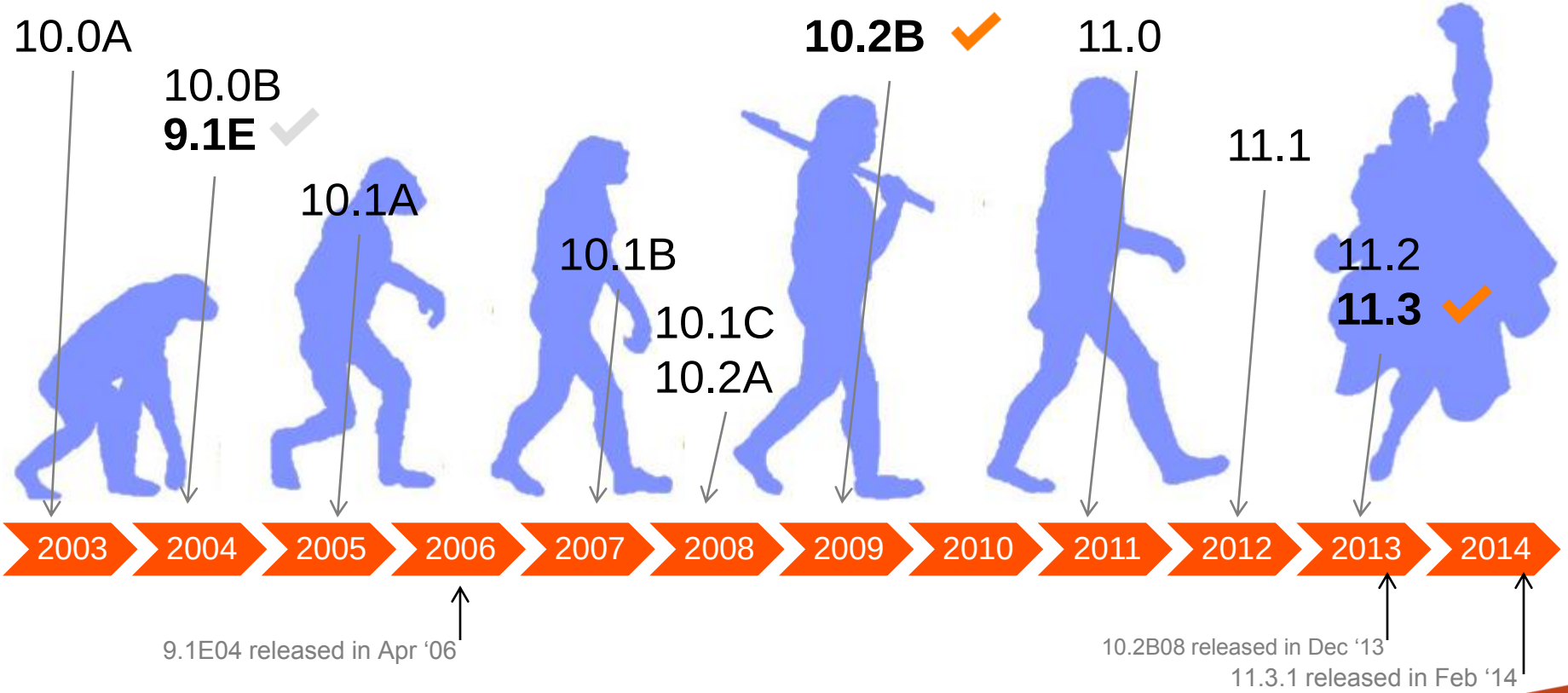
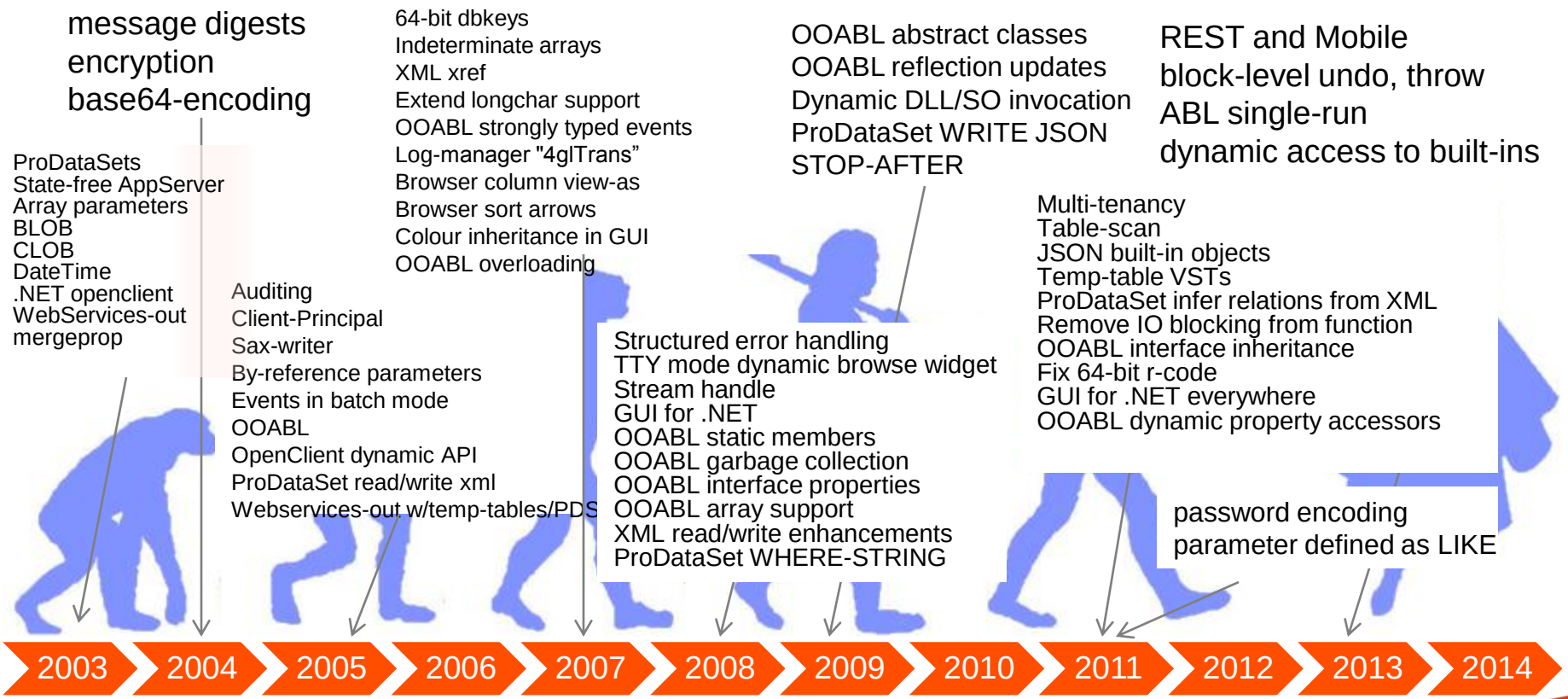
Ten Habits Of Highly-Effective Modern ABL Programmers

