

Neil J Martin  
neilm@4js.com

# Genero User Interface Forms

Genero BDL 3.10  
Genero Studio 3.10



# Goals

- Learn all the elements of a form file.
- Use the LAYOUT section to design a form.
  - Lay out the form with containers and layout tags
  - Add form items such as combo boxes, buttons and radio groups.
- Compile the form definition file ( .per ).
- Read and interpret a compiled form file (.42f ).



# Elements of a Genero Form

- **Container** – a formatted screen region or a container for other containers (Grid, Folder, Table, groupboxes, etc.)
- **Form field** – an area where users can view and/or edit data
- **Widgets** – the decoration for the form field (Edit, ButtonEdit, ComboBox, RadioGroup, etc )
- **Action View** – a form item that can trigger a program action (Button, Topmenu, Toolbar, etc.)
- **Label** – text displaying static information



## Genero Forms – Legacy / Ascii Terminal

A common fixed form (not sizeable, with fixed font) would be represented as a grid with fixed sized cells:

[illegible]

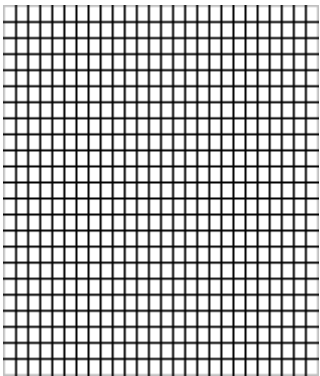
# FOUR Js

The Power of Simplicity



# Genero Forms – Modern

Genero forms are sizeable and can include proportional fonts and graphical objects of different sizes.



- Each cell has its own size
- Default size is about 2 pixels
- The cell takes the size of its content or of the larger cell in the same column.
- An object can use one or more cells, depending on its definition



# Genero Forms – Layouting Rules

<div>▼</div>				
<input type="checkbox"/>	Check Box			

- Adding a Combo box with a width of 5 cells will enlarge these cells to the combo box size.
- The last cell is kept for the combo box button.
- Cells around the combo box are automatically retained to provide a margin.



# Genero Forms – Creating a Form

Three ways to define application screens:

- Write text-based form definition files with a suffix of .per
- Use the drag-and-drop graphical Form Designer in Genero Studio
- Dynamically at runtime



# Genero Forms – Definition File ( .per )

The .per file consists of a set of defined sections, which must appear in the following order in the file:

- SCHEMA
- ACTION DEFAULTS
- TOPMENU
- TOOLBAR
- LAYOUT
- TABLES
- ATTRIBUTES
- INSTRUCTIONS

Only LAYOUT is required, the rest are optional depending on the features of the form.





# Genero Forms – Definition File ( .per )

## SCHEMA Section

- Defines the database schema on which the form is based.
  - You can define fields on the form in terms of the schema tables and columns.
  - The Schema must be defined in a database schema file.
- Optional; defaults to FORMONLY if omitted.

Syntax:

```
SCHEMA ( string [ FORMONLY ]
```



# Genero Forms – Definition File ( .per )

## LAYOUT Section

- The LAYOUT section defines the abstract layout of the elements of the form.
- This section is a tree of *layout nodes*, used to define graphical elements.
- Mandatory

## Syntax:

```
LAYOUT [ ( attribute [ = value ] [, ...] ) ]  
      root-layout-container  
      [...]  
[END]
```



# Genero Forms – Definition File ( .per )

## ATTRIBUTES section

- Describes the properties of elements in the LAYOUT section
- Mandatory

## Syntax:

ATTRIBUTES

```
[item-type] item-tag = field-name [ , attribute-list ] ;
```

```
[item-type] item-tag : item-name [ , attribute-list ] ;
```

