B4J Form Generator

Disclaimer

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:

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

The form is generated by calling the `fg` or `fgw` methods:

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

`instantiatedName.fgw(dataList,callingModule,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 (function) 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 (paragraph 1.3.1 Handle Events).

☞ `subHandleAnswer`, and `subHandleEvents` must be in the `callingModule`.

If the form is only for show data `subHandleAnswer` can be an empty string; calling `fgw` allows to wait for the closing of the form and continue in the same function; see the example below:

```
...
fg.Initialize(MainForm,500,600)
...
fg.fgw(eventForm,Me,"")
Do While fg.fh_isrunning
      Sleep(100)
Loop
If fg.fh_Data.Get("fh_button") <> "Cancel" Then
...
```

Other 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 separated by comma, in this order: node *Type*, node *Name*, node *Label*, node *Length*, *Default* Value and *Extra*. Nodes are separated by semicolon.

☞ Some field (*label*, *default*, *extra*, condition and title) if contains commas or semicolons must be enclosed in `'` or `"`, alternatively commas can be replaced by `#44` and semicolons by `#59`, for this one can use the function `mask(data)`(see paragraph 1.5.6).

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;
    - **IB** in line button;
    - **R** or **RDB** radio button, a set of Radio buttons;
    - **TB** toggle button.
- **CKB** check box;
- **CKL** a set of check boxes.
- combo boxs:
    - **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 ;
    - **D**, directory,
    - **F**, **FILE** file;
    - **WF**, **WFILE** output file;
- Date and Time
    - **DATE**
    - **TIME**
- Text fields:
    - **COMMENT**, **C** comment;
    - **N** numeric field;
    - **DN** decimal numeric field;
    - **S** slider is an extension of the standard node;
    - **QS** qualitative slider;
    - **P** password field, the data entered are masked;
    - **T** text field is the default if the Type is omitted;
    - **U**,**UN** 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 `FieldName` that it is transformed if it has one of those formats:

- *`fieldName`* it becomes *`Field name`*,
- *`field_name`* it becomes *`Field name`*.

1.2.4 Length

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

	Node type	Value	
`A`	Text Area	200	
`DATE`		18	
`DN`	Decimal numeric text	11	
`F,FW,D`	File	`30`	
`N`	Numeric text	`9`	
`R`	Radio button	`12`	
`QS`	Qualitative slider	`10`	Dimension of the value shown
`S`	Slider	`6`	Dimension of the value shown
`P`	Password	`20`	
`TIME`		`8`	
`T`	Text node		maximum from `25` 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 tip
Radio buttons	an item list separated by |
Slider	the slider limits
Qualitative slider	an item list separated by |
combo box	an item list separated by | also see also paragraph 1.2.7.5 Combo boxes
Text Fields	if the default field is empty, is the hint, the possible second *extra* field is a tool-tip
Toggle button	an item list separated by |

1.2.7 Summary by type

1.2.7.1 Default and extra field

Type	Length	Default field	Extra field
`B`	Ignored	possibly `dis[abled]`	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`*
`DATE`		Date in format accepted, also accepted `today`, `tomorrow` and `yesterday`.	
`F`		Initial folder	
`S`		Initial value	Start, end and possibly numbers of decimals, default is `0 100`
`T,N,DN,P`		Initial value	hint, tooltip
`U`		Not modifiable text	

1.2.7.2 Buttons

B,Cancel|Reset,*caption*|*fileName*,[70|*pixelLength*],[dis[able]][,*confirmMessage*][,*fontInfo*]

B|IB,*name*,[[*caption*|*fileName*]:*description*],70|*pixelLength*,[dis],*callBackFunction*[,*fontInfo*]

The package adds the standard buttons `Ok`, `Cancel` and `Reset`.
If there are sections:

- the `Ok` button is in the last form;
- the button whit caption `-->` for access the next form with name `fh_forward` is in all forms except the last one;
- the button with caption `<--` with name `fh_Back` is in all forms except the first one.

The buttons can be used both for take different actions on closing form both for show user caption instead of the default.
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 *fontInfo* in the second *extra* field can contains some text properties separated by space:

- *FontSize* a numeric value, the default is `12`, ☞ for technical reason must be the first parameter.
- *Typeface* values accepted are `mi` or `materialicons` and `fa` or `fontawesome` (indifferent to case). ☞ `FontAwesome` font includes the standard Latin characters, ☞ Material Icons only includes the upper case characters.
- *color* see paragraph 1.2.8.3 Form (Background) for the values accepted.
- Gravity information i.e. `left`, `center` or `right` (not applicable to buttons).