Even Covey Only Has 7!

10ish Years' Worth of Progress



10ish Years' Worth of Progress, put another way



In No Particular Order

1. ProDataSets
2. Passing data by-reference
3. mtime > etime (DATETIME)
4. Indeterminate Arrays
5. Arrays as arguments
6. OOABL: taking the leap from spaghetti code to lasagna code
7. Structured error handling
8. Includes & preprocessors
9. JsonObject
10. Static is dynamic is static

In No Particular Order

1. ProDataSets
2. Passin
3. mtime
4. Indeter
5. Arrays
6. OOAB
7. Structu
8. Include
9. JsonO
10. Static is dynamic

This is *my* list. If you don't like it, feel free to ~~shout your own at~~ disagree with me

code

ProDataSets

Age: Prime of life

- FILL() and related events
- Data sources with multi-table field mapping
  - 2 db tables into 1 temp-table? Field names human-readable? No problem
- Parent-child relationships
  - UI binding goodness (navigate through parents, get child nav free!)
- Change tracking
- Row-level events
- Multiple representations
  - Native ABL
  - JSON and XML serialization
  - OpenClient, BPM, JavaScript versions (aka JSDO)
- Gotcha: fair amount of work to do on save (lots of control)

ProDataSets - defining

```
define temp-table ttCustomer no-undo before-table btCustomer
    field Number as integer
    field Name as character                 /* name matches db */
    field CreditLimit as decimal            /* name matches db */
    field Balance as decimal                /* name matches db */
    field FullAddress as character
    field NumOpenOrders as integer          /* name matches db */
    index idx1 as primary unique Number.
define temp-table ttOrder no-undo before-table btOrder /* etc */

define dataset dsCustomer for ttCustomer, ttOrder
      data-relation relCustOrd for ttCustomer, ttOrder
            relation-fields(Number, Number).
define data-source srcCustomer for Sports2000.Customer.
```

ProDataSets – fetching data

```
/* define temp-table, dataset, data-source as above */
define input parameter pcWhereClause as character no-undo.
define output parameter dataset for dsCustomer.

buffer ttCustomer:attach-data-source(data-source srcCustomer:handle).
data-source srcCustomer:fill-where-string = pcWhereClause.


dataset dsCustomer:fill().


buffer ttCustomer:detach-data-source().
```

ProDataSets - defining

```
define temp-table ttCustomer no-undo before-table btCustomer
    field Number as integer
    field Name as character                 /* name matches db */
    field CreditLimit as decimal            /* name matches db */
    field Balance as decimal                /* name matches db */
    field FullAddress as character
    field NumOpenOrders as integer          /* name matches db */
    index idx1 as primary unique Number.
define temp-table ttOrder no-undo before-table btOrder /* etc */

define dataset dsCustomer for ttCustomer, ttOrder
       data-relation relCustOrd for ttCustomer, ttOrder
              relation-fields(Number, Number).
define data-source srcCustomer for Sports2000.Customer.
```

ProDataSets – fetching data (2)

```
/* define temp-table, dataset, data-source as above */
define input parameter pcWhereClause as character no-undo.
define output parameter dataset for dsCustomer.

buffer ttCustomer:attach-data-source(
                  data-source srcCustomer:handle, 'CustNum,Number').
data-source srcCustomer:fill-where-string = pcWhereClause.

dataset dsCustomer:fill().

buffer ttCustomer:detach-data-source().
```

