B4J Form Generator

Disclaimer

This SOFTWARE PRODUCT is provided by El Condor "as is" and "with all faults." El Condor makes no representations or warranties of any kind concerning the safety, suitability, lack of viruses, inaccuracies, typographical errors, or other harmful components of this SOFTWARE PRODUCT. There are inherent dangers in the use of any software, and you are solely responsible for determining whether this SOFTWARE PRODUCT is compatible with your equipment and other software installed on your equipment. You are also solely responsible for the protection of your equipment and backup of your data, and El Condor will not be liable for any damages you may suffer in connection with using, modifying, or distributing this SOFTWARE PRODUCT.

You can use this **SOFTWARE PRODUCT freely, if you would you can credit me in program comment:**

El Condor – CONDOR INFORMATIQUE – Turin

Comments, suggestions and criticisms are welcomed: mail to rossati@libero.it

Conventions

Commands syntax, instructions in programming language and examples are with font `COURIER NEW`. The optional parties of syntactic explanation are contained between `[square parentheses]`, alternatives are separated by `|` and the variable parties are in *`italics`*.

Contents table

1 Form generator

Form generator, briefly *FormGen*, is a class module for *B4J* which allows to build and handle forms; it is sufficiently generalized for a wide use.

1.1 Using the form generator

Before use FormGen the class `fgen` must be instantiate:

`fg.Initialize(`*`MainForm`*`,`*`width`*`,`*`height`*`)`

```
Sub Globals
...
      Dim fg As fgen      ' instantiate Form Generator class
...
End Sub
...
      fg.Initialize(MainForm,500,600)
```

The form is generated by calling the `fg` method:

`instantiatedName``.fg(`*`dataList`*`,`*`callingModule`*`,`*`subHandleAnswer`*`,`*`subHandleEvents`*`)`

where *`dataList`* is a string containing the form components description, *`callingModule`* is a calling code module or class module, *`subHandleAnswer`* and *`subHandleEvents`* are characters string containing the name of the sub for handle data when the form is closed and the possible sub for handle events or an empty string if you wouldn't handle events.

Example:

```
...
Dim parms As String = "Slide,,5,S,3,10 -10;Rdb,Sexe,,10,,M:Male|F:Female;"
...
fg.fg(Parms,Me,"handleAnswerTest","handleEvents")
...
fg.fg(Parms,Me,"handleAnswerTest","")
...
```

1.2 Data description

Every Node (or widget) is characterized by a list of attributes comma separated, in this order: Node *Type*, field *Name*, field *Label*, *Length*, *Default* Value and *Extra*. Nodes are separated by semicolon.

In addition to the Nodes there can be some others information (*Pseudo types*) with different semantics that will be detailed in the paragraphs dedicated to them.

If the Node list starts with `'` it is a comment which ☞ must also be terminated by semicolon.

1.2.1 Node Type

The Types are indifferent to case.

- Buttons:
  - **B** button;
  - **R** or **RDB** radio button, a set of Radio buttons;
- **CKB** check box;
- combo boxs (or Combo boxes):
  - **CMB** combo box;

    - **CMT** is a combo box with Text associated for insert values not in combo box;
    - **CMX** a text field which can be updated with data extracted from a combo box ;
    - **F**, **FILE** file and directory;
    - **WF**, **WFILE** output file;
- Date and Time
    - **DATE**
    - **TIME** ☞ (for now is a qualitative slider giving only the hour)
- Text fields:
    - **COMMENT**, **C** comment;
    - **N** numeric field;
    - **DN** decimal numeric field;
    - **S** slider is an extension of the standard control;
    - **QS** qualitative slider;
    - **P** password field, the data entered are masked;
    - **T** text field is the default if the Type is omitted;
    - **U** not modifiable field i.e. a protected field.

1.2.2 Node Name

Is the name of the field that, when the form is closed, is returned with the associated value; the name is case-sensitive and it is used to access data and possibly handle the Nodes (see 1.3 CallBack).

1.2.3 Node Label

Label of node or caption of button, if omitted it is used the field *Name*.

