ogTools
Generics

Component's user data sheet

- colors
  - 4D and MD spaces
- io
  - alert, confirm, more, notification, progress
- json
  - editor for ogTools parameters
- picker
  - date, date month, date week, day, period, search, time, year
- rc
  - record choice manager
- window
  - window placement manager

from v16

Version	Date	Rédacteur	Commentaires
Initial	1/12/2018	OG	first version of generics
R1	4/05/2019	OG	some methods updated
R2	6/17/2019	OG	Licence and agreement modification

Protée
IT, app design

Protée sari
71, rue Adolphe Coll
31300 TOULOUSE

Tél : 0970 46 56 46
Mob : 0660 49 33 23
www.protee.org / info@protee.org

Overwiew

Introduction

The ogTools component is designed in an extremely regular and generic way. In this sense, most methods can be used for all widgets and pickers : getters, setters are the same. Here you will find all the generic methods for using this component.

Objects

The use of objects allows the use of plenty parameters for widgets, pickers and managers.

- Widget: the basic flat graphical module, and the last end point. Example a calendar.
- Picker: an object which will open a form with a widget for modification. Example a date picker.
- Manager : some functionalities without ant interface. Example window manager.

In addition we introduce the notion, like in java, of « overload » which provides a very powerful way to change the parameters. This object given technology by 4D makes possible to design a totally generic component, with no to use of any process variable, for a benefice in memory.

- Each widget and picker has a general interprocess parameter object. This interprocess object is of course modifiable, either directly or by overload.
- Each created instance copies and works with a new duplicate set of settings.
- The instance object is obviously modifiable to have as instances as different behaviors.
- It is possible to send a partial object to the instance, and by the « overload » mechanism, the settings will only be updated with the partial object sent, others remained unchanged.

Layout

The widgets and pickers are all drawable in the form according to your wishes. The content will be fully resized to the area you have chosen for your instance.

- Choice of size according to your needs
- Multi-instance

Objects options

Description

The component use two interprocess objects, one for general properties, one for all widgets and pickers options, regrouped by modules.

Access to	Method	Contents
general	wog__getGeneral	current_year, fiscal_month, rwal[4]
modules	wog__getModules	version, colors, io, json, picker, rc, window

Modules contains all modules, and then all options (widgets and pickers), then properties.
Some options properties are common to almost all options. They are listed here:

Property	Type	Contents
is_autoAccept	boolean	For a form, when closing it with the close button, or in case of popup window clicking outside, the behavior is like clicking on « ok » button. Else, you will be prompted with an « save - don't save - cancel » dialog.
is_okCancel	boolean	Display « Ok » and « Cancel » button.
is_onMouseUp	boolean	For widget, the change to data is effective only on mouse up. Else, you will get a continuous update while moving elements on widgets.
ptr_external	pointer	Pointer to data. If not nil, this will be used in conjonction with the internal stored data.
ptr_external1	pointer	Pointer to data1 in case of double data (dates, period…)
state_rwal	longint	version, colors, io, json, picker, rc, window
window_type	longint	Window type used for form opened in pickers. 0 Plain form window - 1 Pop up form window - 2 Sheet form window - 3 Palette form window

Methods

Component

wog_init

wog_init			
Do not require any parameter			

Description

The wog_init command initialize the og_tools component. You have to call this method before using any of the other methods of ogTools.

wog_initSerial

wog_initSerial ({owner} {;serial}) -> isOk			
Parameter	**Type**		**Description**
owner	txt	→	owner of this ogTools licence
serial	txt	→	Serial number attached to owner
isOk	boolean	↩	True if given serial is ok, False otherwise

Description

The wog_initSerial command initialize the serial of your og_tools component.
You need at least to use this call to go in trial mode :

- your full unrestricted time is one hour.
- after one hour, you go in restricted mode.
- In this mode, you can still use ogTools, but you will be prompted with a menu asking you for a very simple answer.

wog__getVersion

wog_getVersion -> $ob			
Parameter	**Type**		**Description**
$version	object	↩	object with version informations

Description

The wog__getVersion command gives you back the version information about ogTools, in $ob.

ob		
property	**Type**	**Description**
product	text	Product name. Here « ogTools ».
version	text	In the form « x.y.zz »
notes	text	Text of release 's notes.

wog__getLicence

wog_getVersion ({$LG}) -> $txt			
Parameter	**Type**		**Description**
$LG	text	→	Language. If omitted, or unknow, FR version is returned. Actually, only EN and FR are allowed.
$txt	text	↩	Text about the licence

Description

The wog__getLicence command gives you back the licence information about ogTools.

