50 Things You May Not Know You Can Do With The 4GL

BUSINESS MAKING PROGRESS™

Gus Björklund. Progress.

PUG Challenge Americas
Westford, MA
June 5 to 8, 2011

PROGRESS SOFTWARE

Agenda

- A smörgåsbord of large and small topics
- Having nothing to do with each other
- In no particular order

Credit

- I didn't think all this up myself.
  - Greg Higgins,
  - Dmitri Levin,
  - Dustin Grau,
  - Tom Bascom,
  - Dan Foreman,
- and others came up with some of these

Call a dynamic shared library function

(Windows .DLL
or
UNIX/Linux .so)

Shared library call example

```
define variable x as integer no-undo.

procedure putenv external "/lib64/libc.so.6":
  define input  parameter env as character.
  define return parameter x   as long.
end.

run putenv( "XYZZY=pflugh", output x ).
display os-getenv( "XYZZY" ).

os-command value( 'echo "$XYZZY"' ).

return.
```

This code was gratuitously stolen from Tom Bascom. He has lots more.

Get Process Identifiers

Using input through

```
define variable pid as
            character no-undo.

input through "echo $PPID".
import pid.

input close.
```

Using UNIX/Linux C library call

```
procedure getpid external
              "/usr/lib/glibc.so" cdecl:

   define return parameter pid
                 as long no-undo.

end procedure.

/* then to use it: */

def var p as integer no-undo.

p = getpid ().
```

Using Windows kernel library call

```
procedure GetCurrentProcessId external
                            "kernel32.dll":

   define return parameter pid as long.

end procedure.


def var p as integer no-undo.
run GetCurrentProcessId (output p).
```

Using Database VST

```
def var p as integer no-undo.
find first _myconnection no-lock.
p = _myconnection._myconn-pid.
```

BUSINESS MAKING PROGRESS™

Time Management

PROGRESS SOFTWARE

Date/Time related stuff

- Data types
  - DATE
  - DATETIME
  - DATETIME-TZ
  - INT64
- Session attributes
  - SESSION:TIMEZONE
  - SESSION:DISPLAY-TIMEZONE

Date/Time related stuff

- "constructor" functions

```
d = date (2011, 6, 7)

dt = datetime (2011, 6, 7, 11, 15, 0, 0)

dtz = datetime-tz (2011, 6, 7, 11, 15, 0, -240)
```

ABL Calendar

- Based on Gregorian Calendar
- Epoch Date

  1 January – 4713 at 00:00:00
- Units

DATE datatype:	days
DATETIME:	milliseconds
DATETIME-TZ:	milliseconds

Different Calendars

- UNIX time
  - epoch is Jan 1, 1970 at 00:00:00
  - unit is seconds
- JMS time
  - epoch is Jan 1, 1970 at 00:00:00
  - unit is milliseconds
- Windows time
  - epoch is Jan 1, 1601 at 00:00:00
  - unit is centinanoseconds (100 nanoseconds) aka "ticks"

Useful Time Constants

Number	Description
116 444 736 000 000 000	ticks from 1/1/1601 to 1/1/1970
11 644 473 600 000	milliseconds from 1/1/1601 to 1/1/1970
2 305 814	days from 1/1/- 4713 to 1/1/1601
2 440 588	days from 1/1/- 4713 to 1/1/1970
134 774	days from 1/1/1601 to 1/1/1970
210 866 889 600	seconds from 1/1/- 4713 to 1/1/1970
3 600	seconds in 1 hour
86 400	seconds in 1 day
31 536 000	seconds in 365 days

Time arithmetic is easy

with datetime and datetime-tz data types
arithmetic units is milliseconds

```
def var startTime as datetime.
def var endTime as datetime.
def var i as int64.

i = endTime - startTime.

endTime = startTime + i.
```

Arithmetic in other units

```
def var startTime as datetime.
def var endTime as datetime.
def var nSecs as int64.
def var nDays as int64.

nSecs = (endTime – startTime) / 1000.

nDays = (endTime – startTime) / 86400000.

/* but this is too hard !!! */
```

INTERVAL: A useful function

i = INTERVAL (endTime, startTime, units).

startTime, endTime are expressions of type
DATE, DATETIME, or DATETIME-TZ

units is a character expression evaluating to one of
"years", "months", "weeks", "days",
"hours", "minutes", "milliseconds"

Changing Times

From Windows to DATETIME:

0. convert from ticks to milliseconds
1. adjust for epoch difference

```
def var wintime as int64 no-undo.
def var dt as datetime no-undo.

wintime = wintime / 10000.
dt = add-interval (datetime (1, 1, 1601, 0, 0, 0, 0),
                   wintime, "milliseconds").
```

Changing Times

From DATETIME-TZ to UNIX:

0) adjust for epoch difference in seconds

```
def var dt as datetime-tz no-undo.
def var unixTime as int64 no-undo.

unixTime = interval (dt, DATETIME-TZ (1, 1, 1970,
                     0, 0, 0, 0, 0), "seconds").
```

Time Zones

DATETIME-TZ data type

- milliseconds from epoch
- stored as GMT
- with originating session time zone offset
  - (in minutes)

```
def var tzoffset as int no-undo.
tzoffset = timezone (dt-tz expression).
```

gives you the timezone offset

```
dtz = datetime-tz (dtz, tzoffset)
```

to change a timezone offset

Time Zones

DATETIME-TZ:

database indexing ignores timezone

arithmetic ignores timezone

comparison operators (>, <, >=, <=, =, <>) ignore timezone

Time Zones

SESSION:DISPLAY-TIMEZONE

integer timezone offset used for formatting

initialized to ?

when ? then SESSION:TIMEZONE is used instead.