1.2.4 Length

The length of the Node in characters; the possibly default depends from the *Type*:

	Node type	Value
DATE		13
DN	Decimal numeric text	9
F	File	30
N	Numeric text	7
R	Radio button	12
QS	Qualitative slider	8
S	Slider	5
P	Password	16
T	Text node	maximum from 20 and the length of the possibly default value

1.2.5 Default value

Is the value proposed in form; the form is restored with the defaults values when the `Reset` button is pushed.

1.2.6 Extra(s)

Extra Field is used for add information to the Node.

Node type	
Check Box	a possible description after the check box

Date	the possibly date format, the possibly the second *extra* field is a tool-tip
Radio buttons	an item list separated by |
Slider	the slider limits, the possible second *extra* field is a tool-tip
Qualitative slider	an item list separated by |
combo box	an item list separated by |
Text Fields	if the default field is empty, is the hint, the possible second *extra* field is a tool-tip
Time	the possible second *extra* field is a tool-tip

1.2.7 Pseudo types

Pseudo fields are flavors for show form; they have a type and the syntax is different from the normal Nodes.

1.2.7.1 After

The pseudo field `after` is useful for insert a button or a check box or a radio buttons at right of a Node:

`after,`*`NodeNameA`*`,`*`NodeNameL`*

where *`NodeNameA`* is the Node to be placed at right of the Node *`NodeNameL`*

☞ In the list of widgets *`NodeNameL`* must appears before *`NodeNameL`* (unless it's a button).

```
...
& "N,Age,,5,,age;" _
& "R,Sex,,10,Man,M:Man|F:Female;" _
& "After,Sex,Age;" _
...
```

1.2.7.2 Check

This pseudo field is used for some controls on fields:

`check,`*`fieldName operator (value|fieldName)[,errorMessage]`*

where operator can be one of =, >, <, <>, >=, <= or `is`; after `is` operator can bee:

`mail|`*`regularExpression`*

```
...
& "T,email,My EMail,20;" _
& "P,password,type password;" _
& "P,repeatPassword,retype password;" _
& "check,email is mail,Incorrect mail form;" _
& "check,password=repeatPassword;" _
& "Check,psw is .{6#44},Password too short;" _     ' #44 is comma
...
```

☞ if *`value`* contains comma or semicolon they must be masked, i.e. comma is `#44` and semicolon is `#59`.

1.2.7.3 Comment

The syntax is: `[C|COMMENT],`*`comment`*

1.2.7.4 Defaults

The syntax is:

`default[s],`*`NodeName`*`:`*`NodeValue`*`[,...]`

it is useful for populate the form.

☞ if *`NodeValue`* contains comma or semicolon they must be masked.

1.2.7.5 Random defaults (☞ experimental)

rdefault[s],*NodeName*[:*NodeRandomValue*][,...]

The rdefault pseudo type can be used for testing purpose. *NodeRandomValue* can have the syntax:

- n_1 n_2 a random value from integers n_1 n_2 extremes included, ☞ must be $n_1 < n_2$;
- a list of items separated by `|`.

☞ In case of combo box, qualitative slider and radio buttons only *NodeName* is required.

```
Title,,Random defaults example;
N,Integer,Integer number,7,;
DN,Real,Decimal number,10;
R,Status,,12,,M:Married|S:Single|W:Widow;
QS,Urgency,,9,Green,White|Green|Yellow|Red;
CMB,cmb,Combo box,20,,Alpha|Beta|Delta|Gamma;
S,Slider,Seek 1,6,,10 -10;
RDEFAULT,Integer:-100 +100,Real:17 97,Status,Urgency:Red,cmb,Slider:-10 10
```

1.2.7.6 Background

The pseudo field `Ground` or `Background` create a background in the form or a gradient with a possible effect of transparency; the syntax is:

[`Ground|Background`][,*`fromColor [toColor direction`*]]

The default is a gray color `C0C0C0`, two colors identify a linear gradient whose default value is toward the `bottom`.