ProDataSets - defining

```
define temp-table ttCustomer no-undo before-table btCustomer
    field Number as integer
    field Name as character                 /* name matches db */
    field CreditLimit as decimal            /* name matches db */
    field Balance as decimal                /* name matches db */
    field FullAddress as character
    field NumOpenOrders as integer          /* name matches db */
    index idx1 as primary unique Number.
define temp-table ttOrder no-undo before-table btOrder /* etc */

define dataset dsCustomer for ttCustomer, ttOrder
       data-relation relCustOrd for ttCustomer, ttOrder
              relation-fields(Number, Number).
define data-source srcCustomer for Sports2000.Customer.
```

ProDataSets – fetching data (3)

```
buffer ttCustomer:set-callback-procedure (
                    'After-Row-Fill', 'AfterRowFill_Customer').
dataset dsCustomer:fill().

procedure AfterRowFill_Customer:
    define input parameter dataset for dsCustomer.
    /* We want this: 14 Oak Park Drive, Bedford, MA 01730, USA */
    ttCustomer.FullAddress = substitute('&1, &2, &3 &5, &4',
        ttCustomer.Address,
        ttCustomer.City,
        ttCustomer.State,
        ttCustomer.Country,
        ttCustomer.PostalCode).
end procedure.
```

ProDataSets – client manipulation (1)

```
buffer ttCustomer:set-callback-procedure(
                        'ROW-CREATE', 'RowCreatedHandler_Customer').

temp-table ttCustomer:tracking-changes = true.
/* do local work - this could be in a grid/browser/screen */
create ttCustomer.
assign ttCustomer.Name = 'Norah'.
temp-table ttCustomer:tracking-changes = false.

procedure RowCreatedHandler_Customer:
    define input parameter dataset for dsCustomer.

    assign ttCustomer.Number = GetNextCustNum().
end procedure.
```

ProDataSets – client manipulation (2)

```
/* continued from ProDataSets – client manipulation (1) */
/* do local work - this could be in a grid/browser/screen */

/* get local changes */
create dataset hTransportDataset.
hTransportDataset:create-like(dataset dsCustomer:handle).
hTransportDataset:get-changes(dataset dsCustomer:handle).

/* update on server */
run SaveCustomers.p on hAppServer (
        input dataset-handle hTransportDataset).
```

ProDataSets – saving data

```
/* define temp-table, dataset, data-source as above */
define input parameter dataset for dsCustomer.

buffer ttCustomer:attach-data-source(
                  data-source srcCustomer:handle, 'CustNum,Number').

/* use before-table */
for each btCustomer:
    buffer btCustomer:save-row-changes().
end.

buffer ttCustomer:detach-data-source().
```

Passing data by-reference

Age: Middle age

- Performance benefits from shallow copy
  - Single copy of a given set of data (TT or PDS)
- Automatic by-value across AVM boundary (ie AppServer)

- Gotcha: careful about cleanup if you're still in same session

Passing by-reference

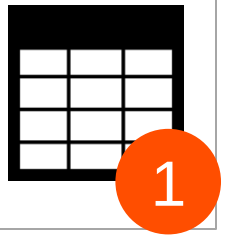
```
/* CreateCustomer.p */
define temp-table ttCustomer no-undo
    field Number as integer
    field Name as character
    /* other fields */
    field NumOpenOrders as integer
    index idx1 as primary unique Number.

create ttCustomer.
assign ttCustomer.Number = 101.

run CalcOpenOrders.p (input table ttCustomer).
```

Passing by-reference (2)

```
/* CalcOpenOrders.p */
define temp-table ttCustomer no-undo
    field Number as integer
    field Name as character
    /* other fields */
    field NumOpenOrders as integer
    index idx1 as primary unique Number.

define input parameter table for ttCustomer.
for each ttCustomer:
  for each Order of ttCustomer: /* pseudo join */
    ttCustomer.NumOpenOrders = ttCustomer.NumOpenOrders + 1.
  end.
end.
```

Passing by-reference (3)

```
/* CreateCustomer.p */
run CalcOpenOrders.p
        (input table ttCustomer).
```

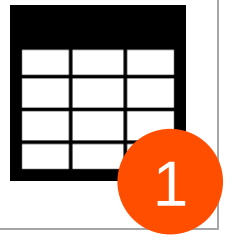



```
/* CalcOpenOrders.p */
define input parameter table for ttCustomer.

for each ttCustomer:
    /* worka worka */
end.
```

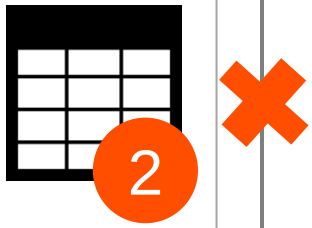
Single session / AVM

Passing by-reference (4)

```
/* CreateCustomer.p */
run CalcOpenOrders.p
        (input table ttCustomer).
```



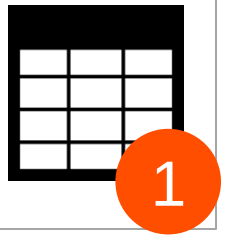

```
/* CalcOpenOrders.p */
define input parameter table for ttCustomer.

for each ttCustomer:
    /* worka worka */
end.
```



Single session / AVM

Passing by-reference (5)

```
/* CreateCustomer.p */
run CalcOpenOrders.p on hdlAppServer
        (input table ttCustomer).
```



