document should be in the format specified in GET_DOC section.

```
void setItemColor(String id, String strNormColor, String strSelColor)
```

Sets the color of specified item.

```
void setItemFont(String id [,String fName] ,String fStyle)
```

Sets the font of specified item

```
void setUnderline(String id, boolean isUnderline)
```

Sets Underline flag for specified item

void setItemImage(String id, String im1 [, String im2])

Sets image for specified item.

Expanding/collapsing functions.

```
void expandAllChildren( String id )
```

Expands all children of item with 'id' identifier (also expands subnodes).

```
void expandAllSelectedChildren( String id )
```

Makes all selected subitems of item with 'id' identifier visible.

```
void collapseAllChildren( String id )
```

Collapses all children of item with 'id' identifier (with its subnodes.).

```
void openItem( String id )
```

Opens item.

Cut & Paste functions.

`int doCut()`

Cuts all selected items from the tree. Returns the number of items that were cut.

`int doPaste(String id)`

Pastes the cut items as child of specified node.
Return values:

0 - successful
1 - no items was cut
2 - One of cut item is root of the tree
3 - specified item not found
4 - Parent - to - child inserting

Miscellaneous functions.

`String getSubItems(String id)`

Returns identifiers of direct subnodes of specified item separated by commas.

`String getAllSubItems(String id)`

Returns identifiers of all subnodes (also subnodes of subnodes) of specified item separated by commas.

`String getLevel(String id)`

Returns zero-based level of item.

`String findRefByLabel(String lbl)`

?? Returns separated by comas list of items with specified label

`String getLastAllSubItems(String id)`

Similar to the getAllSubItems except we only want the last descendant.

`String getNextItem(String id)`

Return the id of the item that appears next sequentially in the tree.

`String getNextSibling(String id)`

Return the id of the next sibling after parameter id.

`void traceTree()`

Prints the content of current tree into the Java log panel as an applet <PARAM> tags.

User specified data manipulation functions.

`boolean setUserData(String id, String key, String value)`

Sets user-specified data to node. Returns true if node was found.

`String getUserData(String id, String key)`

Returns user-specified data from node. Returns null value if node or data key not found.

Checkboxes operation functions

`boolean isItemChecked(String id)`

Returns true if specified item was checked.

`void setCheck(String id, boolean isCheck)`

Sets check state for specified item

`String getAllChecked(boolean isSubitems)`

Returns list of checked items, separated by commas. If isSubitems parameter is set to true, all items will be returned, else only first checked item without it's children.

`void setSubChecked(String id, boolean isCheck)`

Sets all subitems to checked (**isCheck=true**) or unchecked (**isCheck=false**) state. If **id=null** applying for all items in tree.

`String getAboveChecked(String id)`

Gets identifiers of all checked items above specified in top-down order.

`String getBelowChecked(String id)`

Gets identifiers of all checked items below specified in top-down order.