Neil J Martin
neilm@4js.com

Additional Features

Genero BDL 3.10
Genero Studio 3.10

FOUR Js
The Power of Simplicity

January 2018

Additional Features

ON IDLE / CHANGE
The STRING data type
Constants / Types
Dynamic Arrays
Paged Display Arrays
Mutliple Dialogs
Try / Catch
Localized Strings
TextEdit – Find & Spelling
File Transfer
Server Side Extensions

ON CHANGE / ON IDLE

The ON CHANGE & ON IDLE clauses.
Executes a set of instructions when a field is changed by the user. Also see ON ROW CHANGE
Typically used with ComboBox / CheckBox / RadioGroup

```
INPUT BY NAME cust.*
 ON CHANGE cust.stop_flag
   CALL disable_cust( cust.stop_flag )
…
 ON IDLE 300 -- 5 Minutes
   CANCEL INPUT -- Cancel the Input.
END INPUT
```

Can be used in INPUT, INPUT ARRAY, CONSTRUCT and DISPLAY ARRAY statements.

FOUR Js
The Power of Simplicity

STRING's

The String data type can store a character string without a size limitation. String is one of the primitive 4gl types.

```
DEFINE s STRING
```

Warning: Because it has no predefined size, a STRING variable cannot be used in SQL statements.

String Methods

Constants & Types

A constant defines a read-only value identified by a name.

```
CONSTANT  identifier [datatype] = value [,...]
```

where `datatype` can be any of the BDL types (CHAR, INTEGER, ...)

User defined types are useful when you need a structure in multiple functions / modules.

```
TYPE t_my_rec RECORD
   key INTEGER,
   desc STRING
END RECORD
DEFINE my_rec ARRAY [10] OF t_my_rec
```

FOUR Js
The Power of Simplicity

Constants & Types - Example

```
CONSTANT aSize = 100
TYPE r_lookup_values RECORD
  key INTEGER,
  desc STRING
END RECORD
DEFINE arr ARRAY[aSize] OF t_lookup_values
MAIN
  DEFINE i INTEGER
  FOR i=1 TO aSize
    LET arr[i].key = i
    LET arr[i].desc = “Value “,i USING "<<<"
  END FOR
  OPEN WINDOW w1 WITH FORM “myform"
  DISPLAY ARRAY arr TO sr.* ATTRIBUTE(COUNT=aSize)
...
```

DYNAMIC ARRAY

The standard ARRAY definition has been extended to provide dynamic memory allocation and unlimited size.

Syntax 1: Static array definition

```
   ARRAY [ intconst [,intconst
[,intconst]] ] OF       datatype
```

Syntax 2: Dynamic array definition

```
   DYNAMIC ARRAY [ WITH DIMENSION rank ] OF
                   datatype
```