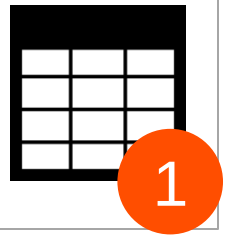
Client session

```
/* CalcOpenOrders.p */
define input parameter table for ttCustomer.

for each ttCustomer:
    /* worka worka */
end.
```

AppServer session

Passing by-reference (6)

```
/* CreateCustomer.p */
run CalcOpenOrders.p on hdlAppServer
        (input table ttCustomer).
```



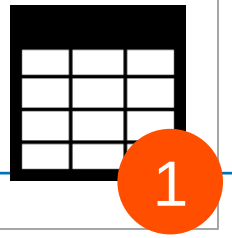
Client session

```
/* CalcOpenOrders.p */
define input parameter table for ttCustomer.

for each ttCustomer:
    /* worka worka */
end.
```

AppServer session

Passing by-reference (7)

```
/* CreateCustomer.p */
run CalcOpenOrders.p
        (input table ttCustomer by-reference).
```

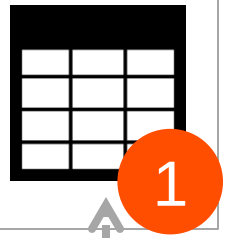


Single session / AVM

```
/* CalcOpenOrders.p */
define input parameter table for ttCustomer.

for each ttCustomer:
    /* worka worka */
end.
```

Passing by-reference (8)

```
/* CreateCustomer.p */
run CalcOpenOrders.p
        (input table ttCustomer by-reference).
```

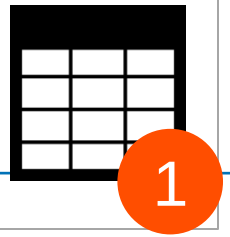



```
/* CalcOpenOrders.p */
define input parameter table for ttCustomer.

for each ttCustomer:
    /* worka worka */
end.
```

Single session / AVM

Passing by-reference (9)

```
/* CreateCustomer.p */
run CalcOpenOrders.p on hdlAppServer
        (input table ttCustomer by-reference).
```



Client session

```
/* CalcOpenOrders.p */
define input parameter table for ttCustomer.

for each ttCustomer:
    /* worka worka */
end.
```

AppServer session

MTIME > ETIME

Age: Middle age

- DATETIME & -TZ
- And datetime-y friends
- Intervals, millisecond precision, timezones

- Gotchas: Still need to store/manage timezones yourself

MTIME > ETIME

```
/* what time is it? */
now.

/* who cares about leap years? */
add-interval(date(2, 28, 2012), 2, 'days')  /* returns 03/01/12 */
add-interval(date(2, 28, 2014), 2, 'days')  /* returns 03/02/14 */

/* find the number of days in a month? */
interval(date(6, 1, 2014), date(7, 1, 2014), 'day').

/* store data in Etc/UTC Americas/New_York */
datetime-tz(now,  000). /* UTC */
datetime-tz(now, -300). /* EST */
datetime-tz(now, -240). /* EDT */

/* measure execution times */
iStart = mtime.
/* do something moderately useful */
iElapsedTime = mtime - iStart.
```

Indeterminate Arrays

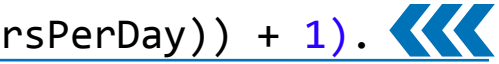
Age: Middle age

- Significantly simpler -> less code
  - No guesswork around size
  - Increased readability
- Array operations faster and cleaner than string operations

Pop quiz: LOOKUP() or INDEX or ENTRY to get the 3rd entry from a string? From an array?

Indeterminate Arrays – delimited string list

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

/* NOTE n – 1 delimiters! */
cOrdersPerDay = fill(',', CalcDaysInMonth(iMonth) - 1).
for each Order where
         OrderDate ge date(iMonth    , 1, year(today)) and
         OrderDate lt date(iMonth + 1, 1, year(today)):
    entry(day(OrderDate), cOrdersPerDay) =
            string(integer(entry(day(OrderDate), cOrdersPerDay)) + 1).
end.
message
  'orders on the 18th=' integer(entry(18, cOrdersPerDay)) skip
  /* runtime error if no 31st */
  'orders on the 31st=' integer(entry(31, cOrdersPerDay))
view-as alert-box.
```



Indeterminate Arrays – fixed extent array

```
/* find the number of orders per day. */
define variable iMonth as integer no-undo.
define variable iOrdersPerDay as integer extent 31 no-undo.

for each Order where
         OrderDate ge date(iMonth,     1, year(today)) and
         OrderDate lt date(iMonth + 1, 1, year(today)):
    assign iOrdersPerDay[day(OrderDate)] =
                  iOrdersPerDay[day(OrderDate)] + 1.
end.

message
  'orders on the 18th=' iOrdersPerDay[18] skip
  /* what about months like Feb or June that have < 31 days? */
  'orders on the 31st=' iOrdersPerDay[31]
 view-as alert-box.
```

Indeterminate Arrays – indeterminate extent

```
define variable iMonth as integer no-undo.
define variable iOrdersPerDay as integer extent no-undo.

extent(iOrdersPerDay) = CalcDaysInMonth(iMonth).

for each Order where
         OrderDate ge date(iMonth,     1, year(today)) and
         OrderDate lt date(iMonth + 1, 1, year(today)):
    assign iOrdersPerDay[day(OrderDate)] =
                  iOrdersPerDay[day(OrderDate)] + 1.
end.

message
  'orders on the 18th=' iOrdersPerDay[18] skip
  'orders on the 31st=' iOrdersPerDay[31]   /* runtime error if no 31st */
view-as alert-box.
```

Array Arguments

Age: Middle age

- Single parameter for many values
  - Single method/function/procedure for all possible array sizes
  - Increased readability & less code
- Super-duper useful with indeterminate arrays

Array Arguments – before

```
DEFINE VARIABLE hClassTable                    AS HANDLE   EXTENT 16   NO-UNDO.