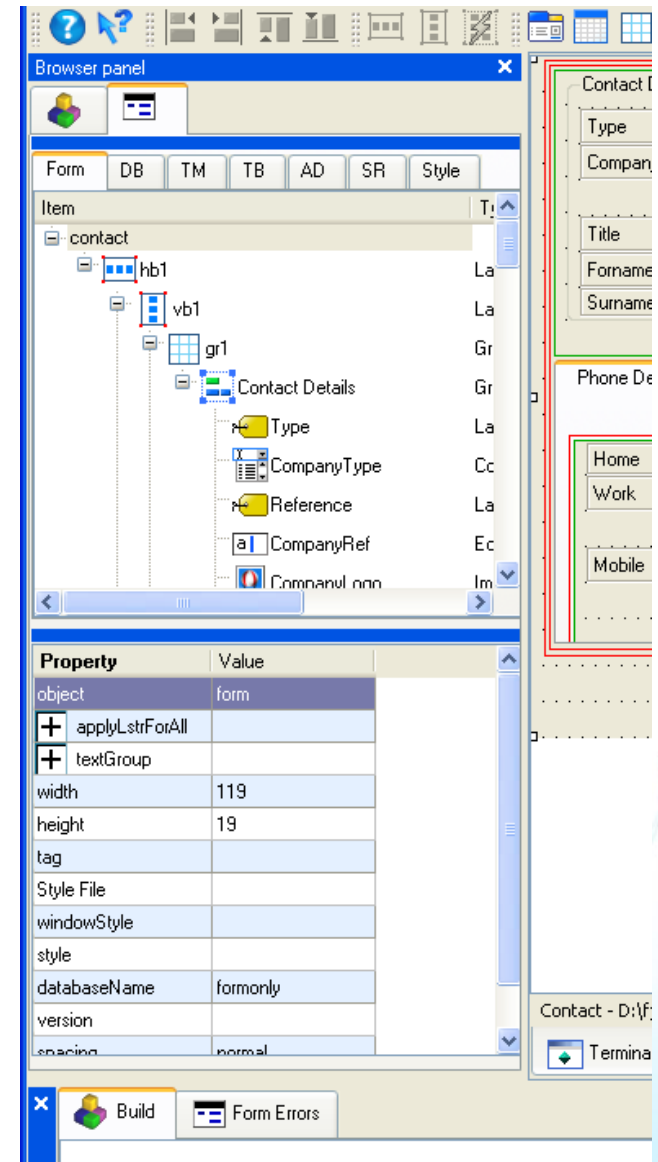
...

END



# Genero Forms – Studio Form Designer ( .4fd )

- In Genero Studio Form Designer all sections and widget attributes are shown and can be updated in the browser panel.
- The browser panel is located by default on the left side of Genero Studio screen.



# Genero Forms – Containers

Containers are blocks holding other containers or defining a formatted screen region.

- They are nested to create the visual layout of the form area of the user interface.
- The END keyword defines the end of the list of children inside a container.

HBOX  
VBOX  
GROUP  
FOLDER  
PAGE  
GRID  
SCROLLGRID  
TABLE



# Genero Form – .per Example

```
LAYOUT ( text = "Customer Orders" )
  VBOX
    GROUP ( text = "Customer Details" )
      GRID
      {
        Name:      [f001 ]
        Phone:     [f002 ]
        Address:   [f003 ]
      }
    END --GRID
  END --GROUP
  TABLE
  {
    OrdNo      Date      Ship date      Weight
    [c01       |c02      |c03          |c04      ]
    [c01       |c02      |c03          |c04      ]
    [c01       |c02      |c03          |c04      ]
  }
  END --TABLE
END --VBOX
END --LAYOUT
```



# Genero Forms – GST Layout Example

The diagram illustrates a form layout with the following components and annotations:

- VBox**: Points to the top yellow header bar.
- Group Box**: Points to the 'Customer Details' section.
- Grid**: Points to the table structure.
- Table**: Points to the table content.

**Customer Details**

Name:   
Phone:   
Address:

OrdNo	ShipNo	Date	Weight
Edit	Edit	Edit	Edit
Edit	Edit	Edit	Edit
Edit	Edit	Edit	Edit
Edit	Edit	Edit	Edit
Edit	Edit	Edit	Edit

The screenshot shows a running application window titled "Example". It contains the same form layout as the diagram above, but with a few additional elements:

- A "Main" panel on the right side containing an "Exit" button.
- A status bar at the bottom right showing "OVR" and a small icon.

**Customer Details**

Name:   
Phone:   
Address:

OrdNo	ShipNo	Date	Weight
-------	--------	------	--------

# Genero Forms – Containers

Containers have a set of supported attributes that can be applied in the .per definition for presentation and behavior.

Example:

GROUP (TEXT = "Customer Details", HIDDEN)

The documentation lists the valid attributes for each container, and includes an Attributes List.