You need to to agree all the terms of this licence. If not, you need to :

- Immediately notify us and return the license, along with your proof of purchase, to obtain a refund of your invoice.
- Delete all elements of the Software on your computer (s), depending on whether you are using the Single User Version or the Client / Server Version of the Software.

General

wog__getModules

wog__getModules -> ob_modules			
Parameter	**Type**		**Description**
ob_modules	object	↩	Object containing all modules, widgets & pickers & managers, properties

Description

The wog__getModules command gives in return the object with all modules and widget & pickers & managers, then options. To give a quick access to each module, methods are given listed below:

Method	Direct access to module
wog__getModule_color	color
wog__getModule_io	io
wog__getModule_json	json
wog__getModule_picker	picker
wog__getModule_rc	rc
wog__getModule_window	window

As the same, methods are given to give quick access to widgets, pickers, functions:

Method
wog__getOptions_alert, wog__getOptions_colorIdx, wog__getOptions_colorIdx_displa, wog__getOptions_colorIdy, wog__getOptions_colorIdy_displa, wog__getOptions_colors, wog__getOptions_colors_display, wog__getOptions_confirm, wog__getOptions_date, wog__getOptions_date_display, wog__getOptions_day, wog__getOptions_iog, wog__getOptions_json, wog__getOptions_json_display, wog__getOptions_json_modules, wog__getOptions_more, wog__getOptions_notif, wog__getOptions_period, wog__getOptions_period_display, wog__getOptions_progress, wog__getOptions_progress_type, wog__getOptions_rc, wog__getOptions_rc_display, wog__getOptions_rotator, wog__getOptions_time, wog__getOptions_time_display, wog__getOptions_window, wog__getOptions_year, wog__getOptions_yearMonth, wog__getOptions_yearWeek

General parameters

wog__getGeneral

wog__getGeneral -> ob_general			
Parameter	**Type**		**Description**
ob_general	object	↩	Object containing general parameters

Description

The wog__getGeneral command in return the object with the general parameters.

wog__get_colors_RWAL

wog__get_colors_RWAL (ptr_array)			
Parameter	**Type**		**Description**
ptr_array	ptr	↩	Pointer to an array. Set back to 4 items.

Description

The wog__get_colors_RWAL command will copy the pointer array with the colors in ogTools.

wog__get_current_year

wog__get_current_year -> year			
Parameter	**Type**		**Description**
year	longint	↩	Current year

Description

The wog__get_current_year command gives in return the current year stored is ogTools.

wog__get_fiscal_month

wog__get_fiscal_month -> month			
Parameter	**Type**		**Description**
isOk	boolean	↩	Fiscal month (1 to 12)

Description

The wog__get_fiscal_month command gives in return the fiscal month stored is ogTools.

wog__set_colors_RWAL

wog__set_colors_RWAL (ptr_array)			
Parameter	**Type**		**Description**
ptr_array	Pointer	→	Pointer to array longint, 4 items

Description

The wog__set_colors_RWAL command gives to ogTools an array with the colors to be used for consultation, modification, creation, linked. This is mandatory to give a 4 lines array !

wog__set_current_year

wog__set_current_year (year)			
Parameter	**Type**		**Description**
year	longint	→	Current year

Description

The wog__set_current_year command gives to ogTools the current year.

wog__set_fiscal_month

wog__set_fiscal_month (month)			
Parameter	**Type**		**Description**
Month	longint	→	Fiscal month

Description

The wog__set_current_month command gives to ogTools the fiscal month.

Overload objects

wog__ob_overload

wog__ob_overload (ob_source ; ob_target)			
Parameter	**Type**		**Description**
ob_source	object	→	Source object for overloading
ob_target	object	→	Target object to be overloaded

Description

The wog__ob_overload command will change, for each property in source, the same property in target, not recursively.

wog__ob_overload_exist

wog__ob_overload_exist (ob_source ; ob_target)			
Parameter	**Type**		**Description**
ob_source	object	→	Source object for overloading
ob_target	object	→	Target object to be overloaded

Description

The wog__ob_overload_exist command acts as wog__ob_overload method, but wont add any property not existing in target.

wog__ob_overloads

wog__ob_overloads (ob_source ; ob_target)			
Parameter	**Type**		**Description**
ob_source	object	→	Source object for overloading
ob_target	object	→	Target object to be overloaded