RUN ry/app/rygetclassp.p ON gshAstraAppserver (
        INPUT pcClassName,                      OUTPUT TABLE ttClass APPEND,
        OUTPUT TABLE-HANDLE hClassTable[01], OUTPUT TABLE-HANDLE hClassTable[02],
        OUTPUT TABLE-HANDLE hClassTable[03], OUTPUT TABLE-HANDLE hClassTable[04],
        OUTPUT TABLE-HANDLE hClassTable[05], OUTPUT TABLE-HANDLE hClassTable[06],
        OUTPUT TABLE-HANDLE hClassTable[07], OUTPUT TABLE-HANDLE hClassTable[08],
        OUTPUT TABLE-HANDLE hClassTable[09], OUTPUT TABLE-HANDLE hClassTable[10],
        OUTPUT TABLE-HANDLE hClassTable[11], OUTPUT TABLE-HANDLE hClassTable[12],
        OUTPUT TABLE-HANDLE hClassTable[13], OUTPUT TABLE-HANDLE hClassTable[14],
        OUTPUT TABLE-HANDLE hClassTable[15], OUTPUT TABLE-HANDLE hClassTable[16] ).
```

Array Arguments - after

```
DEFINE VARIABLE hClassTable                    AS HANDLE    EXTENT    NO-UNDO.

RUN ry/app/rygetclassp.p ON gshAstraAppserver (
        INPUT pcClassName,                      OUTPUT TABLE ttClass APPEND,
        OUTPUT TABLE-HANDLE hClassTable ).
```

OOABL: no more spaghetti, just lasagna!

Age: Teenager

- Compile-time type safety
- Garbage collection
- Objects always passed by reference
- Interop with procedures
  - Lets you incrementally add OOABL
  - Necessary for certain cases (callbacks primarily)
  - Pass objects as parameters
- Interfaces allow you to create application skeleton
- Method overrides
  - Single name, different inputs

OOABL – interop with procedures (1)

```
class OpenEdge.Core.XML.SaxReader:
  constructor public SaxReader():
      run OpenEdge/Core/XML/saxreaderfacade.p persistent set mhParserProc
                                (this-object).
  end constructor.

  method public void ParseDocument(input pcXML as longchar):
      run ParseDocument in mhParserProc (pcXML).
  end method.

  method public void StartDocument():
     /* start processing the XML document */
  end method.
end class.
```

OOABL – interop with procedures (2)

```
/** The facade object that handles the callbacks from the SAX parser, and which
    publishes them as typed events. */
define input parameter poSaxReader as SaxReader no-undo.

procedure StartDocument:
    poSaxReader:StartDocument().
end procedure.
```

OOABL – method overrides

```
oModel:SetValue(10).
oModel:SetValue('Norah').
oModel:SetValue(true).
```

```
interface OpenEdge.PresentationLayer.Model.IModel:
    method public logical SetValue(input pcValue as character).
    method public logical SetValue(input pcValue as longchar).
    method public logical SetValue(input phValue as handle).
    method public logical SetValue(input piValue as integer).
    method public logical SetValue(input piValue as int64).
    method public logical SetValue(input pdValue as decimal).
    method public logical SetValue(input ptValue as date).
    method public logical SetValue(input ptValue as datetime).
    method public logical SetValue(input ptValue as datetime-tz).
    method public logical SetValue(input poValue as Object).
end interface.
```

Structured error handling

Age: Teenager

- Catch and throw
- Plays nicely with RETURN ERROR
- Some performance benefits (no code > some code)
- Bubble up errors with
  block-level or routine-level on error undo, throw
- And last, special mention for FINALLY
  - Always runs. Always. Well, almost
  - Doesn't require catch/throw/errors.
  - Great for block, function or procedure cleanup (queries, memptr, disconnect, etc)

Structured error handling - basics

```
/* must be first line of .p or .cls */
block-level on error undo, throw.

define variable iTest as integer no-undo.

iTest = integer('this should fail miserably').

catch oError as Progress.Lang.Error :
    message
    oError:NumMessages  skip /* 0 */
    oError:Severity     skip /* 0 */
    oError:GetMessage(1)
        /* ** Invalid character in numeric input d. (76)*/
    view-as alert-box.
end catch.
```

Structured error handling – basics (2)

```
/* must be first line of .p or .cls */
block-level on error undo, throw.

define variable iTest as integer no-undo.

if iTest eq 0 then
    undo, throw new AppError('iTest should be non-zero').