# Genero Forms – Containers



## VBOX (Vertical Box)

- Displays the elements defined within the container vertically from top to bottom.



## HBOX (Horizontal Box)

- Displays the elements defined within the container horizontally from left to right.

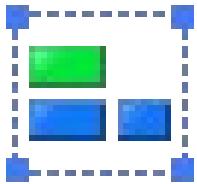
## Syntax:

```
VBOX [ ( attribute [ = value ] ) ]  
      layout-container  
      [...]
```

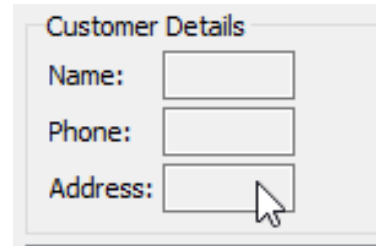
END



# Genero Forms – Containers



A GROUP container can be used to display a titled box (usually called a groupbox) around contained elements.

A screenshot of a web form titled "Customer Details". It contains three input fields: "Name:", "Phone:", and "Address:". A mouse cursor is pointing at the "Address:" field.

Syntax:

```
GROUP [ ( attribute [ = value ] ) ]  
      layout-container  
      [...]   
END
```

Where *layout-container* can be:

VBOX, HBOX, GROUP, FOLDER, GRID, TABLE



# Genero Forms – Containers



A FOLDER container can be used to display PAGE containers as folder tabs.

Syntax:

```
FOLDER [ ( attribute [ = value ] ) ]  
    PAGE  
        layout-container  
        [...]  
    END  
    [...]  
END
```

Page 1   Page 1

Customer Details

Name:

Phone:

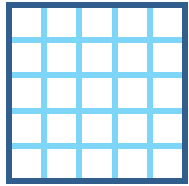
Address:

PAGE containers can include other containers such as GROUP and GRID

Where *layout-container* can be.

VBOX, HBOX, GROUP, FOLDER, GRID,  
SCROLLGRID, TABLE

# Genero Forms – Containers



The GRID container declares a formatted text block, encased in braces “{...}”, defining the dimensions and the relative positions of the logical elements of a screen. Form fields are normally defined here.

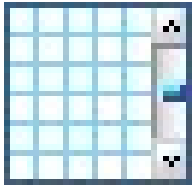
A form layout example showing three stacked input fields labeled Name, Phone, and Address. A mouse cursor is pointing at the Phone field.

## Syntax:

```
GRID [ ( attribute [ = value ] ) ]  
  {  
    { text | item-tag | layout-tag }  
    [...]  
  }  
END
```



# Genero Forms – Containers



SCROLLGRID is similar to the GRID container, except that you can repeat the screen elements on Several "record rows" in order to design a multiple-record view, which will appear with a scrollbar.

A screenshot of a web form titled 'SCROLLGRID'. It contains three identical rows of input fields. Each row has a 'Name:' label followed by a text input box, a 'Phone:' label followed by a text input box, and an 'Address:' label followed by a larger text input box. A vertical scrollbar is on the right side of the form, indicating that the rows can be scrolled through.

Syntax:

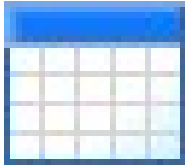
```
SCROLLGRID [ ( attribute [ = value ] )  
[  
{  
    row-template  
    [...]  
}  
END
```

where *row-template* is a text block containing:  
{ text | item-tag | h-line }  
[...]

## Do Chapter 2 - Exercise 1



# Genero Forms – Containers - Tables



The TABLE container defines the presentation of a list of records.

The first line of a table-area can be text defining the column titles.

The second line must be field identifiers that define the columns receiving data.

Syntax:

```
TABLE [ ( attribute [ = value ] ) ]  
{  
  [ title [...] ]  
  [ identifier [|...] ]  
}  
END
```

OrdNo	ShipNo	Date	Weight
265	4534326	01/03/2018	45.21
266	1214315	01/03/2018	102.19



# Genero Forms – Containers - Tables

A Table can also have a summary line for either automatically calculated values or program calculated.

```
TABLE
{
  Stock#  Description      Qty    Unit    Price    Total
[f08     |f09              |f10    |f11     |f12      |f13    ]
[f08     |f09              |f10    |f11     |f12      |f13    ]
[f08     |f09              |f10    |f11     |f12      |f13    ]
[f08     |f09              |f10    |f11     |f12      |f13    ]
                                         [sum    ]
}
END
```

```
AGGREGATE  sum = FORMONLY.total;
```