Description

The wog__ob_overloads command will change, for each property in source, the same property in target, recursively.

wog__ob_overloads_exist

wog__ob_overloads_exist (ob_source ; ob_target)			
Parameter	**Type**		**Description**
ob_source	object	→	Source object for overloading
ob_target	object	→	Target object to be overloaded

Description

The wog__ob_overloads_exist command acts as wog__ob_overloads method, but wont add any property not existing in target.

wog__ob_update_exists

wog__ob_update_exists (ob_source ; ob_target)			
Parameter	**Type**		**Description**
ob_source	object	→	Source object for overloading
ob_target	object	→	Target object to be overloaded

Description

The wog__ob_update_exists command will update ob_target, recursively, where the « flat » ob_source properties exist. This is useful to modify a specific property everywhere in all widgets.

Overload options and general

wog__setOptions

wog__setOptions (ob_source)			
Parameter	**Type**		**Description**
ob_source	object	→	Source object to overload the main modules object

Description

The wog__setOptions command gives you the ability to overload, recursively, the main object for modules.

wog__setOptions_general

wog__setOptions_general (ob_source)			
Parameter	**Type**		**Description**
ob_source	object	→	Source object to overload the general object

Description

The wog__setOptions command gives you the ability to overload, recursively, the general object.

Instances: data and options getters

All widgets and pickers instances have a name. With this name, you can access all of them individually.
When you are inside the instance, which means inside the method of the instance, and you want to access this own instance, there is no need to give that name, this is done by default.
That is why in all the following methods, the {widget} parameter is optional.

From outside, you can use all those methods but giving this « widget » name to the methods.

To get and set values for widgets and pickers, you will see different methods depending of the type of data expected. This can be longint, text, date, object…
So it is important to use the correct getter and setter regarding the expected type for the widget.

For time value, this is managed with the longint type.

wog_getValue

wog_getValue ({widget}) -> value			
Parameter	**Type**		**Description**
{widget}	text	→	Widget name, or current widget
value	longint	↩	Value stored in widget

Description

The wog_getValue command gives back the value stored in longint format when this is relevant for a widget.

wog_getValues

wog_getValues (ptrValue ; ptrValue1 ; {widget})			
Parameter	**Type**		**Description**
value	ptr	→	Pointer to first value
value1	ptr	→	Pointer to second value
{widget}	text	→	Widget name, or current widget

Description

The wog_getValues command gives back the two values stored in longint format when this is relevant for a widget.

wog_getValue_date

wog_getValue_date ({widget}) -> date			
Parameter	**Type**		**Description**
{widget}	text	→	Widget name, or current widget
date	date	↩	Value stored in widget

Description

The wog_getValue_date command gives back the value stored in date format when this is relevant for a widget.

wog_getValue_dates

wog_getValue_date (ptr_date ; ptr_date1 ; {widget}) -> date			
Parameter	**Type**		**Description**
ptr_date	pointer	→	Pointer to date one
ptr_date1	pointer	→	Pointer to date two
{widget}	text	→	Widget name, or current widget
date	date	↩	Value stored in widget

Description

The wog_getValue_dates command gives back the date values stored in date format when this is relevant for a widget. Dedicated to period widgets.

wog_getValue_ob

wog_getValue_ob ({widget}) -> ob			
Parameter	**Type**		**Description**
{widget}	text	→	Widget name, or current widget
ob	object	↩	Value stored in widget

Description

The wog_getValue_ob command gives back the value stored in object when this is relevant for a widget.

wog_getValue_txt

wog_getValue_txt ({widget}) -> txt			
Parameter	**Type**		**Description**
{widget}	text	→	Widget name, or current widget
txt	text	↩	Value stored in widget

Description

The wog_getValue_txt command gives back the value stored in text when this is relevant for a widget.

wog_getOptions

wog_getOptions ({widget}) -> ob_options			
Parameter	**Type**		**Description**
{widget}	text	→	Widget name, or current widget
ob_options	object	↩	Value stored in widget

Description

The wog_getOptions command gives back the object value stored in options for a widget.

wog_getRWAL

wog_getRWAL ({widget}) -> rwal			
Parameter	**Type**		**Description**
{widget}	text	→	Widget name, or current widget
rwal	longint	↩	Value stored in widget

Description

The wog_getRWAL command gives back the value stored in options for rwal (read, write, add, link).

wog_getEditable

wog_getEditable ({widget}) -> is_editable			
Parameter	**Type**		**Description**
{widget}	text	→	Widget name, or current widget
is_editable	boolean	↩	Value stored in widget

Description

The wog_getEditable command gives back the value stored in options based on rwal (1 or 2 gives true).