catch oError as Progress.Lang.AppError :
    message
    oError:ReturnValue  skip /* iTest should be non-zero */
    oError:NumMessages  skip /* 0 */
    oError:Severity     skip /* 0 */
    oError:GetMessage(1)     /*   */
    view-as alert-box.
end catch.
```

Structured error handling – basics (3)

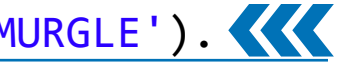
```
case iTest:
    when 0 then undo, throw new AppError('iTest should be non-zero').
    when 2 then undo, throw new CustomError('iTest cannot be 2').
    otherwise /* happy place */.
end case.

catch oCustomError as Org.PugChallenge.CustomError: /* written by me */
    message 'CustomError' view-as alert-box.
end catch.

catch oAppError as Progress.Lang.AppError: /*PSC equiv of RETURN ERROR*/
    message 'AppError' view-as alert-box.
end catch.

catch oError as Progress.Lang.Error:  /* PSC generic error */
    message 'Error' view-as alert-box.
end catch.
```

Structured error handling – interop (1)

```
block-level on error undo, throw.

procedure ReturnsError:
    return error 'AGGLE FLABBLE KLABBLE'.
end procedure.

/* ****  Main Block  **** */
run ReturnsError.

catch oAE as Progress.Lang.AppError :
    message
    oAE:ReturnValue /* AGGLE FLABBLE KLABBLE */
    view-as alert-box.
end catch.
```

Structured error handling – interop (2)

```
block-level on error undo, throw.

procedure ReturnsError:
    return error 'AGGLE FLABBLE KLABBLE'.
end procedure.

procedure ThrowsError:
    return error new Progress.Lang.AppError('GURGLE FLURGLE MURGLE').
end procedure.

/* ****  Main Block  **** */
run ThrowsError.

catch oAE as Progress.Lang.AppError :
    message
    oAE:ReturnValue /* GURGLE FLURGLE MURGLE */
    view-as alert-box.
end catch.
```



Structured error handling – interop (3)

```
block-level on error undo, throw.

procedure ReturnsError:
    return error 'AGGLE FLABBLE KLABBLE'.
end procedure.

procedure ThrowsError:
    return error new Progress.Lang.AppError('GURGLE FLURGLE MURGLE').
end procedure.

/* ****  Main Block  **** */
run ThrowsError no-error.

message
    return-value /* GURGLE FLURGLE MURGLE */
    view-as alert-box.
```

Structured error handling – finally

```
define variable hAppServer as handle no-undo.
define variable cResult as character no-undo.

create server hAppServer.
hAppServer:connect('-AppServer asbroker1 -H localhost').

run foo/bar/baz.p on hAppServer (output cResult).

return cResult.

catch e as Progress.Lang.Error :
    return error e:GetMessage(1).
end catch.
finally:
    if valid-handle(hAppServer) then
        hAppServer:disconnect().
    delete object hAppServer no-error.
end finally.
```

Structured error handling – finally (2)

```
define variable hAppServer as handle no-undo.
define variable cResult as character no-undo.

create server hAppServer.
hAppServer:connect('-AppServer asbroker1 -H localhost').

run foo/bar/baz.p on hAppServer (output cResult).

return cResult.




finally:
    if valid-handle(hAppServer) then
        hAppServer:disconnect().
    delete object hAppServer no-error.
end finally.
```

Includes & preprocessors

Age: Methuselah

- When you need ‘em, you really need ‘em – can work very well with OO types in lieu of generics
- Feasible replacement for code generation for boilerplate code

- Not everyone likes them: “the preprocessor was the worst feature we ever built” (gus)

Includes & preprocessors - boilerplate

```
class OpenEdge.Net.HTTP.MethodEnum inherits EnumMember:
    {EnumMember.i GET       0  MethodEnum}
    {EnumMember.i HEAD      1  MethodEnum}
    {EnumMember.i POST      2  MethodEnum}
    {EnumMember.i PUT       3  MethodEnum}
    {EnumMember.i PATCH     4  MethodEnum}
    {EnumMember.i DELETE    5  MethodEnum}
    {EnumMember.i TRACE     6  MethodEnum}
    {EnumMember.i OPTIONS   7  MethodEnum}
    {EnumMember.i LINK      8  MethodEnum}

  constructor protected MethodEnum(input piValue as integer,
                                   input pcName as character):
    super (input piValue, input pcName).
  end constructor.

  {EnumFromString.i OpenEdge.Net.HTTP.MethodEnum}
end class.
```

Includes & preprocessors - boilerplate

```
define public static property {1} as {3} no-undo
    get:
        if not valid-object ({3}:{1}) then
            {3}:{1} = new {3} ({2}, "{1}":u).

        return {3}:{1}.
    end get.
    private set.