- *`fromColor`*, *`toColor`* are colors in hexadecimal notation with two digit for every component: `RRGGBB`, in this case the color is opaque; the transparency is a hexadecimal value from `00` to `FF` appended before the color; it is accepted also a three digit color: `RGB` and a color name, in these cases transparency is ignored;
- *`direction`* is the linear gradient direction: `[left | right] | [top | bottom]`

```
"Ground;"                ' default FFC0C0C0
"Ground,FF;"             ' blue
"Ground,7F0000FF;"       ' half transparent blue
"Ground,FF 8000 top left;"     ' gradient from green to blue on top left
```

1.2.7.7 Menu

Add menu in menu bar:

`Menu,`*`father`*`,`*`son`*`[,[Main.]`*`function`*`]`

The program generate the menu structure based on simple rule i.e only *`son`*s that have a function, (they must not be present as `father`), generates a menu item; the first father is an item of menu bar.
Besides a *`son`* can't have a name of a father unless it is under another menu bar.

The menu at right has been generated by a string of characters containing:

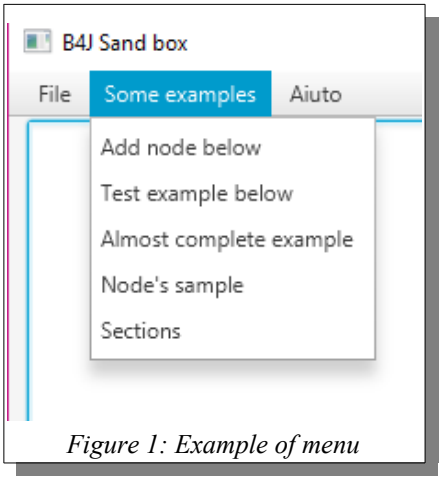

Figure 1: Example of menu

```
Menu,File,End,btnClose_Action
Menu,Some examples,Add node below,Add_Action;
Menu,Some examples,Test example below,Test_Action
Menu,Some examples,Almost complete example,Sample_Action
Menu,Some examples,Node's sample,Widgets_Action
Menu,Some examples,Sections,Sections_Action;
Menu,Aiuto,Help
```

```
Menu,Help,About,About;"
```

1.2.7.8 Required

A list of fields that mus be inserted:

```
required,field1[,field2...]]
```

1.2.7.9 Hidden field

Is the type **H** or **HIDDEN**; the syntax is: `[H|Hidden],fieldName,value` es:

```
DateTime.DateFormat = "yyyy-MM-dd HH:mm:ss"
...
"Hidden,TimeStamp," & DateTime.Date(DateTime.Now) & ";"
```

1.2.7.10 Section

```
section,sectionName[,condition[,condition[,...]]]
```

This pseudo type is for multi-forms, he must precede its nodes.

The *condition* has the form: *fieldName operator* (*value*|*fieldName*) where operator can be one of =, >, <, <>, >=.

Sections are displayed in the order in which they are present, a `Section` whit condition is displayed only if all condition are verified.

In case of multi section are added two navigation button: `fh_Forward` and `fh_Back` with caption respectively `-->` and `<--`; the `Ok` button is present only on the last section.

1.2.7.11 Title

```
Title,name,formTitle[,color[,backgroundColor]]
TITLE,Title,Send mail parameters,FF202020;
```

The default *color* is `black`, the background default of *backgroundColor* is the background color of form.

If *name* is omitted it is set to `fh_title`.

1.2.8 Summary by type

1.2.8.1 Default and extra field

Type	Length	Default field	Extra field
B	Ignored	possibly `dis[able]` or `cancel`	Possibly name of CallBack function
CKB		`1` or `check[ed]` = checked	Possible Description at right of check box
CMB		*value*	An item list separated by `|:` [*key*:]*value*
F		Initial folder	
S		Initial value	Start and end value, default is `0 100`
T,N,DN,P		Initial value	hint, tooltip
U		Not modifiable text	

1.2.8.2 Buttons

The package adds the standard buttons `Ok`, `Cancel` and `Reset`, if there are sections on all forms except the last the `Ok` button has caption `-->` and all forms, except the first, has a button with caption `<--` and name `fh_Back`.

The buttons can be used both for take different actions on closing form both for show user caption instead of default `Ok`, this last can be obtained not only for `Ok` button with this syntax:

```
B,[Ok|Cancel|Reset],caption
```

The value of *default* field `dis[able]` is used to start the form with the button disabled.

