

# DB Tree View

## Known Subclasses

[DB Tree Table View](#)

## Return Types

[Binding](#) [RelationInfo](#)

## Property Summary

<a href="#">String</a>	<a href="#">bgcolor</a> Property for getting/setting the background color of the bean.
<a href="#">Boolean</a>	<a href="#">enabled</a> Property for getting/setting the enabled state of the bean
<a href="#">String</a>	<a href="#">fgcolor</a> Property for getting/setting the foreground color of the bean.
<a href="#">Object[]</a>	<a href="#">selectionPath</a> Get/Set the selection (path), array with pk records values (only single pk key supported)
<a href="#">String</a>	<a href="#">toolTipText</a> Property for getting/setting the text of the bean's tooltip
<a href="#">Boolean</a>	<a href="#">transparent</a> Property for getting/setting the transparent or opaque state of the bean
<a href="#">Boolean</a>	<a href="#">visible</a> Property for getting/setting the visibility of the bean.

## Method Summary

<a href="#">Number</a>	<a href="#">addRoots(foundSet)</a> Add foundset to the list of foundsets used to create the tree's root nodes.
<a href="#">Binding</a>	<a href="#">createBinding(args)</a> Create and add binding object for a database table used to set data bindings for nodes.
<a href="#">RelationInfo</a>	<a href="#">createRelationInfo()</a> Create relation info object used to set multiple child relations for a tree node
<a href="#">Number</a>	<a href="#">getHeight()</a> Returns the height of the tree.
<a href="#">Number</a>	<a href="#">getLocationX()</a> Get the x coordinate of the location of the tree.
<a href="#">Number</a>	<a href="#">getLocationY()</a> Get the y coordinate of the location of the tree.
<a href="#">String</a>	<a href="#">getName()</a> Returns the name of the tree.
<a href="#">Number</a>	<a href="#">getWidth()</a> Returns the width of the tree.
<a href="#">Boolean</a>	<a href="#">isNodeExpanded(nodePath)</a> Check the path (array with pk records values (only single pk key supported)) expanded status
<a href="#">void</a>	<a href="#">refresh()</a> Refresh the tree display
<a href="#">void</a>	<a href="#">removeAllRoots()</a> Remove all root foundsets
<a href="#">void</a>	<a href="#">setExpandNode(nodePath, expand_collapse)</a> Expand/collapse the path, array with pk records values (only single pk key supported)
<a href="#">void</a>	<a href="#">setFont(font)</a> Sets the specified font as the font of the tree.
<a href="#">void</a>	<a href="#">setLocation(x, y)</a> Sets the location of the tree.
<a href="#">void</a>	<a href="#">setNodeLevelVisible(level, visible)</a> Set the level of visible nodes (expand or collapse to certain level)
<a href="#">void</a>	<a href="#">setOnDrag(fOnDrag)</a> Set method to be called when a drag is started on the tree.

void	<a href="#">setOnDragEnd</a> (fOnDragEnd) Set method to be called when a drag of on the tree is ended.
void	<a href="#">setOnDragOver</a> (fOnDragOver) Set method to be called during a drag over the tree.
void	<a href="#">setOnDrop</a> (fOnDrop) Set method to be called on a drop on the tree.
void	<a href="#">setRowHeight</a> (height) Set row height
void	<a href="#">setSize</a> (w, h) Sets the size of the tree.

## Property Details

### bgcolor

Property for getting/setting the background color of the bean.

#### Returns

[String](#) - the background color

#### Sample

```
elements.customer_id.bgcolor = "#FFFFFF";
var x = elements.customer_id.bgcolor;
```

### enabled

Property for getting/setting the enabled state of the bean

#### Returns

[Boolean](#) - the enabled state of the bean

#### Sample

```
var b = elements.customer_id.enabled;
if (b) elements.customer_id.enabled = false;
```

### fgcolor

Property for getting/setting the foreground color of the bean.

#### Returns

[String](#)

#### Sample

```
elements.customer_id.fgcolor = "#000000";
var y = elements.customer_id.fgcolor;
```

### selectionPath

Get/Set the selection (path), array with pk records values (only single pk key supported)

#### Returns

[Object\[\]](#)

#### Sample

```
elements.customer_id.selectionPath = new Array(14,24,45,67);
var currentSelectionArray = elements.customer_id.selectionPath;
```

### toolTipText

Property for getting/setting the text of the bean's tooltip

#### Returns

[String](#) - the tooltip text of the bean

**Sample**

```
elements.customer_id.toolTipText = 'Hello world!';
```

**transparent**

Property for getting/setting the transparent or opaque state of the bean

**Returns**

[Boolean](#) - the transparent state of the bean

**Sample**

```
var t = elements.customer_id.transparent;
if (!t) elements.customer_id.transparent = true;
```

**visible**

Property for getting/setting the visibility of the bean.

**Returns**

[Boolean](#) - the visibility of the bean

**Sample**

```
var v = elements.customer_id.visible;
```

**Method Details****addRoots**

[Number](#) **addRoots** (foundSet)

Add foundset to the list of foundsets used to create the tree's root nodes.\nNote: the bean will use a clone of the foundset, so any changes on the foundset parameter will be ignored in the tree.

**Parameters**

[Object](#) foundSet

**Returns**

[Number](#) - The number of added root nodes

**Sample**

```
var addedRootNodes = elements.customer_id.addRoots(foundset);
```

**createBinding**

[Binding](#) **createBinding** (args)

Create and add binding object for a database table used to set data bindings for nodes.