Stock#	Description	Qty	Unit	Price	Total
310	sink stoppers	5	grss	12.85	64.25
456	lightbulbs	10	ctn	5.55	55.50
744	faucets	60	6/bx	250.95	15057.00
					15176.75





## Do Chapter 2 - Exercise 2



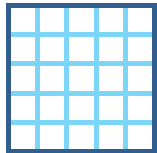
# Genero Forms – Layout Tag

Layout tags define layout regions within a GRID or SCROLLGRID container.

G - Groupbox layout tag gives same presentation as a GROUP container.

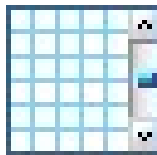
T - Table layout tag gives same presentation as the TABLE.

S - ScrollGrid layout tag resulting in the same presentation as the SCROLLGRID.



GRID

```
{
<G "First"          ><G "Second"          ><G "Third" >
  Name:   [f001 ]   Address: [f003 ]       Some TEXT
  Phone:  [f002 ]   State:   [a0]          Some TEXT
                               Zipcode: [f010 ]   Some TEXT
<                               ><                               >
<G "Fourth"          >
  Some More Text or Form Items
<                               >
}
END -- GRID
```



First	Second	Third
Name: <input type="text"/>	Address: <input type="text"/>	Some TEXT
Phone: <input type="text"/>	State: <input type="text"/>	Some TEXT
	Zipcode: <input type="text"/>	Some TEXT
Fourth Some More Text or Form Items		



# Genero Forms – Spaces

An HBox Tag defines the position and size in a GRID of a horizontal box containing several form items.

Syntax:

[ element [...] ]

Where element can be:

{ identifier [-] | string-list } [:...]

Where string-list is:

{ string-literal | spacer } [...]



# Genero Forms – Solving Misaligned Elements

GRID

```
{  
  First Name: [fldfname           ]   Last Name: [fldlname           ]  
  Address:    [fldaddress         ]  
              [                   ]  
              [                   ]  
  Zip Code:   [fldzip ] City: [fldcity           ]  
  Country:    [fldcountry         ]  
}  
END --GRID
```

The screenshot shows a window titled 'Contact' with several input fields. The 'First Name' and 'Last Name' fields are on the same line, but the 'Last Name' field is misaligned to the right. The 'Address' field is a single large box. The 'Zip Code' and 'City' fields are on the same line, but the 'City' field is misaligned to the left. The 'Country' field is a single box below them. On the right side, there is a 'Main' section with a power button icon and the text 'Exit'. At the bottom right, there is a label 'OVR'.



# Genero Forms – Containers

GRID

```
{  
  First Name: [fldfname : : "Last Name": fldlname ]  
  Address:    [fldaddress  
              [  
              [  
  Zip Code:   [fldzip : : "City": fldcity ]  
  Country:    [fldcountry  
}  
END --GRID
```

Spacers:

When the window is drawn, the length will change within the spacer, not within the whole line.

# Do Chapter 2 - Exercise 3



# Genero Forms – Form Elements - .per File.

```
LAYOUT ( text = "Customer orders" )
```

```
  VBOX
```

```
    GROUP ( text = "Customer details" )
```

```
      GRID
```

```
      {
```

```
        Name:      [f001 ]
```

```
        Phone:     [f002 ]
```

```
        Address:   [f003 ]
```

```
      }
```

```
    END --GRID
```

```
  END --GROUP
```

```
  TABLE
```

```
  {
```

```
    OrdNo
```

```
    Date
```

```
    Ship date
```

```
    Weight
```

```
    [c01      |c02      |c03
```

```
    [c01      |c02      |c03
```

```
    [c01      |c02      |c03
```

```
  }
```

```
END --TABLE
```

```
END --VBOX
```

```
END --LAYOUT
```

Text  
labels

Form  
Fields

Pipe symbol used to make items  
appear next to each other

# Genero Forms – Form Elements - .4fd File

The diagram illustrates a Genero Forms interface with three main components highlighted by callouts:

- Text labels:** Points to the 'Name:', 'Phone:', and 'Address:' labels in the 'Customer Details' section.
- Form Fields:** Points to the input fields for 'Name:', 'Phone:', and 'Address:'.
- Table Widget:** Points to the table structure below the form fields.