Time Zones

SESSION:TIMEZONE

integer session timezone offset

initialized to ?

set with timezone function:

```
SESSION:TIMEZONE = TIMEZONE.
```

Some other things

Import data thruough stdin, stdout

On UNIX and Linux:

```
cat cdata.txt | pro -b -p import.p | cat
```

On Windows:

```
type cdata.txt | pro -b -p import.p | more
```

the program imp.p:

```
def var c as char no-undo.
import cv.
put unformatted string (c).
```

How many BI clusters exist?

```
find _AreaStatus where
    _AreaStatus-Areanum = 3.

find _dbStatus

display _AreaStatus-Hiwater *
        _dbStatus._DbStatus-BiBlkSize /
        _dbStatus-BiClSize /
        1024
        .
```

Can’t tell how many are active though

To get formatted data into Excel

Excel can load HTML

Create an HTML table text file
Use one HTML table record per table row
One cell per field

Sample:

```
<table border=0 cellpadding=0 cellspacing=0 width=1034>
 <col width=146>
 <col width=185>
 <col width=159>
 <col width=181>
 <tr height=13>
  <td>Marv Stone</td>
  <td>Systems Engineering</td>
  <td>Progress Software</td>
  <td>marv@example.com</td>
 </tr>
</table>
```

Sample:

```
<table border=0 cellpadding=0 cellspacing=0 width=1034>
 <col width=146>
 <col width=185>
 <col width=159>
 <col width=181>
 <tr height=13>
  <td>Marv Stone</td>
  <td>Systems Engineering</td>
  <td>Progress Software</td>
  <td>marv@example.com</td>
 </tr>
</table>
```

How much space is used?

```
for each _AreaStatus where
   ( not _AreaStatus-Areaname matches "*After Image Area*" )
   no-lock:

 display
  _AreaStatus-Areanum  format ">>>" column-label "Num"
  _AreaStatus-Areaname format "x(20)" column-label "Area Name"
  _AreaStatus-Totblocks column-label "Tot blocks"
  _AreaStatus-Hiwater column-label "High water mark"
  _AreaStatus-Hiwater /
          _AreaStatus-Totblocks * 100 column-label "% use"
  _AreaStatus-Extents  format ">>>" column-label "Num Extents"
  _AreaStatus-Freenum column-label "Free num"
  _AreaStatus-Rmnum   column-label "RM num"
  .
end.
```

What tables are being used?

```
find first _MyConnection no-lock.
for each _UserTableStat where
  _UserTableStat-Conn = _MyConnection._MyConn-UserId
  no-lock:

  find _file where _file-num = _UserTableStat-Num
       no-lock no-error.
  if available _file then
  display
    _UserTableStat._UserTableStat-Num
    _file-name format "x(20)"
    _UserTableStat-read   format ">>>>>>>>>>"
    _UserTableStat-create format ">>>>>>>>>>"
    _UserTableStat-update format ">>>>>>>>>>"
    _UserTableStat-delete format ">>>>>>>>>>".
end.
```

```
Table # File-Name       read create update delete
::::::: ::::::::::::: :::::: :::::: :::::: ::::::
      1 Invoice                          5
      2 Customer           2             1
      3 Item
      4 Order              6
      5 Order-Line                      21
      6 Salesrep
      7 State             52
      8 Local-Default
      9 Ref-Call
```

What indexes are being used?

```
Same technique as for tables, but read
the data from the _UserIndexStat table
```

Which areas are tables in?

```
for each _StorageObject no-lock
   where _StorageObject._Object-type = 1 and
             _StorageObject._Area-number > 6
  find _Area
    where _Area._Area-number = _StorageObject._Area-number
    no-lock no-error.
  find _File
    where _File._File-number = _StorageObject._Object-number
    no-lock no-error.

  display
    _StorageObject._Area-number format ">>9”
      column-label "Area"
    _Area._Area-name format "x(30)" column-label "Name"
    _File._File-nam when available _File  column-label "Table".
end.
```

List tables by storage area

```
for each _Area, each _Storageobject
          where (_Storageobject._Area-number = _Area._Area-number),

     each _File
          where (_File._File-Number = _Storageobject._Object-number)
              and (_File._File-Number > 0)

     break by _File._File-name:

     display _Area._Area-name _File._File-name.
end.
```

Listing of tables by storage area

Area-name	File-Name
Schema Area	agedar
Schema Area	agedar
Schema Area	customer
Schema Area	customer
Schema Area	item
Schema Area	item
Schema Area	monthly
Schema Area	monthly