FOUR Js
The Power of Simplicity

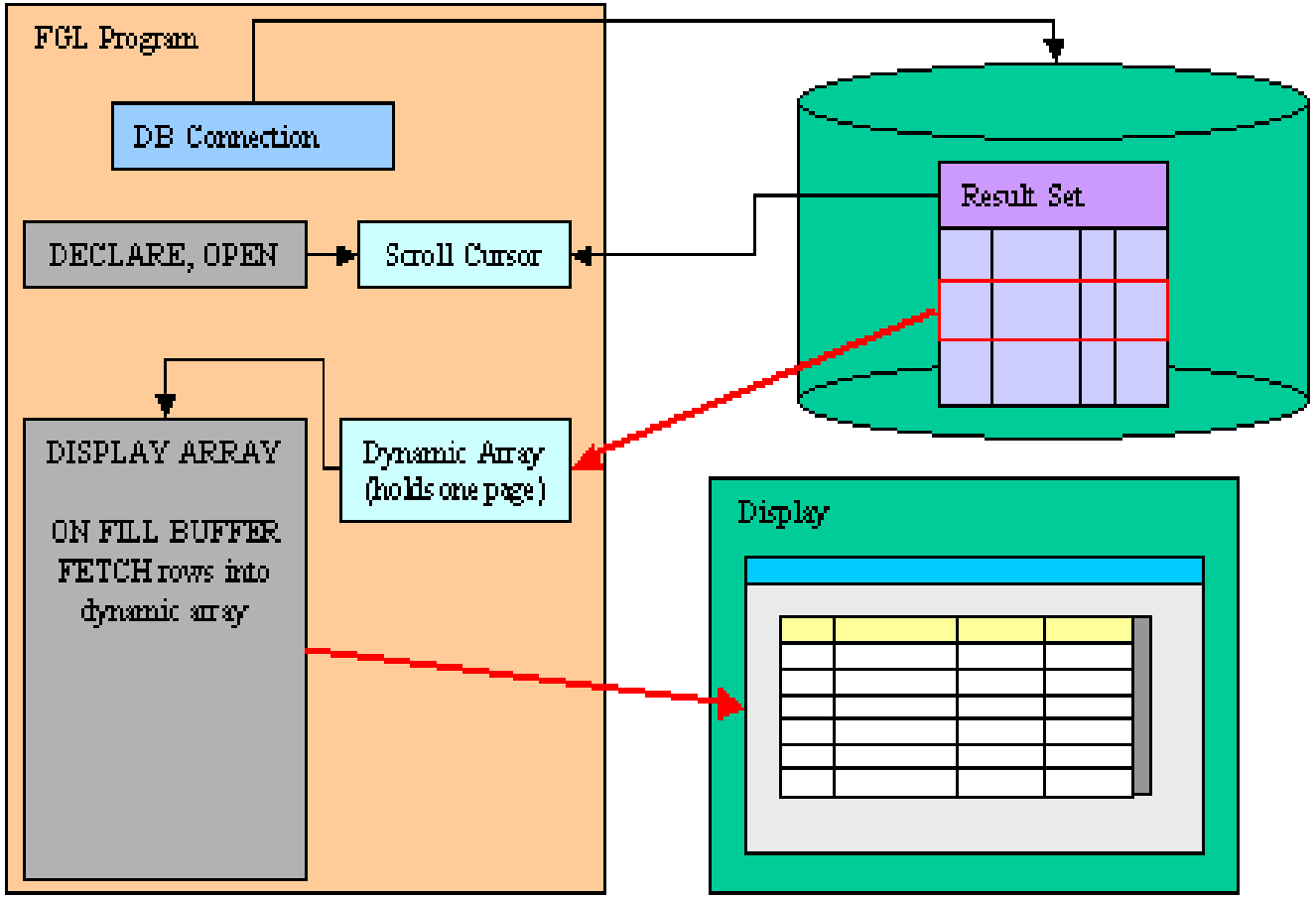
Paged Display Array

The DISPLAY ARRAY instruction can now be used with a dynamic array, which is filled dynamically during the dialog.

This can be used to scroll an infinite list of records without using a huge static array.

- In *paged* mode, the DISPLAY ARRAY instruction uses a DYNAMIC ARRAY defined with a record corresponding to all or part of a screen array of a form.
- The program array is filled dynamically with data rows during the DISPLAY ARRAY execution, as needed.
- The ON FILL BUFFER clause is used to feed the DISPLAY ARRAY instruction with pages of data. The statements in the ON FILL BUFFER clause are executed automatically by the runtime system each time a new page of data is required.

Paged Display Array



Paged Display Array

```
MAIN
  DEFINE arr DYNAMIC ARRAY OF RECORD
       id INTEGER,  fname CHAR(30),  lname CHAR(30)
     END RECORD
  DEFINE cnt, ofs, len INTEGER
  DATABASE stores7
  OPEN FORM f1 FROM "custlist"
  DISPLAY FORM f1
  SELECT COUNT(*) INTO cnt FROM customer
  DECLARE c1 SCROLL CURSOR FOR
        SELECT customer_num, fname, lname FROM customer
  OPEN c1
  DISPLAY ARRAY arr TO srec.* ATTRIBUTES(COUNT=cnt)
   ON FILL BUFFER
     LET ofs = fgl_dialog_getBufferStart()
     LET len = fgl_dialog_getBufferLength()
     FOR i=1 TO len
       FETCH ABSOLUTE ofs+i-1 INTO arr[i].*
     END FOR
  END DISPLAY
  CLOSE c1
END MAIN
```

Multiple Dialogs

A new DIALOG instruction allows multiple interactive statements to be live simultaneously.

