CLIENTTREE

This chapter covers the following topics:

- Example
- Properties

Example

The following example shows a simple client tree:

The XML layout definition is:

```
<rowarea name="Clienttree">
    <clienttree treecollectionprop="tree" height="200" withplusminus="true"
                treestyle="background-color:#FEFEEE">
    </clienttree>
</rowarea>
```

In this example, the client tree is directly put as row into the ROWAREA container. The property `treecollectionprop` contains a reference to the property `tree` which contains the net data of the tree. With the property `treestyle`, an explicit background color is set.

The Java code of the adapter is:

```
// This class is a generated one.

import com.softwareag.cis.server.Adapter;
import com.softwareag.cis.server.util.NODEInfo;
import com.softwareag.cis.server.util.TREECollection;

public class ClientTreeAdapter
    extends Adapter
{
    // class >TreeItem<
    public class TreeItem extends NODEInfo
    {
```

```
        public TreeItem(String text)
        {
            super(text);
        }

        public void reactOnSelect()
        {
            outputMessage("S", "Node " + getText() + " was selected.");
        }

        public void reactOnToggle()
        {
            // TODO Auto-generated method stub
        }
    }

    // property >tree<
    TREECollection m_tree = new TREECollection();
    public TREECollection getTree() { return m_tree; }
    public void setTree(TREECollection value) { m_tree = value; }

    /** initialisation - called when creating this instance*/
    public void init()
    {
        TreeItem file = new TreeItem("File");
        m_tree.addTopNode(file, false);
        m_tree.toggleNode(file); // open this file node to open immediately
        TreeItem news = new TreeItem("New");
        m_tree.addSubNode(news, file, false, false);
        TreeItem project = new TreeItem("Project");
        m_tree.addSubNode(project, news, true, false);
        TreeItem packages = new TreeItem("Package");
        m_tree.addSubNode(packages, news, true, false);
        TreeItem classes = new TreeItem("Class");
        m_tree.addSubNode(classes, news, true, false);
        TreeItem close = new TreeItem("Close");
        m_tree.addSubNode(close, file, true, false);
        TreeItem closeAll = new TreeItem("Close All");
        m_tree.addSubNode(closeAll, file, true, false);
        TreeItem save = new TreeItem("Save");
        m_tree.addSubNode(save, file, true, false);
        TreeItem saveAll = new TreeItem("Save All");
        m_tree.addSubNode(saveAll, file, true, false);
        TreeItem print = new TreeItem("Exit");
        m_tree.addSubNode(print, file, true, false);
        TreeItem exit = new TreeItem("Exit");
        m_tree.addSubNode(exit, file, true, false);
        TreeItem edit = new TreeItem("Edit");
        m_tree.addTopNode(edit, false); // open this file node to open immediately
        m_tree.toggleNode(edit);
        TreeItem undo = new TreeItem("Undo");
        m_tree.addSubNode(undo, edit, true, false);
    }
}
```

You see that the implementation of the server-side representation of the client tree control is using the same mechanisms as the one which you got to know when the TREENODE2 control was explained.

The difference is:

- The `reactOnToggle()` method needs not to be implemented. There is no explicit toggle event passed to the server. The toggling is done completely on the client side.

You also see that the top nodes are toggled immediately via the `TREECollection` API after creation:

```
MyNODEInfo edit = new MyNODEInfo("Edit");
m_tree.addTopNode(edit, false); // open this file node to open immediately
m_tree.toggleNode(edit);
```

Reason: by default, foldable tree nodes are opened with "closed" status. If they are toggled in the creation process, then they are already open when the tree is shown the first time.