**Parameters**

[String...](#) args

**Returns**

[Binding](#) - Binding object for a database table

**Sample**

```
var companies_binding = elements.customer_id.createBinding('example_data', 'companies');
companies_binding.setTextDataprovider('company_name');
companies_binding.setNRelationName('companies_to_companies');
companies_binding.setImageURLDataprovider('type_icon');
companies_binding.setChildSortDataprovider('company_sort');
```

**createRelationInfo**

**RelationInfo** **createRelationInfo ()**

Create relation info object used to set multiple child relations for a tree node

**Returns**

**RelationInfo** - RelationInfo object

**Sample**

```
var company_relations = new Array();
company_relations[0] = elements.customer_id.createRelationInfo();
company_relations[0].setLabel('Employees');
company_relations[0].setNRelationName('companies_to_employees');
company_relations[1] = elements.customer_id.createRelationInfo();
company_relations[1].setLabel('Customers');
company_relations[1].setNRelationName('companies_to_customers');
companies_binding.setNRelationInfos(company_relations);
```

**getHeight****Number** **getHeight ()**

Returns the height of the tree.

**Returns**

**Number** - The height

**Sample**

```
elements.customer_id.getHeight();
```

**getLocationX****Number** **getLocationX ()**

Get the x coordinate of the location of the tree.

**Returns**

**Number** - The x coordinate

**Sample**

```
elements.customer_id.getLocationX();
```

**getLocationY****Number** **getLocationY ()**

Get the y coordinate of the location of the tree.

**Returns**

**Number** - The y coordinate

**Sample**

```
elements.customer_id.getLocationY();
```

**getName****String** **getName ()**

Returns the name of the tree.

**Returns**

**String**

**Sample**

```
elements.customer_id.getName();
```

**getWidth****Number** **getWidth ()**

Returns the width of the tree.

**Returns**

[Number](#) - The width

**Sample**

```
elements.customer_id.getWidth();
```

**isNodeExpanded**

[Boolean](#) **isNodeExpanded** (nodePath)

Check the path (array with pk records values (only single pk key supported)) expanded status

**Parameters**

[Object\[\]](#) nodePath

**Returns**

[Boolean](#) - True if the node is expanded, False otherwise

**Sample**

```
var pathArray = new Array(14,24,45,67);
elements.customer_id.isNodeExpanded(pathArray);
```

**refresh**

void **refresh** ()

Refresh the tree display

**Returns**

void

**Sample**

```
elements.customer_id.refresh();
```

**removeAllRoots**

void **removeAllRoots** ()

Remove all root foundsets

**Returns**

void

**Sample**

```
elements.customer_id.removeAllRoots();
```

**setExpandNode**

void **setExpandNode** (nodePath, expand\_collapse)

Expand/collapse the path, array with pk records values (only single pk key supported)

**Parameters**

[Object\[\]](#) nodePath

[Boolean](#) expand\_collapse

**Returns**

void

**Sample**

```
var pathArray = new Array(14,24,45,67);
elements.customer_id.setExpandNode(pathArray, true);
```

**setFont**

void **setFont** (font)

Sets the specified font as the font of the tree.

---

**Parameters**

{[String](#)} font

**Returns**

void

**Sample**

```
elements.customer_id.setFont('Times New Roman, 1, 22');
```

**setLocation**

void **setLocation** (x, y)

Sets the location of the tree.

**Parameters**

{[Number](#)} x

{[Number](#)} y

**Returns**

void

**Sample**

```
elements.customer_id.setLocation(120,80);
```

**setNodeLevelVisible**

void **setNodeLevelVisible** (level, visible)

Set the level of visible nodes (expand or collapse to certain level)

**Parameters**

{[Number](#)} level

{[Boolean](#)} visible

**Returns**

void

**Sample**

```
elements.customer_id.setNodeLevelVisible(scopes.globals.g_treeview_level, (scopes.globals.g_treeview_expand == 1 ? true : false));
```

**setOnDrag**

void **setOnDrag** (fOnDrag)

Set method to be called when a drag is started on the tree. For more details about the method arguments and return value check the same property of a form

**Parameters**

{[Function](#)} fOnDrag

**Returns**

void

**Sample**

```
elements.customer_id.setOnDrag(onDrag);
```

**setOnDragEnd**

void **setOnDragEnd** (fOnDragEnd)

Set method to be called when a drag of on the tree is ended. For more details about the method arguments and return value check the same property of a form

**Parameters**

{[Function](#)} fOnDragEnd

**Returns**

void

---

**Sample**

```
elements.customer_id.setOnDragEnd(onDragEnd);
```

**setOnDragOver**

void **setOnDragOver** (fOnDragOver)

Set method to be called during a drag over the tree. For more details about the method arguments and return value check the same property of a form

**Parameters**

{Function} fOnDragOver

**Returns**

void

**Sample**

```
elements.customer_id.setOnDragOver(onDragOver);
```

**setOnDrop**

void **setOnDrop** (fOnDrop)

Set method to be called on a drop on the tree. For more details about the method arguments and return value check the same property of a form

**Parameters**

{Function} fOnDrop

**Returns**

void

**Sample**

```
elements.customer_id.setOnDrop(onDrop);
```

**setRowHeight**

void **setRowHeight** (height)

Set row height

**Parameters**

{Number} height

**Returns**

void

**Sample**

```
elements.customer_id.setRowHeight(40);
```

**setSize**

void **setSize** (w, h)

Sets the size of the tree.

**Parameters**

{Number} w

{Number} h

**Returns**

void

**Sample**

```
elements.customer_id.setSize(400,300);
```