The inline button **IB** occupies the space of the label, the *caption* can be character(s) or a graphic file; the possible *description* is shown to the right of the button; ☞ *fontInfo* applies only to the caption and not to the *description* that is affected by a possible *fontInfo* of *Label* pseudo type.

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

```
...
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* or #[x|X]*eeee* where *nnnn* is a decimal value of the Unicode character or *eeee* his hexadecimal value.

The button can also have an image that must be in the assets folder; the name of image is in the label field

Name	Decimal value	Symbol	Hexadecimal value
edit	#9998	✎	#x270E
delete	#10008	✘	#x2718
check	#10003	✓	#x2713
check bold	#10004	✔	#x2714

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

email	#9993	✉	#x2709
cross	#10006	✖	#x2716
	#8630	↶	#x21b6
euro		€	#x20AC
pound		£	#xA3
white square		▢	#x25a2
Right arrow	#10143	➟	#x279f

1.2.7.3 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(iIF(fh_Data.Get("Mandatory") = "1","M",""))
```

1.2.7.4 Check box list

`CKL` type generates a set of check boxes.

The extra field contains a list of field names separated by `,` (comma) with syntax: `[`*`key`*`=]`*`value`*`[, [`*`key`*`=]`*`value`*`[,...`; the field name of check box is *`key`* if present, otherwise is *`value`*; *`value`* is the description that appears after the check box.

The possible default must be one of the items; it is possible set others default by the *pseudo type* `Default`.

The *Field Nam*e of the check box list will contain the number of check boxes selected.

Example:

```
CKL,Languages,,,Rust,C++|Julia|Rust|PHP|Others;
Default,Others:1
```

1.2.7.5 Combo boxes

CMB is a combo box, if no member is selected the value returned is an empty string; if the form has only one combo box, there aren't buttons and the form is exited when an item is selected.

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

The *extra* field can also contain:

`%days`	is generate a list of days name, the names are in local language.
`%#days`	is generate a list of days name that returns the number corresponding `1 = Sunday`, … `7 = Saturday`
`%months`	is generate a list of months name
`%#months`	is generate a list of months name that returns the number corresponding (from `1`).

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.

☞ For **CMB** or **CMT** the possible default value can be both value both key.

1.2.7.6 Date

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

In *default* are accepted `toady`, `tomorrow`, `now` and `yesterday` (also prefixed by `%`).

1.2.7.7 File and Folder

The type **F** is for input files, **WF** for output files, **D** for folder.

The first *extra* field can contains a hint , the subsequent extras of **F** and **WF**, contain file extensions:

```
WF,File,Image File,30,,Image,*.jpg,*.png;
```

```
F,psFile,PDF and PS files,50,,PDF and PS files,*.pdf,*.ps;
```

☞ If the default is omitted the folder is `File.DirApp`.

1.2.7.8 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;
```

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

1.2.7.9 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, the end values and a possibly numbers of decimals in the form *`start end [decimals]`*, e.g. `-5 5`; the range, if omitted, is `0 100`; if only one value is present, the default value for the second is `100`; the result, if *`decimals`* isn't present, has decimals depending on the difference from `start` and `end` value, see table at right.

`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 note that after a possible Reset the value is set to minimum.

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

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
...
```

1.2.7.10 Text fields

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

The **A** type is a multi line text.
If the form has only one text field (not type A) the form is exited also for `enter` key.

1.2.7.11 Time

The **Time** node is realized by two sliders for hours and minutes in the form `HH:mm` therefore also the default value must have this format; if only one number is provided this is considered as the hour, see examples below.

```
Time,t1;          ' no default time
Time,t2,,,10;     ' ten o clock
time,t3,,,8:05;   ' eight and five
```

1.2.7.12 Toggle

Toggle or **TB** is like an inline **Button** that when clicked change the caption.

```
Toggle|TB,name,,pixellength,default,stateList
```

`stateList` is a list of states in the form of the element of radio buttons,; the toggle must have two or more states, the *`stateList`* is omitted this is assumed to be `On|Off`.

1.2.8 Pseudo types

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

1.2.8.1 After

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

`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`* and the pseudo type after the two.

```
...
& "Title,title,After example;" _
& "B,InfoPD," &  Chr(0xF05A) & ",30,,infoShow,14 fa blue;After,InfoPD,title" _
& "N,Age,,5,,age;" _
& "R,Sex,,10,Man,M:Man|F:Female;" _
& "After,Sex,Age;" _
...
```

1.2.8.2 Check

This pseudo type is used for some controls on fields; the control occurs when the field lost the focus and when the form is exited (by `Ok` button(s)).

