window

Return Types

CheckBox Menu MenuBar MenuItem Popup RadioButton ToolBar

Method Summary

ToolBar	#addToolBar(name, [displayname], [row]) Add a toolbar by name and optional displayname and row.
Popup	#createPopupMenu()
Boolean	#createShortcut(shortcut, method, [form_name]) Create a shortcut.
MenuBar	#getMenuBar([windowName]) Get the menubar of a window.
ToolBar	#getToolBar(name) Get the toolbar from the toolbar panel by name.
String[]	#getToolbarNames() Get all toolbar names from the toolbar panel.
void	#maximize([windowName]) Maximize the current window or the window with the name provided (Smart client only)
Boolean	#removeShortcut(shortcut, [form_name]) Remove a shortcut.
void	#removeToolBar(name) Remove the toolbar from the toolbar panel.
void	#setFullScreen()
void	#setStatusBarVisible()
void	#setToolBarAreaVisible()

Method Details

addToolBar

ToolBar **addToolBar**(name, [displayname], [row])

Add a toolbar by name and optional displayname and row.

Parameters

name
[displayname]
[row]

Returns

ToolBar

Sample

```
// Note: method addToolBar only works in the smart client.

// Add a toolbar by name and optional displayname and row.
// add a toolbar with only a name
var toolbar = plugins.window.addToolBar("toolbar_0");

// add a toolbar with a name and internal name
// var toolbar = plugins.window.addToolBar("toolbar_1", "toolbar_1");

// add a toolbar with a name, internal name and the row you want the
// toolbar to show at. rownumber starts at 0
// var toolbar = plugins.window.addToolBar("toolbar_2", "toolbar_2", 3);

// REMARK: normally you would add buttons, checkboxes etc in the same method
// this example will show no buttons for now!
// we will add them via the other methods on this form.
```

createPopupMenu

Popup **createPopupMenu**()

Replace with description

Returns

Popup

Sample

```
var popupmenu = plugins.window.createPopupMenu()

var menuitem1 = popupmenu.addMenuItem('A',myMethod)
var menuitem2 = popupmenu.addRadioButton('B',myMethod)
var menuitem3 = popupmenu.addRadioButton('C',myMethod)
var menuitem4 = popupmenu.addSeparator()
var menuitem5 = popupmenu.addMenuItem('<html><b>Hello</b></html>',myMethod)
var menuitem6 = popupmenu.addMenuItem('G', globals.myGlobalMethod)
//add arguments to the method call
menuitem6.methodArguments = ['arg1', 'another argument']

var submenu = popupmenu.addMenu('SubMenu')
var subitem1 = submenu.addCheckBox('i18n:bla_bla',myMethod)
var subitem2 = submenu.addCheckBox('he' , globals.myOtherGlobalMethod , 'media:///day_obj.gif')
var subitem3 = submenu.addCheckBox('more' , globals.myOtherGlobalMethod ,null, 'm') //last parameter is
mnemonic-key

menuitem2.selected = true;
menuitem6.enabled = false
subitem2.selected = true;

var source = event.getSource()
if (source != null)
{
        popupmenu.show(source);
        //or you can set the coordinates popupmenu.show(10, 10);
}
```

createShortcut

Boolean **createShortcut**(shortcut, method, [form_name])

Create a shortcut.

Parameters

shortcut
method
[form_name]

Returns

Boolean

Sample

```
// Create a shortcut.
// this plugin uses the java keystroke parser
// see http://java.sun.com/j2se/1.5.0/docs/api/javax/swing/KeyStroke.html#getKeyStroke(java.lang.String)

// global shortcut (on all forms) on  'apple 1' on a mac client and 'control 1' on other client platforms
plugins.window.createShortcut('menu 1', 'globals.handleShortcut');
// global handler, only triggered when on form frm_orders
plugins.window.createShortcut('control shift I', globals.handleOrdersShortcut, 'frm_orders');
// form method called when shortcut is used
plugins.window.createShortcut('control LEFT', 'frm_products.handleShortcut', 'frm_products');
// same, but use method in stead of string
plugins.window.createShortcut('control RIGHT', forms.frm_contacts.handleMyShortcut, 'frm_contacts');

// remove global shortcut and form-level shortcut
plugins.window.createShortcut('menu 1');
plugins.window.removeShortcut('control RIGHT', 'frm_contacts');

// shortcut handlers are called with an jsevent argument
///**
//* Handle keyboard shortcut.
//*
//* @param {JSEvent} event the event that triggered the action
//*/
//function handleShortcut(event)
//{
//  application.output(event.getType()) // returns 'menu 1'
//  application.output(event.getFormName()) // returns 'frm_contacts'
//  application.output(event.getElementName()) // returns 'contact_name_field' or null when no element is
selected
//}

// NOTE: shortcuts will not override existing operating system or browser shortcuts,
// choose your shortcuts careful to make sure they work in all clients.
```

getMenuBar

MenuBar **getMenuBar**([windowName])

Get the menubar of a window.