Properties

Basic			
treecollectionprop	Name of adapter property representing the tree of items on server side. The property must be of type "TREECollection". Please view in Java API Documentation for more information.	Optional	
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor’s tree view.	Optional	
Appearance			
withplusminus	If set to "true" then +/- Icons will be rendered in front of the tree items.	Optional	true false

withtooltip	If set to "true" then the text of an item is also available as tool tip. Use this option in case you expect that the horizontal space of the item will not be sufficient to display the whole text of the item.	Optional	true false
selectionvisible	If set to "true" then the clicked item will also marked with a certain background color. The background color is defined by the style sheet settings.	Optional	true false
singleselect	If set to "true" then only one item can be selected. If set to "false" then multiple icons can be selected.	Optional	true false
imageopened	Image of a tree node that has subnodes and that is currently showing its nodes. The image either is defined statically by this property or also may be defined dynamically - see the corresponding properties defined with this control.	Optional	
imageclosed	Image of a tree node that has subnodes and that is currently not showing its nodes. The image either is defined statically by this property or also may be defined dynamically - see the corresponding properties defined with this control.	Optional	
imageendnode	Image of a tree node that is an end node (leaf node). The image either is defined statically by this property or also may be defined dynamically - see the corresponding properties defined with this control.	Optional	
treestyle	Style (following cascading style sheet definitions) that is directly passed to the background area of the client tree. You can manipulate e.g. the colour of the tree’s background. The style can also be set dynamically by specifying the property TREESTYLEPROP.	Optional	
selectionstylevariant	Some controls offer the possibility to define style variants. By this style variant you can address different styles inside your style sheet definition file (.css). If not defined "normal" styles are chosen, if defined (e.g. "VAR1") then other style definitions (xxxVAR1xxx) are chosen. Purpose: you can set up style variants in the style sheet defintion and use them multiple times by addressing them via the "stylevariant" property. CIS currently offerst two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!	Optional	VAR1 VAR2

hscroll	Definition of the horizontal scrollbar’s appearance. You can define that the scrollbars only are shown if the content is exceeding the control’s area ("auto"). Or scrollbars can be shown always ("scroll"). Or scrollbars are never shown - and the content is cut ("hidden"). Default is "auto".	Optional	auto scroll hidden
pixelshift	Number of pixels that each hierarchy level is indented. If not defined then a standard is used.	Optional	1 2 3 int-value
pixelshiftendnode	Number of pixels that end nodes are indented. If not defined then a standard is used.	Optional	1 2 3 int-value
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767
withleftpadding	Flag that indicates if the control has a 10 pixel padding on left side. Default is true.	Optional	true false
Binding			
treecollectionprop	(already explained above)		
dynamicloading	If set to "true" then you indicate to the tree control that not all tree information may be loaded when initializing the tree (i.e. the tree collection on server side). As consequence the tree control will pass the "toggle-event" to the server - in case the subnodes of a certain nodes are not yet loaded. In the case the toggle event is passed to the server, the method onToggle() is called inside the tree item.	Optional	true false

imageopenedprop	Property of the tree item object that provides for the image URL which is shown for opened tree nodes or end tree nodes. The value may be different from tree node to tree node -each tree node may have an own image.	Optional	
imageclosedprop	Property of the tree item object that provides for the image URL which is shown for closed tree nodes. The value may be different from tree node to tree node -each tree node may have an own image.	Optional	
treestyleprop	Property of the adapter object that dynamically provides for a style value that is passed to the control's area (background of the client tree). You can as consequence e.g. define the background-colour of the tree dependent on your server side logic.	Optional	
treeclassprop	Name of adapter property that passes back the name of a style sheet class that is taken to render the client tree's background area. - Similar to the property TREESTYLEPROP, but now a style class is passed, not the style itself.	Optional	
tooltipprop	Name of property of the item objects that provides for a text that is shown if the user moves the mouse over the tree item (tooltip).	Optional	
oncontextmenumethod	Name of the method on adapter level that is called if the user presses the right mouse button in an empty area of the client tree. Note: if the user presses right mouse button on a node then the method "reactOnContextMenuRequest()" is called in the item object.	Optional	
directselectevent	Event that represents a tree node selection. A tree node selection is done when the user clicks/doubleclicks on the tree node text. In this case the select() method is called in the corresponding node object on server side.	Optional	ondblclick onclick
focusedprop	Name of property of the item objects - representing the individual rows of the collection - that indicates if the row receives the keyboard focus. Must be of type "boolean"/ "Boolean". If more than one lines are returning "true" the first of them is receiving the focus.	Optional	
Drag and Drop			
enabledrag	If set to true then drag and drop is enabled within the tree.	Optional	true false