Instances: data and options setters

wog_setValue

wog_setValue (value ; {widget})			
Parameter	**Type**		**Description**
value	longint	→	Value to store in widget
{widget}	text	→	Widget name, or current widget

Description

The wog_setValue command gives stores the value in longint format (when this is relevant for a widget).

wog_setValue_date

wog_setValue_date (date ; {widget})			
Parameter	**Type**		**Description**
date	date	→	Value to store in widget
{widget}	text	→	Widget name, or current widget

Description

The wog_setValue_date command stores the value in date format (when this is relevant for a widget).

wog_setValue_dates

wog_setValue_dates (date ; date1 ; {widget})			
Parameter	**Type**		**Description**
date	date	→	Value to store in widget
date1	date	→	Value to store in widget
{widget}	text	→	Widget name, or current widget

Description

The wog_getValue_dates command store the values in date format (when this is relevant for a widget, dedicated to period widgets).

wog_setValue_ob

wog_setValue_ob (ob ; {widget})			
Parameter	**Type**		**Description**
ob	object	→	Value to store in widget
{widget}	text	→	Widget name, or current widget

Description

The wog_getValue_ob command stores the value in object format (when this is relevant for a widget).

wog_setValue_txt

wog_setValue_txt (txt ; {widget})			
Parameter	**Type**		**Description**
txt	text	→	Value to store in widget
{widget}	text	→	Widget name, or current widget

Description

The wog_setValue_txt command stores the value in text format (when this is relevant for a widget).

wog_setOptions

wog_getOptions (ob_options ; {widget})			
Parameter	**Type**		**Description**
ob_options	object	→	Value to store in widget
{widget}	text	→	Widget name, or current widget

Description

The wog_setOptions command stores the value in object format to the options of the instance.

wog_setRWAL

wog_setRWAL (rwal ; {widget})			
Parameter	**Type**		**Description**
rwal	longint	→	Value to store in widget
{widget}	text	→	Widget name, or current widget

Description

The wog_setRWAL command stores the value in longint format for rwal (read, write, add, link).

wog_setEditable

wog_setEditable (is_editable ; {widget})			
Parameter	**Type**		**Description**
is_editable	boolean	→	Value to store in widget
{widget}	text	→	Widget name, or current widget

Description

The wog_setEditable command stores the value in boolean format to rwal (0 or 1).

wog_setValue_ptr

wog_setValue_ptr (ptr ; {widget})			
Parameter	**Type**		**Description**
ptr	pointer	→	pointer to a value provider
{widget}	text	→	Widget name, or current widget

Description

The wog_setValue_ptr command stores a pointer to a value.

wog_setValues_ptr

wog_setValues_ptr (ptr ; ptr1 ; {widget})			
Parameter	**Type**		**Description**
ptr	pointer	→	Pointer to store in widget for value
ptr1	pointer	→	Pointer to store in widget for value1
{widget}	text	→	Widget name, or current widget

Description

The wog_setValue_ptr command stores two pointers to two values.

Instances: miscellaneous

wog_resize

wog_resize ({border} ; {widget})			
Parameter	**Type**		**Description**
{border}	longint	→	Border to apply to widget. By default, « border none » is used. Pass a value <0 to get it unchanged.
{widget}	text	→	Widget name, or current widget

Description

The wog_resize command is very important and as to be used in almost all widgets on event « on load ». This will:

- resize correctly the widget content to the instance size in the form
- remove any used border for layout purpose. As it is easier to see your widget with a border in a layout, don't worry as this method will set it to « border none » when called without parameters.
- in case of a color widget, this will manage the instance name to detect « § » in it. If yes, then the 3 next chars are used to determine the space, the sf mode, and the type of display.
  - « 4 » for 4D, « M » for MD
  - « Y » for double colors - single otherwise
  - « n » value from 0 to 9 for the type of color widget

wog_redraw

wog_redraw ({widget})			
Parameter	**Type**		**Description**
{widget}	text	→	Widget name, or current widget

Description

The wog_redraw command let you redraw the content based on the value stored in the widget, but first outside with the data referred with the pointer if used.
You can use it when you modify yourself the value from outside the widget, to refresh it.