Parameters

[windowName]

Returns

MenuBar

Sample

```
// Get the menubar of a window.
// get the menubar of the main window
var mainMenubar = plugins.window.getMenuBar();

// get the menubar of a named window
application.showFormInWindow(forms.contacts,100,80,500,300,'my own window title',false,true,'mywindow');
var myWindowMenubar = plugins.window.getMenuBar('mywindow');
```

getToolBar

ToolBar **getToolBar**(name)

Get the toolbar from the toolbar panel by name.

Parameters

name

Returns

ToolBar

Sample

```
// Note: method getToolBar only works in the smart client.

// Get the toolbar from the toolbar panel by name.
// get the toolbar at the panel by name
var toolbar = plugins.window.getToolBar("toolbar_0");

// add a button with a text and a method
toolbar.addButton("button", feedback_button);

// add an input array to the button for feedback in the selected method
toolbar.addButton("button", feedback_button, [1, "2", "three"]);

// add an icon to the button
toolbar.addButton("button", feedback_button, [1, "2", "three"], "media:///yourimage.gif");

// add a tooltip to the button
toolbar.addButton("button", feedback_button, [1, "2", "three"], "media:///yourimage.gif", "tooltip.");

// show only an icon on the button and disable the button
toolbar.addButton(null, feedback_button, [1, "2", "three"], "media:///yourimage.gif", "tooltip.", false);

// add a separator
toolbar.addSeparator();

// make the button non visible
toolbar.addButton(null, feedback_button, [1, "2", "three"], "media:///yourimage.gif", "tooltip.",true, false);

// and validate the changes
// to make them know to the user interface)
toolbar.validate();
```

getToolbarNames

String[] **getToolbarNames**()

Get all toolbar names from the toolbar panel.

Returns

String[]

Sample

```
// Note: method getToolbarNames only works in the smart client.

// Get all toolbar names from the toolbar panel.
// create an array of toolbar names
var names = plugins.window.getToolbarNames();

// create an empty message variable
var message = "";

// loop through the array
for (var i = 0 ; i < names.length ; i++) {
        //add the name(s) to the message
        message += names[i] + "\n";
}

// show the message
plugins.dialogs.showInfoDialog("toolbar names", message);
```

maximize

void **maximize**([windowName])

Maximize the current window or the window with the name provided (Smart client only)

Parameters

[windowName]

Returns

void

Sample

```
// Maximize the current window or the window with the name provided (Smart client only)
// maximize the main window:
plugins.window.maximize();
// or a window constructed with the name 'test':
plugins.window.maximize('test');
```

removeShortcut

Boolean **removeShortcut**(shortcut, [form_name])

Remove a shortcut.

Parameters

shortcut

[form_name]

Returns

Boolean

Sample

```
// Remove a shortcut.
// this plugin uses the java keystroke parser
// see http://java.sun.com/j2se/1.5.0/docs/api/javax/swing/KeyStroke.html#getKeyStroke(java.lang.String)

// global shortcut (on all forms) on  'apple 1' on a mac client and 'control 1' on other client platforms
plugins.window.createShortcut('menu 1', 'globals.handleShortcut');
// global handler, only triggered when on form frm_orders
plugins.window.createShortcut('control shift I', globals.handleOrdersShortcut, 'frm_orders');
// form method called when shortcut is used
plugins.window.createShortcut('control LEFT', 'frm_products.handleShortcut', 'frm_products');
// same, but use method in stead of string
plugins.window.createShortcut('control RIGHT', forms.frm_contacts.handleMyShortcut, 'frm_contacts');

// remove global shortcut and form-level shortcut
plugins.window.createShortcut('menu 1');
plugins.window.removeShortcut('control RIGHT', 'frm_contacts');

// shortcut handlers are called with an jsevent argument
///**
//* Handle keyboard shortcut.
//*
//* @param {JSEvent} event the event that triggered the action
//*/
//function handleShortcut(event)
//{
//  application.output(event.getType()) // returns 'menu 1'
//  application.output(event.getFormName()) // returns 'frm_contacts'
//  application.output(event.getElementName()) // returns 'contact_name_field' or null when no element is
selected
//}

// NOTE: shortcuts will not override existing operating system or browser shortcuts,
// choose your shortcuts careful to make sure they work in all clients.
```

removeToolBar

void **removeToolBar**(name)

Remove the toolbar from the toolbar panel.

Parameters

name

Returns

void

Sample

```
// Note: method removeToolBar only works in the smart client.

// Remove the toolbar from the toolbar panel.
// removing a toolbar from the toolbar panel is done by name
// the plugin checks the existence of the toolbar
// when the toolbar does not exist it will not throw an error though.
plugins.window.removeToolBar("toolbar_0");
plugins.window.removeToolBar("toolbar_1");
plugins.window.removeToolBar("toolbar_2");
```

setFullScreen

void **setFullScreen**()

Replace with description

Returns

void

Sample

```
plugins.window.setFullScreen(true)
```

setStatusBarVisible

void **setStatusBarVisible**()

Replace with description

Returns

void

Sample

```
plugins.window.setStatusBarVisible()
```

setToolBarAreaVisible

void **setToolBarAreaVisible**()

Replace with description

Returns

void

Sample

```
plugins.window.setToolBarAreaVisible(false)
```