See Wizard Demo

TRY / CATCH

Cleaner exception handling with TRY/CATCH instead of WHENEVER ERROR.

```
TRY
  CONNECT TO "custdemo"
CATCH
  CALL fgl_winMessage(
   "Error",
   "Database Connecton Failed!\n"|SQLERRMESSAGE,
   "exclamation")
  EXIT PROGRAM
END TRY
```

FOUR Js
The Power of Simplicity

Localization

Localized Strings allow you to internationalize your application using different languages, and to customize it for specific industry markets in your user population.

Any string that is used in your Genero BDL program, such as messages to be displayed or the text on a form, can be defined as a Localized String.

At runtime, the Localized String is replaced with text stored in a String File. String Files must be compiled, and then deployed at the user site.

You can read in the product tutorial to learn how to use this feature.

FOUR Js
The Power of Simplicity

File Transfer

You can transfer files from to and from the server and client using fgl_getFile and fgl_putFile.

```
CALL fgl_getfile(src STRING, dst STRING)
CALL fgl_putfile(src STRING, dst STRING)
```

These can used with frontCall's dialog windows.

```
CALL ui.interface.frontCall(
  "standard“ ,"openfile",
 ["..\\etc","XML Files","*.xml",
 "GDC Config File"], file_rmt)
```

FOUR Js
The Power of Simplicity

Server Side Extensions

Genero comes with three built-in packages (base, ui, om) but it also comes with some non-built-in packages, these can be loaded and used at runtime using the IMPORT statement.

- **util** - These topics cover the classes for the util package.
- **os** - These topics cover the classes for the os package.
- **com** - The Genero Web Services com package provides classes and methods that allow you to perform tasks associated with creating Services and Clients, and managing the services.
- **xml** - The Genero Web Services XML package provides classes and methods to handle any kind of XML documents, including documents with namespaces.
- **security** - The Genero Web Services security package provides classes and methods to support basic cryptographic features.

Util - Package

This package contains the following classes:

- **Date** - provides DATE data-type related utility methods.
- **Datetime** - provides DATETIME data-type related utility methods.
- **Integer** - provides INTEGER data-type related utility methods.
- **Interval** - provides INTERVAL data-type related utility methods.
- **Strings** - provides STRING data-type related utility methods.
- **Math** - provides basic mathematical functions based on floating point numbers (FLOAT).
- **JSON** - provides a basic interface to convert program variable values to/from JSON data.
- **JSONObject** - provides methods to handle an structured data object following the JSON string syntax.
- **JSONArray** - provides methods to handle an array of values, following the JSON string syntax.

Util - Package

An example of using one of the 'util' package classes.

```
IMPORT util
MAIN
 DEFINE x INTEGER

 DISPLAY "Pi=", util.Math.pi()
 CALL util.Math.srand()
 FOR x = 1 TO 10
    DISPLAY util.Math.rand(50)
 END FOR
```

OS - Package

This package contains only the Path class, it has numerous methods for handling OS level tasks in a platform indepedant way. Eg:

```
IMPORT os
MAIN
  DEFINE l_fname, l_path STRING
  LET l_dir = ARG_VAL(1)
  LET l_fname = ARG_VAL(2)
  IF os.path.chdir( l_dir ) THEN
    IF os.path.exists( l_fname ) THEN
      DISPLAY “File ‘”||l_fname||”’ Exists:”
      DISPLAY “Last Accessed:”|
       os.path.atime(l_fname)
      EXIT PROGRAM
    END IF
  END IF
```

FOUR Js
The Power of Simplicity

Q&A

Intelligent Business Application Infrastructure