wog_calculateRWAL

wog_calculateRWAL (isModif ; isAdd) -> rwal			
Parameter	**Type**		**Description**
isModif	boolean	→	true if modification mode
isAdd	boolean	→	true if new record
rwal	longint	↩	Value for rwal

Description

The wog_calculateRWAL command gives back the value to use as a rwal indicator for a widget.

wog_calculate_newModif

wog_calculate_newModif (table I field ; ptr_isNew ; ptr_isModif) -> is_new			
Parameter	**Type**		**Description**
table I field	pointer	→	Pointer to a field or a table gives if record is modifiable or add mode.
ptr_isNew	pointeur	→	true if new record
ptr_isModif	pointer	→	true if record not locked and modifiable
is_new	boolean	↩	redondant to allow the use with « if » statement to initialize value

Description

The wog_calculate_newModif command gives back the information on the record (table or field).

wog_calculate_newModif

wog_calculate_newModif (table I field ; ptr_isNew ; ptr_isModif) -> is_new			
Parameter	**Type**		**Description**
table I field	pointer	→	Pointer to a field or a table to determine if record is modifiable or in add mode.
ptr_isNew	pointeur	→	true if new record
ptr_isModif	pointer	→	true if record not locked and modifiable
is_new	boolean	↩	redondant to allow the use with « if » statement to initialize value

Description

The wog_calculate_newModif command gives back the information on the record (table or field).

wog_setRWAL_fromField

wog_setRWAL_fromField (table I field ; {widget}) -> isNew			
Parameter	**Type**		**Description**
table I field	pointer	→	Pointer to a field or a table to determine if record is modifiable or in add mode.
{widget}	text	→	Widget name, or current widget
isNew	boolean	↩	redondant to allow the use with « if » statement to initialize value

Description

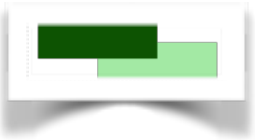
The wog_getEditable command gives back the value stored in options based on rwal (1 or 2 gives true).

Examples and usage

Description

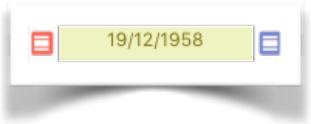
This is a full generic example, where only the initialization in case of a new record will change, according to the type.

Method in the widget (single value, color)



```
C_POINTER($field)
$field:=->[Evénements]colors

C_LONGINT($evt)
$evt:=Form event
Case of
     : ($evt=On Load)
          wog_setOptions (<>ob_colors_txt) // option for the colors widget
          wog_resize (Border Dotted)

          If (wog_setRWAL_fromField ($field))
               // Initialisation in case of a new record
               $field->:=wog_colorsMDrandom
          End if
          wog_setValue_ptr ($field)

End case
```

Method in the widget (single value, date)

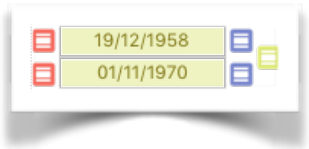



```
C_POINTER($field;$table)
$field:=->[Plans]DateSem

C_LONGINT($evt)
$evt:=Form event
Case of
     : ($evt=-On Load)
          wog_resize
          If (wog_setRWAL_fromField ($field))
               $field->:=vDATE_Calcul (-4;Current date)
          End if
          wog_setValue_ptr ($field)
          v_AnneeSemDF:=F_Plan_weekDisplay_txt ([Plans]DateSem)

     : ($evt=On Data Change)
          v_AnneeSemDF:=F_Plan_weekDisplay_txt ([Plans]DateSem)
          F_PLA_PLL_groutine ("LB_REDRAW")

End case
```

You can add this to catch a change of data

```
: ($evt=On Data Change)
// Do what you need
```

Method in the widget (double values, dates)




```
C_POINTER($ptr_date_start;$ptr_date_end)
$ptr_date_start:=->[Congés]DébutDate
$ptr_date_end:=->[Congés]FinDate

C_LONGINT($evt)
$evt:=Form event
Case of
     : ($evt=On Load)
          wog_resize
          If (wog_setRWAL_fromField ($ptr_date_start))
               $ptr_date_start->:=Current date
               $ptr_date_end->:=$ptr_date_start->+6
          End if
          wog_setValues_ptr ($ptr_date_start;$ptr_date_end)

     : ($evt=On Data Change)
     // Do what you need

End case
```