The *extra* field contains a name of the function called when a CallBack Button is pushed, in the form: *moduleName.functionName*. or *functionName* if it is in the module which required the creation of the form.

The Ok button is replaced if there is almost one type **B** control in the list not associated, by `AFTER` pseudo type, to some control.

```
...
Dim fg As fgen      ' instantiate Form Generator class
...
Dim frm as String = "N,n1,Integer,10,;n2,DN,Decimal,10;B,Multiply,,10,,Main.CallBack"
fg.fg(Activity,frm,"Main.handleAnswerTest","")
...
Sub CallBack(btnName As String)   ' CallBack event handler
      Dim n1 As Float = fg.iIF(IsNumber(fg.valueOf("n1")),fg.valueOf("n1"),0)
      Dim n2 As Float = fg.iIF(IsNumber(fg.valueOf("n2")),fg.valueOf("n2"),0)
      Msgbox(n1*n2,btnName)
End Sub
...
```

The label of button can be a Unicode character which is a simple and efficient way to create buttons with pictures: the Unicode characters is in the form `#nnnn` where *nnnn* is a decimal value of the Unicode character.

The button can also have an image that must be in the assets folder; the name of image is in the label field and are accepted `png`, `jpg`, `jpeg`, `gif` and `ico` images.

```
B,Cancel,#10008;
B,Reset,#8630;
B,Start,#9998,,myHandler,Go;
```

Table 1: Some UNICODE characters

Name	Decimal value	Symbol	Hexadecimal value
edit	#9998	✎	#x270E
delete	#10008	✘	#x2718
check	#10003	✓	#x2713
check bold	#10004	✔	#x2714
email	#9993	✉	#x2709
cross	#10006	✖	#x2716
	#8630	↶	#x21b6
euro		€	#x20AC
pound		£	#xA3
white square		▢	#x25a2

1.2.8.3 Check box

For checked box insert into **default** field `check[ed]` or `1`.
The **extra** field can contain a possibly description at right of the check box.
The value returned of check box is a string containing 0 or 1, they must be compared as string:

```
cmd.Append(fg.iIF(fh_Data.Get("Mandatory") = "1","M",""))
```

1.2.8.4 Date

The possibly *extra* field is the date format; the possibly second *extra* field is the ToolTip.

1.2.8.5 Time

☞ (for now is a qualitative slider giving only the hour)

1.2.8.6 File

The first *extra* field can contains a hint that is used also as description of the file type, the subsequent extras contain file extensions:

```
WF,File,Image File,30,,Image,*.jpg,*.png;
F,psFile,PDF and PS files,50,,PDF and PS files,*.pdf,*.ps;
```

1.2.8.7 Radio buttons

It is possible to have more than one set of radio buttons.

The **length** is the length of the single Node; the **extra** field contains the item list separated by `|`. For get a key instead the description, the item must have the form: `key:value`.

The **default** value can be the data showed or the key:

```
Rdb,Status,10,,Single,M:Married|S:Single|W:Widow;
Rdb,AgeType,10,,Y,M:Months|Y:Years;
```

1.2.8.8 Slider

The *length* is the length of the text which shows the slider value.

The *extra* field of the type **S** can contains the start and end values in the form `start end`, e.g. `-5 5`; the range is `0 100` if omitted, if only one value is present, the default value for the second is `100`; the result can have decimals depending on the difference from `start` and `end` value, see table at right.

abs(start - end)	n. decimals
> 99	0
<100 and > 10	1
<10 and > 1	2
<1 and > 0.1	3
...	...

`start` can be greater of `end` e.g.:

```
S,Slider,,5,-3,10 -10
```

☞If there isn't a default value and the slide is not moved the value returned is empty.

The qualitative slider (type **QS**) returns qualitative values taken from the *extra* field where they have the same syntax of the values of radio buttons:

```
...
QS,Urgency,,8,Green,White|Green|Yellow|Red
...
```

1.2.8.9 Combo box

CMB is a simple combo box, if no member is selected the value returned is an empty string; if the form has only one combo box, there is only a `Cancel` button and the form is exited when a combo box item is selected.