List tables by storage area

```
for each _Area, each _Storageobject
          where (_Storageobject._Area-number = _Area._Area-number),

     each _File
          where (_File._File-Number = _Storageobject._Object-number)
              and (_File._File-Number > 0)

     break by _File._File-name:

     display _Area._Area-name _File._File-name.
end.
```

List tables by storage area 2

```
for each _Area, each _Storageobject
          where (_Storageobject._Area-number = _Area._Area-number),

     each _File
          where (_File._File-Number = _Storageobject._Object-number)
              and (_File._File-Number > 0)
              and (_StorageObject._Object-type eq 1)

     break by _File._File-name:

     display _Area._Area-name _File._File-name.
end.
```

List indexes by storage area and table

```
for each _Area, each _Storageobject
          where (_Storageobject._Area-number = _Area._Area-number
                    and (_StorageObject._Object-type eq 2) ,

    each _Index
          where (_Index._Idx-num = _Storageobject._Object-number):

    find _File of _Index.
    if (_File._File-number > 0) then
       display _Area._Area-name _File._File-name _Index._Index-name.
end.
```

List Tables and Their Fields

```
output to tables.txt.
for each _file
    where (0 < _file-num):

    put _file-name skip.
    for each _field of _file:
        put "     " _field-name skip.
    end.
    put "" skip.
end.
output close.
```

Table and fields

Invoice
- Adjustment
- Amount
- Cust-Num
- Invoice-Date
- Invoice-Num
- Order-Num
- Ship-Charge
- Total-Paid

Customer
- Address
- Address2
- Balance
- City

You can do range checks in CASE statements

```
DEF VAR MyVar AS INT.
MyVar = RANDOM(-10,11).

CASE TRUE:
   WHEN MyVar LE 10 AND MyVar GT 1 THEN
      MESSAGE "case1" MyVar VIEW-AS ALERT-BOX.
   WHEN MyVar LE 1  AND MyVar GT 0 THEN
      MESSAGE "case2" MyVar VIEW-AS ALERT-BOX.
   WHEN MyVar LE 0 THEN
      MESSAGE "case3" MyVar VIEW-AS ALERT-BOX.

   OTHERWISE
      MESSAGE "case4" MyVar VIEW-AS ALERT-BOX.

END CASE.
```

For dynamic query result fields, instead of this:

```
REPEAT:
   hQuery:GET-NEXT().
   IF hQuery:QUERY-OFF-END THEN LEAVE.
   hBufferField1 = hBuffer:BUFFER-FIELD('Name').
   hBufferField2 = hBuffer:BUFFER-FIELD('CustomerCode').

   DISPLAY hBufferField1:BUFFER-VALUE
           hBufferField2:BUFFER-VALUE.
END.
```

You can do it this way

```
REPEAT:
   hQuery:GET-NEXT().
   IF hQuery:QUERY-OFF-END THEN LEAVE.

   DISPLAY hBuffer::Name
           hBuffer::CustomerCode.
END.
```

Use PUBLISH for debugging

Instead of adding and deleting MESSAGE statements in your code for debugging purposes,
add PUBLISH statements and leave them in there forever:

At runtime, you can SUBSCRIBE to this information when you need it, even in production, and decide what you do with it.

You can DISPLAY the information in a Window.
Write it to a log file (on AppServer or WebSpeed).

The overhead is minimal if you don't subscribe.

```
PUBLISH “message” (PROGRAM-NAME(1), <level>,
                   <message>).
```

PUBLISH from classes:

You can PUBLISH debug messages from classes using the following syntax:

```
PUBLISH “message” FROM (SESSION:FIRST-PROCEDURE)
          (PROGRAM-NAME(1), <level>, <message>).
```

You can process these messages in exactly the same way as from a procedure

PROGRAM-NAME(1) returns the name of the class.

Make sure there is a SESSION:FIRST-PROCEDURE

To send email from WebSpeed

Use smtp server that is built in to Microsoft IIS.
It is very simple to use. Does not need usercode/password setup and is very fast.

```
def var chMessage as com-handle no-undo.
create "CDO.Message" chMessage.
chMessage:Subject = "Test Subject".
chMessage:From = "TestFrom@test.com".
chMessage:To = "TestTo@test.com".
chMessage:TextBody = "Test Body".
chMessage:Send().
release object chMessage.
```

Call .Net assemblies from 4GL

Regenerate the .Net assembly with

"register for COM Interop" = true

in Build settings.

That will generate .tlb (Type library). Now you can use that from Progress in the same manner as a .dll.

If you don't own the source code to regenerate, you can code a .Net wrapper around dll and expose the wrapper as type library. This is a good way to get functionality in Progress that is readily available in .Net.

Are we up to 50 yet ?

With your OpenEdge install, there are variety of functions and programs in the
$DLC/src/samples folder.

Examples includes source code for finding weekday, weeknum, get current folder path, get unique numbers, sample code for activex, .Net, sockets etc.

Go read it, you are likely to find something useful. Some of them are good.

Questions

email: gus@progress.com