**Customer Details**

Name:	
Phone:	
Address:	

OrdNo	ShipNo	Date	Weight
Edit	Edit	Edit	Edit
Edit	Edit	Edit	Edit
Edit	Edit	Edit	Edit
Edit	Edit	Edit	Edit
Edit	Edit	Edit	Edit





# Genero Forms – Form Elements

Form elements have a set of supported attributes that can be applied in the .per definition for presentation and behavior.

The Genero documentation lists the valid attributes for all the form elements, and contains an Attributes List.



# Genero Forms – Form Elements - The Widgets

EDIT

BUTTON

BUTTONEDIT

CANVAS

COMBOBOX

CHECKBOX

DATEEDIT

DATETIMEEDIT

GROUP

IMAGE

LABEL

PROGRESSBAR

RADIOGROUP

SCROLLGRID

SLIDER

SPINEDIT

TABLE


TEXTEDIT

TIMEEDIT



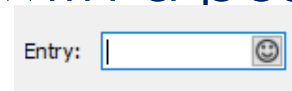
# Genero Forms – The Widgets

A **EDIT** item is a common edit field.



```
EDIT item-tag = field-name[,attribute-list];
```

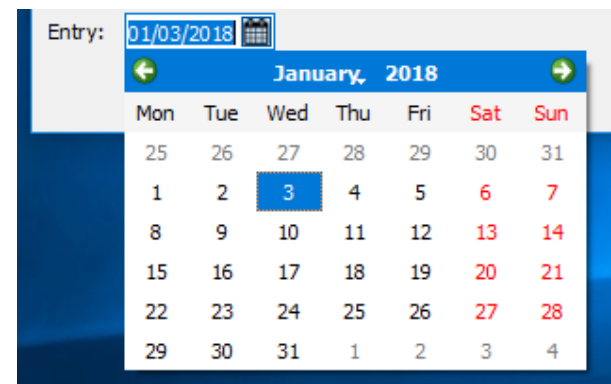
A **BUTTONEDIT** defines an edit field with a push button that can trigger an action.



```
BUTTONEDIT f001 = customer.state,REQUIRED,  
                IMAGE="smiley",ACTION=zoom;
```

A **DateEdit** defines an edit box with a button that opens a calendar.

```
DATEEDIT item-tag =  
    field-name  
    [,attribute-list];
```



January, 2018						
Mon	Tue	Wed	Thu	Fri	Sat	Sun
25	26	27	28	29	30	31
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

# Genero Forms – Widgets

The **ProgressBar** item type defines a horizontal bar with a progress indicator.

```
PROGRESSBAR r1 = FORMONLY.r1 TYPE INTEGER,  
                VALUEMIN=0, VALUEMAX=100;
```

The **SLIDER** item type defines a horizontal or vertical slider.

```
SLIDER f01 = workstate.duration,  
           VALUEMIN=0, VALUEMAX=5,  
           STEP=1;
```



# Genero Forms – Widgets

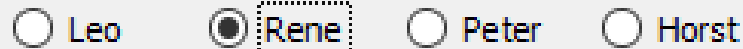
A **CHECKBOX** defines a boolean entry with a box and a text label.

```
CHECKBOX check1= formonly.check1,  
valueChecked="Y",  
valueUnchecked="N",  
text="Confirm"
```



A **RADIOGROUP** defines a set of radio buttons.

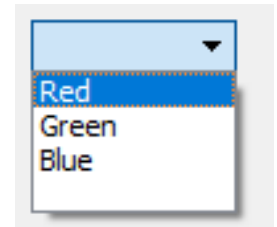
```
RADIOGROUP r1 = formonly.r1,  
items=((0, "Leo"), (1, "Rene"),  
(2, "Peter"), (3, "Horst")),  
ORIENTATION=HORIZONTAL;
```