The *extra* field of combo box contain the item list separated by `|` (see description in Radio button).

CMT type is a combo box with text associated for insert a possibly value not present in combo box.

CMX type is a combo box with text, every choice in combo box is recorded in the text, this is useful for example to compile a list of symptoms.

1.2.8.10 Text fields

For text type (**T**, **P**, **N**, **DN**) the possibly *extra* field is the *hint*; the possibly second *extra* field is the ToolTip.

1.2.9 Returned Values

The data are accessible in the Sub indicated as third parameter of the call, which is called when the buttons `Ok` or `Cancel` or the possible type **B** button is pushed.

The sub has a parameter which is the map which contains the data which are accessible via `Get` or `GetDefault` methods: the key is the field name, besides there is the element with key `fh_button` which contains the name of the button pushed (`Ok` or `Cancel` or the name of the button pushed).

☞If `Cancel` button was pressed there is only `fh_Button` element.

1.3 CallBack

The fourth parameters of the call can be a name that FormGen calls for handle events; also the **B** buttons can can be associated to a function (in the *extra* field) called when they are pressed.

1.3.1 Handle Events

This function can be used to personalize the form e.g. modify the state of the Node, perform controls end so on. The functions receive an array which contains Node name and event, besides, for Nodes, contain the Node handle, the value and possibly extra field, see below.

Parameters		
	Event	**Note**
event	`Start`	Node name = `fh_start`, no Node handle and value.
event	`Cancel`	Node name = `fh_cancel`, no Node handle and value.
event	`End`	Node name = `fh_end`, no Node handle and value.
event	`Reset`	Node name = `fh_reset`
`Buttons`	`CLICK`	
`EditText`	`FOCUS`, `LFOCUS`	Focus and lost focus.
`EditText`	`VREND`	At end of Voice recognition (if is active)
`Check box`	`CHK`	Value = 1 if checked, else 0
`Slider`	`SLIDE`	Extra is the handle of text containing the value.
`combo box text`	`CLICK`	

The button `CLICK` event precedes the closing of the form, however, it is possible to inhibit the closing by setting the property `fh_yesToExit` to `False`.

```
Sub handleEvents(parm() As Object)      ' widget events handler
  Dim value As String = ""
  If parm.Length > 2 Then
       If parm.Length > 3 Then value = parm(3)
  End If
  Log("Handle event " & parm(0) & " event: " & parm(1) & " Value: " & value)
       Select parm(0)
         Case "fh_start","fh_reset"
         Case "Message"
              If parm(1) = "VREND" Then
                Dim txt As String = parm(3)
                parm(3) = txt.SubString2(0,1).ToUpperCase & txt.substring(1) & "."
              End If
         Case "Parms"
              fg.fg(Activity,"Prova,,,t","Main.handleAnswer","Main.handleEvents")
         Case "Send"
              sendMail(parm(0))
              fg.fh_yesToExit = False
       End Select
End Sub
```

Example of sub for handle events

1.3.2 Handle CallBack

The function is called when a type **B** button with call back function (the *extra* field) is clicked; the function receive the name of the button. The values of Node can be accessed by the function `valueOf(`*`NodeName`*`)`.

```
Sub Globals
...
   Dim fg As fgen    ' instantiate Form Generator class
...
   Dim Finder As String = $"T,Name;CMB,list,List;B,Find,-->,6,,Main.loadNames;"$  _
              & $"B,New;B,Ok,See,,disabled;After,Find,Name;H,app,listProd"$
...
End Sub
...
fg.fg(Activity,"Title,title,Find Products;" & Finder,"Main.handleAnswer","")
...
Sub handleAnswer(fh_Data As Map)
   If fh_Data.Get("fh_button") <> "Cancel" Then
...
    End If
End Sub
Sub loadNames(btnName As String)
   Msgbox(fg.valueOf("app"),"")
End Sub
```

Sample of Callback function

☞ We can also manage the CallBack in the event management function, in this case it is not necessary to specify a dedicated function.

1.4 Remarks

1.4.1 Handling Buttons

Form Generator inserts the `Ok` button, the `Cancel` button and the `Reset` button depending on the Nodes contained in the form:

- the `Cancel` button is always present,
- the `Reset` button is present if there are data fields**,**
- the `Ok` button is not present if there is only one combo box (**CMB** type) or some others buttons.