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

where operator can be one of `=`, `eq`, `<>`, `ne`, `lt`, `<`, `le`, `<=`, `gt`, `>`, `ge`, `>=`, or `is`; after `is` operator can have:

`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, the function `mask` can be used for mask commas `(#44)`, semicolons `(#59)`, apostrophes `(#39)`, double quotes `(#34)` and line feed `(#10)`.

1.2.8.3 Form (Background)

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

`[Form|Ground|Background][,`*`Color`*` [`*`toColor direction`*`]]`
`[,style[=| ]`*`assetFileStyle`*`]`

Ex. `Form,silver,style sb2Fgen.css;`

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

- *`Color`*, *`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[1], the name can be prefixed by two hexadecimal characters for sets a transparency level;
- *`direction`* is the linear gradient direction: `[left | right] | [top | bottom]`

```
"Form;"                       ' default FFC0C0C0
"Form,7FBlue;"                ' blue half transparent
"Form,7F0000FF;"              ' half transparent blue
```

---

[1] The colors accepted (case ignored) are: aqua, black, blue, crimson, cyan, gray, green, lightgray, magenta, olive, orange, purple, red, silver, teal, white, yellow.

```
"Form,FF 8000 top left;" ' gradient from green to blue on top left
```

1.2.8.4 Comment

The syntax is: `[C|COMMENT],fieldName,comment[,pixelLength][,pixelHeight][,fontInfo]`

The size of the comment is determined by the maximum width of the lines and by the number of these (being terminated with new line); if `length` is present is the width dimension and the lines are calculated by comment length / length; if `height` is present the comment is inserted in a scroll view.

```
...;C,c1,Fill out this form carefully!,red center;…
```

`FontInfo` (see paragraph 1.2.7.2 Buttons) can have the parameter `fill` that force the comment to occupy horizontally the entire form.

1.2.8.5 Defaults

The syntax is:

```
defaults,NodeName:NodeValue[,...]
```

`Defaults` it is useful for populate the form.

In case of radio buttons or combos the default can be both value both key.

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

1.2.8.6 Dictionary

```
Dict[ionary],function[,param]
```

This pseudo permits translation. `function` is a custom function that return a map where the key is the English word or sentence and the value is the translation. `param` normally indicates the target language.

1.2.8.7 Label

For manage the view's labels:

```
label,,fontInfo|left,afterLabel]
```

`afterLabel` are the possibly characters inserted after the label.

☞ The second value is intentionally empty and it is ignored.

`FontInfo` see paragraph 1.2.7.2 Buttons.

Example:

```
Ground,FF 8000 LEFT_RIGHT;Title,,Label example;
Label,,14 fontawesome Right,: ;T,$,,25;
```

☞ For a correct determination of space for labels `label` must precedes the views.

1.2.8.8 Menu

Add menu in menu bar:

```
Menu,father,son,[Main.|modulName.]function
```

The menu at right has been generated by :

```
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;
```

☞ the sub items must be (for now) all together.

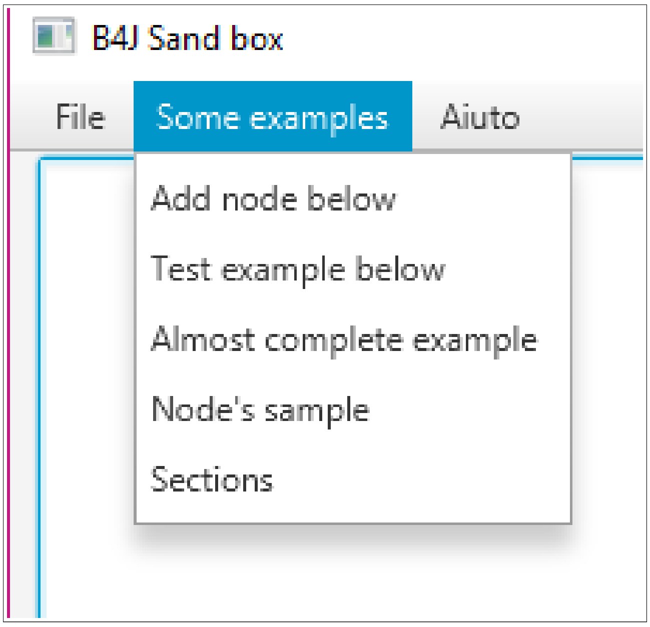

Figure 1: Example of menu

1.2.8.9 Required

A list of fields that must be inserted:

`required,`*field*$_1$`[,`*field*$_2$`...]]`

1.2.8.10 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,%now;"
```

1.2.8.11 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.

For every section it is generated a title with name `fh_section`*`i`* and value *`sectionName`*.

1.2.8.12 Title

```
Title,name,formTitle[,fontInfo[,backgroundColor]]
TITLE,Title,Send mail parameters,20 olive,FF202020;
```

The default *`color`* is `black`; colors can be the hexadecimal value or (some) color name (see paragraph 1.2.8.3 Form (Background) and note).

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

1.2.8.13 Window

`Window,[,`*`backGroundColor`*` [`*`toColor`*` [`*`direction`*`]][,border[`*`color`*`]|shadow]`

Ex. `Window,teal,shadow;`

With the `Window` pseudo type the form isn’t generated in form but can be show in the Main form, this is accomplished in the *`subHandleEvents`* by means the event name `Window` at `fh_start`.

The generated form inherits any associated style.

```
Dim frmParms As String = "Window,FF0000FF FF808080,border;" _
      & "Title,t,Initial Parameters;" _
      & "C,'This form is on first time, insert your email name and language.';" _
      & "T,user,My name,20;" _
      & "T,email,My eMail,25;" _
      & "CMB,language,Language,20,EN,EN:English|FR:Francais|IT:Italiano|
ES:Español|PT:Português|SW:Swahili;" _
      & "check,email is mail;" _
      & "Required,language,email;"
      fg.Initialize
      fg.fgw(frmParms,Me,"handleShowForm")
      Do While fg.fh_isrunning
            Sleep(100)
      Loop
...
Sub handleShowForm(parm() As Object)
  If parm(1) = "Window" Then
    Dim wleft As Int = IIf(MainForm.WindowWidth < parm(4)+10,0,0.5*(MainForm.WindowWidth
- parm(4)+10))
    Dim wtop As Int = IIf(MainForm.WindowHeight < parm(5)+10,0,0.5*(MainForm.WindowHeight
- parm(5)+10))
    MainForm.RootPane.AddNode(parm(3),wleft,wtop,parm(4)+10,parm(5)+10)
  End If
End Sub
```

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 that is the map which contains the data, data that are accessible via `Get` or `GetDefault` methods: whit the key is the field name; besides there is the element with key `fh_button` that 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 last parameters of the call can be a name of sub that *FormGen* calls for handle events.

1.3.1 Handle Events

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

Parameters				
View or state	**Event Name**	**Handle**	**Value**	**Extra - Note**
`fh_start`	`Start`		*`TitleName`*	*`Section`* number
`fh_start`	`Window`	*`panelhandle`*	`sWidth` `sHeight`	At start of form in the main form

`fh_cancel`	`End`			*`Section`* number
`fh_end`	`End`			Ok button(s)
`fH_Forward` `fh_Back`	`End`		*`section`*	
`fh_reset`	`Reset`			*`Section`* number
`viewName`	`SETVALUE`	*`viewHandle`*	`value`	`setValue` function
`buttonName`	`CLICK`	*`viewHandle`*		
`viewName`	`CHANGED`	*`viewHandle`*	`newValue`	Text fields `oldValue` in fourth cell
`viewName`	`ENTER`	*`viewHandle`*	`Value`	Text fields

`viewName`	`FOCUS,`
	`LFOCUS`
		`viewHandle`	`value`	Text fields Focus and lost focus.

`checkboxName`	`CKB`	*`viewHandle`*	`value`	Value = 1 if checked, else 0
`seekbarName`	`SLIDE`	*`viewHandle`*	`value`	Extra is the handle of label that contains the value.
`comboName`	`CLICK`	*`viewHandle`*	`value`	Handle to spinner for CMT and CMX type)

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 Nodes 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"))
```

The style-sheet can be added to a generated form by the pseudo type `Form`:

```
Form,silver,style sb2Fgen.css;
```

```
.button {
      -fx-background-color: linear-gradient(#81C2D1, #4A7078);
      -fx-text-fill: #7F0000;
      -fx-padding: 0px;
}
.button:hover {
    -fx-background-color: linear-gradient(#4A7078, #81C2D1);
}
.button:pressed {
    -fx-background-color: grey;
}
```

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`**`(Node`*`Name`*`)`
- 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.

☞ In order to use the *FormGen* functions, it is advisable to initialize an instance to be used for this purpose.