# Genero Forms – Widgets

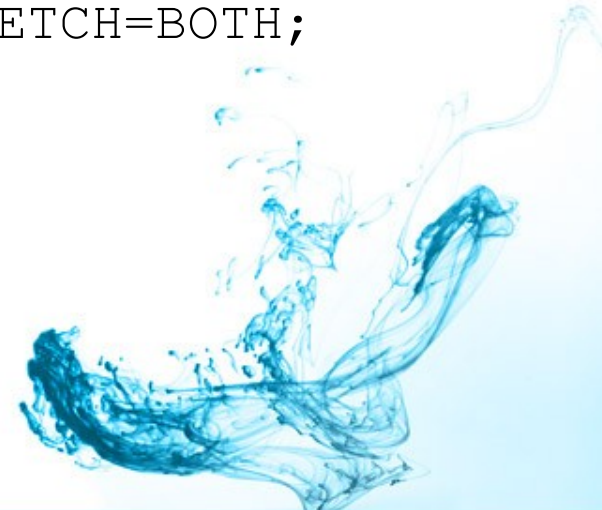
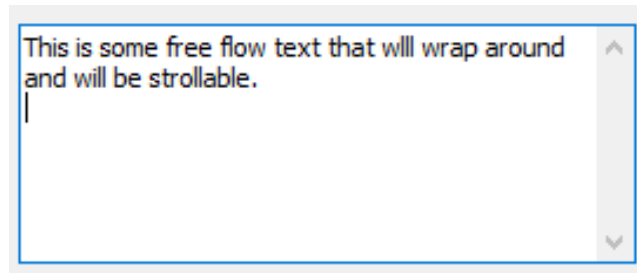
The **COMBOBOX** item type defines a line-edit with a drop-down list of values.

```
COMBOBOX cb = FORMONLY.l_colours,  
  ITEMS=( (1, "Red"), (2, "Green"), (3, "Blue"));
```



The **TEXTEDIT** form item type defines a multi-line edit form field.

```
TEXTEDIT f1 = customer.comments, STRETCH=BOTH;
```



# Genero Forms – Widgets

An IMAGE item defines an area in which you can display an image from a pixel-map file.

Form Field Images – for images that change often.

```
IMAGE f01 = contact.picture,  
          HEIGHT=400 PIXELS,  
          WIDTH=200 PIXELS,  
          STRETCH=BOTH;
```

Static Images – for images that do not change.

```
IMAGE f01 : logo, IMAGE="fjs.gif",  
          STRETCH=BOTH;
```



# Genero Forms – Widgets

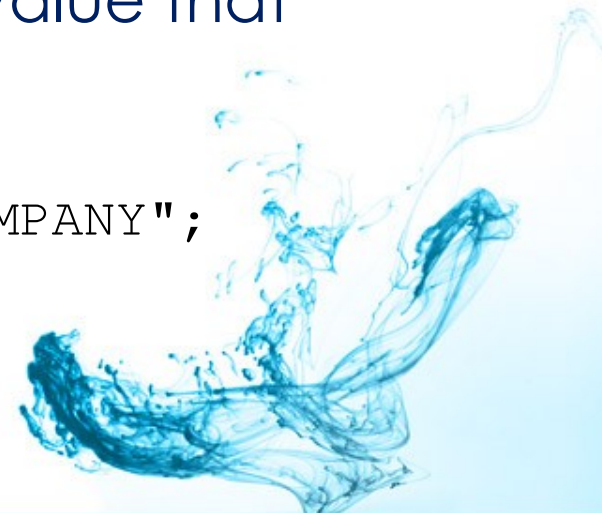
The LABEL form item defines a simple text area to display read-only data. There are two types, and each has unique syntax:

Form Field Labels - for labels that have a value that changes often, such as data retrieved from a database.

```
LABEL custid = customer.customer_id;
```

Static Labels - for labels that have a value that does not change.

```
LABEL lab1 : copyright,  
TEXT = "2003 ABC COMPANY";
```





# Genero Forms – Widgets

A **BUTTON** is an action view, not associated with database columns and therefore using the static notation.

```
BUTTON btn1 : help, TEXT="Click me",  
                IMAGE="question";
```

A **CANVAS** item defines an area in which you can draw shapes.

```
CANVAS cvs1 : canvasarea1;
```