&IF "{&EnumMembers}":U NE "":U &THEN
&GLOBAL-DEFINE EnumMembers {&EnumMembers},{1}
&ELSE
&GLOBAL-DEFINE EnumMembers {1}
&ENDIF

&IF "{&EnumValues}":U NE "":U &THEN
&GLOBAL-DEFINE EnumValues {&EnumValues},{2}
&ELSE
&GLOBAL-DEFINE EnumValues {2}
&ENDIF
```

Includes & preprocessors - boilerplate

```
class OpenEdge.Net.HTTP.MethodEnum inherits EnumMember:
    define public static property GET as MethodEnum no-undo
    get:
        if not valid-object (MethodEnum:GET) then
            MethodEnum:GET = new MethodEnum (0, "GET":U).
        return MethodEnum:GET.
    end get.
    private set.

    define public static property HEAD as MethodEnum no-undo
    get:
        if not valid-object (MethodEnum:HEAD) then
            MethodEnum:HEAD = new MethodEnum (1, "HEAD":U).
        return MethodEnum:HEAD.
    end get.
    private set.
    /* etc */
end class.
```

Includes & preprocessors - generics

```
/* Caller*/
define variable oBE as Business.Logic.IBusinessEntity no-undo.
oBE = {getinstance
          &Name=App.BL.CustomerBE
          &Type=Business.Logic.IBusinessEntity}

/* Generic shim/glue/noddy include */
{getinstance
    cast(ServiceManager:Instance("{&Name}"), {&Type}).
}

/* Actual methods */
class ServiceManager:
    method static public Object Instance(input pcType as character):
        return dynamic-new pcType ().
    end method.
end class.
```

Includes & preprocessors – generics (2)

```
/* Example at https://community.progress.com/technicalusers/f/19/p/965/2317.aspx#
   from Consultingwerk */
{foreach.i System.Windows.Forms.Control oControl in THIS-OBJECT:Controls}
    MESSAGE oControl:Text VIEW-AS ALERT-BOX.
END.

/* foreach.i */
DEFINE VARIABLE {2}           AS {1} NO-UNDO .
DEFINE VARIABLE {2}Enumerator AS System.Collections.IEnumerator NO-UNDO.

ASSIGN {2}Enumerator = {4}:GetEnumerator() .

{2}Enumerator:Reset() .

DO WHILE {2}Enumerator:MoveNext() ON ERROR UNDO, THROW:
    ASSIGN {2} = CAST({2}Enumerator:Current, {1}) .
```

JSONObject

Age: Toddler

- JSON manipulation using built-in classes
  - JsonObject, JsonArray
  - Traverse objects, arrays
- Read/write to/from files, memptrs, temp-tables, etc
- Values converted to native ABL types
- Why?
  - Fantastic for config files: less rigid structure, more readable than XML
  - Lots of internet/www data sent as JSON (J-is-for-JavaScript)

JSONObject

```
define variable oArray  as JsonArray  no-undo.
define variable oObject as JsonObject no-undo.

oArray:Add(10).      /* Add for new property and value */
oArray:Set(1, 12).   /* Set for replacing value */

oArray:Add('Norah').
oArray:Add(true).
oArray:Add(now).      /* Note: JSON has no native date/time type; ABL uses ISO */
tSavedTime = oArray:GetDateTimeTz(4).

oJsonObject:Add('messages', oArray).
oJsonObject:Set('messages', new JsonArray()).

oArray:Read(buffer ttData:handle).
oArray:Write(buffer ttData:handle).

/* Write JSON into the longchar */
oJsonObject:Write(input-output longcharResult).
/* Dump JSON to disk (last 'true' makes it readable) */
oJsonObject:WriteFile(input session:temp-dir + 'data.json', true).
```

Static is dynamic is static

Age: Middle age

- Define specialised stuff & pass to generic code

```
DEFINE <widget>
<widget>:HANDLE and GET-*-HANDLE()
TABLE and TABLE-HANDLE
DATASET and DATASET-HANDLE
```

- Queries, buffers, datasets, widgets
  - Static usually faster at runtime
  - Static usually more readable
  - Dynamic/generic more reusable

Static is dynamic is static – these things are the same