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

The function `setComboValues(`*`viewName`*`,`*`data`*`)` replace the values of spinners (**CMB**, **CMT**, and **CMX**). *`data`* is a string of items in the form: `[key:]value`. separated by `|`.
The function returns the number of items.

1.5.2 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

1.5.3 Get filename components

`pathinfo`[2] 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`

1.5.4 Get the node 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)
...
```

1.5.5 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.

1.5.6 Masking data and Insert Unicode characters

In some fields (*labels*, *extras*,...) the program replace the numeric sequence `#`*`nn`*`...`*`n`* or hexadecimal sequence `#x`*`hh`*`...`*`h`* with the corresponding ASCII character, comma is `#44` 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
```

1.5.7 Miscellany

Clean form	`cleanForm`	Efface the form
Contains parameter	`containsParm(`*`Key`*` As String,`*`data`*` As String) As Boolean`	
Extract parameter	`extractParm(`*`Key`*` As String,`*`data`*` As String) As String`	*`data`* is `key=value`
Hide form	`hide(`*`form`*` As Form)`	
Left string function	`Left(`*`s`*` As String,`*`l`*` As Int) As String`	

---

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

Right string function	`Right(`*`s`*` As String,`*`l`*` As Int) As String`	
Show form	`show(`*`form`*` As Form)`	

Text dimension	`measureText(s As String, `*`fontSize`*` As Int) As B4XRect`	`B4XRect.Width`
		`B4XRect.Height`
		(font `default`)

Text width	`getLabelWidth(`*`t`*` As String) As Int`	
Type Of Object	`TypeOf(`*`obj`*` As Object) As String`	returns a type of object, stripping `java.lang` from what is returned by `GetType` B4x function.

1.6 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

`CSSUtils`
`jRandomAccessFile`
`jXUI`
`JavaObject`
`jReflection`

2.2 Maps

Name	Key	Value	Note
`allButtons`	*`ButtonName`*	`Array(`*`Caption,event,CallBack,Handle`*`|0,`*`width,confirmString`*`)`	String array, no IB and TB buttons
`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`	*`fieldNameDecimals`* *`fieldName`* (*QS type*)	`value` `items` (*QS type*)	
`mapValues`	`Field name & value`	`key`	For Radio buttons and Combo box
`required`	`Field name`		for required fields
`widgetRef`	`Field name`	`Id, widgetType, label,` $\texttt{extraField}_1$`,` $\texttt{extraField}_2$`,right node extension`	(2)

1) `checks` array of bi dimensional 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`*`)`

   1. *`Id`* is a widget `ID`, for RadioButtons is a panel container
   2. *`widgetType`*
   3. *`label`* is the field label; for Comment is the *`ID`* i.e. the comment itself.
   4. $\textit{extraField}_1$ is:
      - ~~for button a possibly sub for CallBack,~~
      - for seekbar (type **S**) is the ID of text containing the value,
      - for check list (type **CKL**) is the checks list,
      - for combo box text (type **CMT** and **CMX**) is the ID of the combo box associated.

    - for Radio Button is a radio button selected.
    - For Title a button after
5. *`extraField`*$_2$ is the minutes slider of Time node
6. the right extension of node

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).

3 History

The ☞ symbol indicates a change from the previous version.

0.5.3	1 October 2018	first release used in work
0.5.5	22 October 2018	Added **`Data`** node and **`Menu`** pseudo type
0.6.0	9 December 2021	• Added A node (textarea), • ☞ the background of title has no default, • ☞ the logical operators `<,` <=, <>, >, >= of Check pseudo type, have been replaced by respectively `lt`, `le`, `ne`, `gt` and `ge`. • the Time node manage hours and minutes, • the size of the form is automatically adapted to its content, • the labels have a correct dimension, • the empty Date returns empty instead of 1970…, • the combo box can be set after other node, • added the `%#days`, ☞ the days list start at `Sunday` instead of `Monday.`
0.6.1	30 January 2022	• Added the type **D** folder, • in the type **F** and **WF** if the default is omitted the folder is `File.DirApp`, • correct bug in `pathinfo` function.
0.7.0	13 April 2023	• Added type **`CKL`** check list, • Added **Window** pseudo type, • Added type **`TG`** Toggle, • Colors can be prefixed by two Hexadecimal characters for sets the transparency, • Added **`Form`** which replaces the (deprecated) **`GROUND`**, it accepts the parameter `style=`*`AssetFileStyle`*

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,

`(?:...)` a non-capturing group,

`?=`*`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