If the form contains only not modifiable Nodes (*type* **U**), there is only the `Cancel` button.

The Forward (`-->`) and Back (`<--`) buttons are always present in case of multi sections.

1.4.2 Data presentation

The data are presented in the order they appears in the parameters list, except for the Type **B** buttons that appears together buttons inserted by *FormGen*, at the bottom of the form.

For Node of Type Text, if the length exceed the maximum characters allowed for the line, the Node is multi lined; this maximum characters for line depends from the labels width.

With the pseudo type `after` buttons, radio buttons or check box can be placed at right of another Node.

1.4.3 CSS customization

The possibly style-sheets of parent form are added to style-sheets of form generated.

```
MainForm.Stylesheets.Add(File.GetUri(File.DirAssets,"sbFgen.css"))
```

```
.button {
      -fx-background-color: linear-gradient(#61a2b1, #2A5058);
      -fx-text-fill: yellow;
      -fx-padding: 0px;
}
.button:hover {
    -fx-background-color: linear-gradient(#2A5058, #61a2b1);
}
```

```
.button:pressed {
    -fx-background-color: black;
}
.labelClass {-fx-alignment: center-right; -fx-text-fill: black;}

/* thank to Michael Hoffer */
.comment {
    -fx-background-color: rgba(53,89,119,0);
}
.comment .scroll-pane {
    -fx-background-color: transparent;
}
.comment .scroll-pane .viewport{
    -fx-background-color: transparent;
}
.comment .scroll-pane .content{
    -fx-background-color: transparent;
}
```

1.4.4 Work with Nodes

We can modify the Node properties getting the Node by the function `getHandle`; therefore for some properties there are specific functions:

- enable Node: **`enable`**`(`*`NodeName`*`)`
- disable Node: **`disable`**`(`*`NodeName`*`)`
- get the handle of the Node: **`getHandle`**`(`*`NodeName`*`)` In case of radio button is the handle of radio button checked.
- Change the value: **`setValue`**`(`*`NodeName`*`,`*`value`*`)`
- Get the Node value: **`valueOf`**`(`*`NodeName`*`)`

Examples:

```
fg.disable("btnGo")
Dim lbl As Label = fg.getHandle("title")
lbl.TextColor = Colors.Green
If fg.valueOf("Consent") = 1 Then fg.enable("btnGo")
fg.SetValue("Slider",0.2)
fg.setValue("Number",400)
fg.setValue("combo box","Delta")
fg.setValue("UnMod","**********")
fg.setValue("Rdb","Married")
```

1.5 Others functions and utilities

FormGen module contains, may be, useful functions; some are just seen above at paragraph 1.4.4 Work with Nodes.

1.5.1 Add values to combo box

```
splitKeyValue(name As String, data As String) As List
```

Where `name` is the combo box name, `data` is a string of items in the form: `[key:]value`. separated by `|`. The `splitKeyValue` populates the `mapValues` map; the returned List is used to populate the combo box.

```
Dim cmb As combo box = fg.getHandle("listProp")
cmb.clear
cmb.addall(fg.splitKeyValue("listProp",fg.implode("|",listExpl)))
```

1.5.2 iIf

`iIf` function return an object depending on a test:

```
Dim Number As Int = fg.valueOf("Number")
Log(Number & " is " & fg.iIf((Number Mod 2) = 0,"even","odd"))
```

1.5.3 Mask

Mask function replaces comma and semicolon respectively by `#44` and `#59`.