# Genero Forms – Widgets

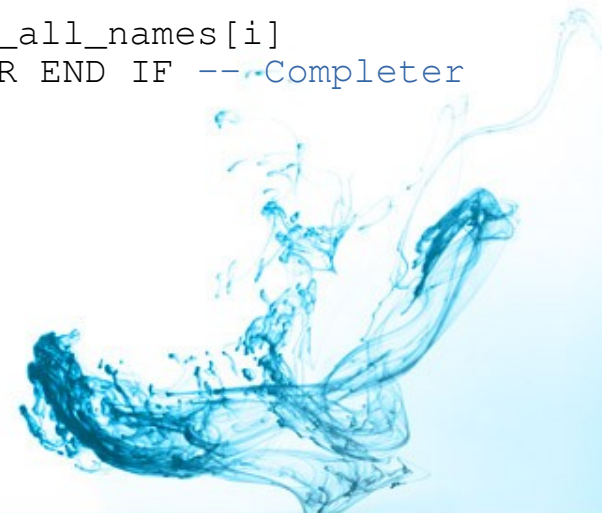
The COMPLETER attribute provides a way to do auto completion, ie a list of valid results based on an array of data that's filtered as the user types.

ATTRIBUTES

```
f1 = FORMONLY.f1, COMPLETER;
```

In the 4gl code:

```
FUNCTION set_completer(l_d ui.Dialog, l_in_str STRING)
  DEFINE l_items DYNAMIC ARRAY OF STRING
  DEFINE i INTEGER
  IF l_in_str.getLength() > 0 THEN
    FOR i = 1 TO m_all_names.getLength()
      IF UPSHIFT(m_all_names[i])
        MATCHES UPSHIFT(l_in_str.append("*"))
      THEN -- case insensitive filter
        LET l_items[ l_items.getLength() + 1 ] = m_all_names[i]
        IF l_items.getLength() == 50 THEN EXIT FOR END IF --Completer
        is limited to 50 items
      END IF
    END FOR
  END IF
  CALL l_d.setCompleterItems(l_items)
END FUNCTION
```



## Do Chapter 2 - Exercise 4



# Genero Forms – Additional Form Elements

Topmenus and Toolbars are views for actions that can trigger events in your code.

Top menu

Toolbar

The screenshot displays the 'NGJM's Demos' application window, titled '03/01/2018: materialDesignTest 3.1 - Material Design Test'. The window features a top menu bar with 'File', 'Edit', 'Option', and 'Help'. Below the menu is a toolbar with icons for 'Quit', 'Cut', 'Copy', 'Paste', 'Show Form', 'Inactive', 'Message', 'Error', 'Model Window', 'ProgressBar', and 'Prog Bar 50'. The main content area is divided into several sections:

- Form Fields:** Includes 'ButtonEdit' (Active), 'DateEdit' (03/01/2018), 'Combo' (Red), 'Readonly' (Inactive), and 'Ok?' checkbox.
- Group #1:** Contains 'Live Field' (Active) and 'Dead Field' (Inactive).
- Group #2:** Contains 'Live Field' (Active) and 'Dead Field' (Inactive).
- Table:** A table with columns 'Title1' and 'Fld2' containing 5 rows of data.
- Footer:** Includes 'Quit', 'Message', and 'Error' buttons.

The right sidebar shows 'Page1' and 'Page2' tabs, with 'Page1' selected. It contains the text 'Some text on a page.' and a progress bar. The bottom right corner displays 'OVR'.

Title1	Fld2
Row 1	1
Row 2	2
Row 3	3
Row 4	4
Row 5	5

# Genero Forms – Additional Form Elements

## ACTION DEFAULTS

### ACTION DEFAULTS

```
ACTION action_id  
( action_attribute [,...] )  
  [...]  
END
```

## TOPMENU

## TOOLBAR



# Genero Forms – Compiling

Form definitions in text files (.per files) can be compiled from the command line using the tool **fglform**:

```
fglform custform.per
```

Genero Studio has menu options and icons that allow you to validate or compile form definitions.

The resulting runtime form file is an XML file with a .42f extension.



# Genero Forms – Summary

- A form definition file is used to design the form area of a Genero program.
- The definition file is translated into an XML document and loaded into the **AUI** at runtime.
- The **LAYOUT** section in the file manages the layout of the form.
- Form items are used within **GRID**, **SCROLLGRID** and **TABLE** containers to provide fields for static and database-driven values.
- Containers and form items have specific attributes.
- **TopMenus** and **Toolbars** can be defined.
- The **Action Defaults** section can be used to centralize the definition of attributes for action views.



## Do Chapter 2 - Exercise 5





# Q&A



*Intelligent Business Application Infrastructure*