1.5.4 Menu

The function `createMenu(`*`params`*`)` can be invoked for generate menus outside a Form.
`params` is same of pseudo type `menu` where every item doesn't contains the node type `menu`.
The function returns a menubar node.

```
Dim menuParms As String = "" _
  & "File,End,btnClose_Action;" _
  & "Some examples,Add node below,btnAdd_Action;" _
  & "Some examples,Test example below,btnTest_Action;" _
  & "Some examples,Almost complete example,btnSample_Action;" _
  & "Some examples,Node's sample,btnWidgets_Action;" _
  & "Some examples,Sections,btnSections_Action;" _
  & "Aiuto,Help;" _
  & "Help,About,btn_About;"
MainForm.RootPane.AddNode(fg.createMenu(menuParms),0,0,300,40)
```

Example 1: Creation of menu

1.5.5 Right and Left string functions

Right and Left functions returns a pieces of string stripped by some extent:

```
Log(fg.Left(fg.Right("Condor Informatique",12),6))    ' Inform
```

1.5.6 Get filename components

`pathinfo`[1] returns a map containing information about a file ex:

```
Dim fileInfo As Map = pathinfo("/www/htdocs/index.html")
```

- `dirname` `/www/htdocs`
- `basename` `index.html`
- `extension` `html`
- `filename` `index`

1.5.7 Get the widget Handle

```
Dim cmb As combobox = fg.getHandle("listProp")
cmb.clear
cmb.addall(fg.splitKeyValue("listProp",fg.implode("|",listExploiteurs)))
```

☞ the handle of a not modifiable field (**U**) is a label handle.

☞ For Combo Text (**CMT**) and Combo Extended Text (**CMX**) the handle is relative to the text associated with the combo box, for access to the combo box:

```
Dim cmbTxt As Node = fg.getHandle("Commune")
Dim aWdg() As Object = fg.widgetRef.Get(cmbTxt.Tag)
Dim cmb As combo box = aWdg(3)
...
```

[1] This is similar to PHP `pathinfo` function.

1.5.8 Extend Log function

`toText` is a function for speedy logs; it has two parameters, the first is a string containing some text and a formatting command (`%` or `%n` or `%n.d`), the second is an array of object. Every `%` is replaced by an element of the second parameter. The possible *n* is a length of output, the possible *d* is the number of decimals to show.

```
Dim aaa() As Object
aaa = Array As Object (3,3.14,"vintun",False,True, 18723)
Dim t As String = "% %6.3 %11 %1 aaaa %3 %12.2"
Log(fg.toText(t,aaa))
```

The result is:

```
3  3.140 vintun      f aaaa tru    18,723.00
```

1.5.9 Implode

`implode(`*separator* `As String,` *obj* `As List) As String`

`implodeFrom(`*separator* `As String,` *obj* `As List,` *From* `As Int,` *At* `As Int) As String`

`implode` function returns a string containing all elements of an array or list with separator between each element:

```
Log(fg.implode("<br>",aaa))
```

The risult is:

```
set<br>3<br>3.14<br>vintun<br>false<br>true<br>18723
```

1.5.10 TypeOf Object type

`typeOf` is a function which return a type of object, stripping `java.lang` from what is returned by `GetType` Basic4Android function.

```
Select TypeOf(values(j))
   Case "Integer","Double"
      ...
```

1.6 Masking data and Insert Unicode characters

In some fields (*labels*, *extras*,...) the program replace the sequence `#`*nn...n* with the corresponding ASCII character, comma is `#44`, colon is `#58` and semicolon is `#59`.

```
Title,,Buttons and text example;
B,Go,see.png;
B,No;
B,Yes,#9998,,diSabled;
B,Cancel,#x2718;
B,Reset,#x21B6;
T,mail,,,ross@lib.it;
check,mail Is Mail;
After,Go,mail
```

The function `mask(`*data*`)` can be used for mask comma and semicolon.

1.7 Sand box

The Sand box is a demo of *FormGen* and a tool for testing the forms. The initial form, not created by *FormGen,* has a text box and some buttons:

- Add button generate a form for create a Node.
- Test button generate a form starting from what is contained in the text box.
- Button for generate samples of selected type.
- Sample button generate a sample form with some Nodes and a CallBack button.

2 Technical notes

2.1 Library

JCore
jFX
CSSUtils
jRandomAccessFile

2.2 Maps

Name	Key	Value	Note
`allButtons`	*`ButtonName`*	`Array(`*`Caption,CallBack`*`)`	String array
`after`	*`NodeAfter`*	*`NodeBefore`*	
`Elem`	`Field name`	*`array field description`*	
`checks`	`Field name`	`control on Node`	array of comma separated checks (1)
`Defaults`	`Field name`	`value`	
`fh_Data`	`Field name`	`value`	
`fh_FieldsType`	`Field name`	`Field type`	
~~`hiddensMap`~~	~~`Field name`~~	~~`value`~~	~~map of hidden fields~~
`limits`	*`fieldName`*`Min`, *`fieldName`*`Max`, *`fieldName`*`Delta`, *`fieldName`* (*QS type*)	`value` `items` (*QS type*)	Slider limits; for qualitative slider also a list of values
`mapValues`	`Field name & value`	`key`	For Radio buttons and Combo box (combo box)
`requred`	`Field name`		for required fields
`widgetRef`	`Field name`	`Id, widgetType, label, extraField1`	(2)

1) `checks` array of bidimensional array:
   - the check condition *`fieldName operator (value|fieldName)`*
   - possibly error message.

2) `widgetRef` contains a reference to all Node
   - key = *`fieldname`*, value = `array(`*`Id, widgetType, label, extraField`*`)`
   - `Id` is a widget `ID`, for RadioButtons is a panel container
   - *`extraField`* is:
     - for button a possibly sub for CallBack,
     - for seekbar (type **S**) is the ID of text containing the value,
     - ~~for file (type **F**) is the file path,~~
     - for combo box text (type **CMT** and **CMX**) is the ID of the combo box associated.
     - for RadioButton is a radiobutton selected.

2.3 Lists and arrays

- `rdbList` the data array contains normalized RadioButtons data description

- `Sections` two dimension array (number of sections, 3):
  - section widgets,
  - section name,
  - check condition(s).
- `label` an array in the form `("",left|right,`*`afterLabel`*`)`.

3 History

0.5.3	1 October 2018	first release used in work
0.5.5	22 October 2018	Added **`DATE`** node and **`MENU`** pseudo type
0.5.8	25 November 2018	Fixed lack of default for node **`DATE`**
		Fixed error on management of combo box

4 Annexes

4.1 Introduction to regular expressions

A regular expression is a string of characters used to search, check, extract part of text in a text; it has a cryptic syntax and here there is a sketch with a few examples.
The regular expression can be prefixed by modifiers such as **(?i)** to ignore the case.
The expression is formed with the characters to search in the text and control characters, among the latter there is a \ said *escape* used to introduce the control characters or categories of characters:

- **\ escape character**, for special characters (for example asterisk) or categories of characters:
  - **\w** any alphabetical and numerical character, **\W** any non alphabetical and numerical character,
  - **\s** *white space* namely. tabulation, line feed, form feed, carriage return, and space,
  - **\d** any numeric digits, **\D** any non digit,
- **.** any character,
- **quantifiers**, they apply to the character(s) that precede:
  - ***** zero or more characters
  - **+** one or more characters
  - **?** zero or one character (means possibly)
  - **{*n*}, {*n*,}** and **{*n*,*m*}** respective exactly *n* characters, almost *n* characters and from *n* to *m* characters.

(...) what is between parentheses is memorized,
?=*pattern* checks if *pattern* exists,
[a-z] any letter from a to z included,
[*a*|*b*] a or b,
\b word boundary,
$ (at the bottom),
^ (at start).

4.1.1 Examples

^\s*$	Empty set or white spaces
aa+	Find a sequence of two or more a, like aa, aaa,....
(\w+)\s+(\w+)\s+(\w+)	Find and memorize three words
(\-[a-z])	Find and memorize minus followed by one alphabetic character
\.(jpg|jpeg)$	Controls file type jpg or jpeg
^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$	Control of mail address
^\d+$	Only integers
((?=.*\d)(?=.*[a-z]+)(?=.*[\W]).{6,12})	(?=.*\d) almost a digit from 0-9 (?=.*[a-z]) almost one lowercase character (?=.*[\W]+) almost one special character . match anything with previous condition checking {6,12} length at least 8 characters and maximum 20
^[-+]?\d{1,2}(\.\d{1,2})?$	**Numeric values** [-+]? the sign is possible \d{1,2} one or two digits (\.\d{1,2})? It is possible to have a decimal point followed by one or two digits

5 Alphabetical Index