```
procedure AfterRowFill_Customer:
  define input parameter dataset for dsCustomer.
  ttCustomer.FullAddress = substitute('&1, &2, &3 &5, &4',
        ttCustomer.Address, ttCustomer.City, ttCustomer.State,
        ttCustomer.Country, ttCustomer.PostalCode).
end procedure.
/* Both these procedures are specific to a ttCustomer temp-table */
procedure AfterRowFill_Dataset:
  define input parameter dataset-handle for phDataset.
  hCustomer = phDataset:get-buffer-handle('ttCustomer').
  hCustomer::FullAddress = substitute('&1, &2, &3 &5, &4',
        hCustomer::Address, hCustomer::City, hCustomer::State,
        hCustomer::Country, hCustomer::PostalCode).
end procedure.
```

Static is dynamic is static – these things are similar

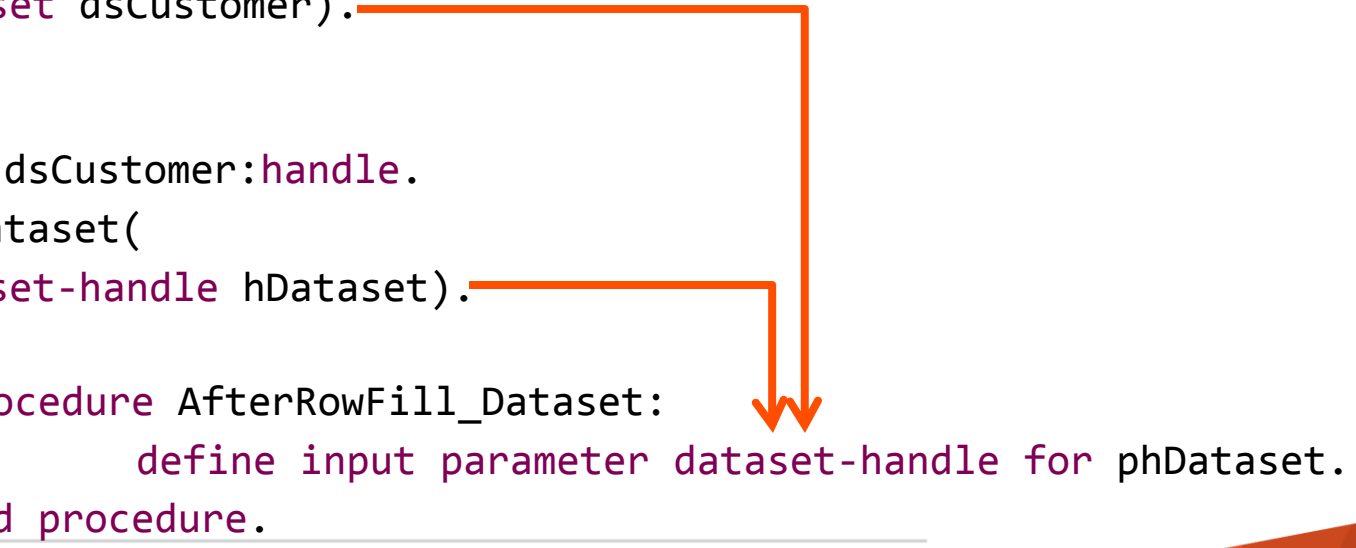
```
procedure AfterRowFill_Dataset:
  define input parameter dataset-handle for phDataset.
  /* alt. loop thru all tables via :num-buffers and get-buffer-handle(n) */
  hBuffer = phDataset:get-buffer-handle(1).
  /* let's say that any table with FullAddress has Address, City etc
     fields capable of building that field */
  hField = hBuffer:buffer-field('FullAddress') no-error.
  if valid-handle(hField) then
        hField = substitute('&1, &2, &3 &5, &4',
                            hBuffer::Address, hBuffer::City,
                            hBuffer::State, hBuffer::Country,
                            hBuffer::PostalCode).
end procedure.
```

Static is dynamic is static – calling

```
define temp-table ttCustomer no-undo /* fields, indexes etc */
define temp-table ttOrder no-undo   /* etc */
define dataset dsCustomer for ttCustomer, ttOrder.

run AfterRowFill_Customer(
        input dataset dsCustomer).


hDataset = dataset dsCustomer:handle.
run AfterRowFill_Customer(
        input dataset-handle hDataset).

                procedure AfterRowFill_Customer:
                        define input parameter dataset for dsCustomer.
                end procedure.
```

Static is dynamic is static – calling (2)

```
define temp-table ttCustomer no-undo /* fields, indexes etc */
define temp-table ttOrder no-undo   /* etc */
define dataset dsCustomer for ttCustomer, ttOrder.

run AfterRowFill_Dataset(
        input dataset dsCustomer).


hDataset = dataset dsCustomer:handle.
run AfterRowFill_Dataset(
        input dataset-handle hDataset).

                procedure AfterRowFill_Dataset:
                        define input parameter dataset-handle for phDataset.
                end procedure.
```



Static is dynamic is static – query

```
define query qryCustomer for Customer, Order.
define query qryDepartment for Department, Employee, Benefits.
run DumpQuery(input query qryCustomer:handle).
run DumpQuery(input query qryDepartment:handle).
procedure DumpQuery:
    define input  parameter phQuery as handle no-undo.
    phQuery:query-open().
    phQuery:get-first().
    do while not phQuery:query-off-end:
        do iBufferLoop = 1 to phQuery:num-buffers:
            hBuffer = phQuery:get-buffer-handle(iBufferLoop).
            do iFieldLoop = 1 to hBuffer:num-fields.
                put unformatted hBuffer:buffer-field(iFieldLoop):string-value() skip.
            end. /* field loop */
        end. /* buffer loop */
        phQuery:get-next().
    end.
    phQuery:query-close().
end procedure.
```

I Lied About There Being 10

11. Things that are not ABL

Immensely Useful Things That Are Not ABL

- A modern editor like PDSOE / Eclipse
  - No, not vi or emacs
- Source control: Git or Mercurial or Roundtable or SVN or …
- Scripted or automated builds / Continuous Integration
- You are not alone: ask – and answer - on PEG / Progress Communities / StackOverflow / Google
- PUG Meetings & - Challenges / Exchange

Summarily

- Older is not necessarily better. Neither is newer.
  - Pick the ABL that, for you, fulfills the promise of making business apps easy to write
- Programming is not religion: keep an open mind and try different things

Relatedly

- Finally, we can catch errors thrown to us! — Paul Guggenheim, Paul Guggenheim & Associates, Inc.
- Fun with OOABL Events — Tim Kuehn, TDKCS
- How I Stopped Using Shared Variables and Learned to Love OO — Tom Bascom, White Star Software
- Workshop: Intro to OOABL — Tim Kuehn, TDK Consulting Services Inc
- Managing Data in an Object World — Mike Fechner, Consultingwerk Ltd.

PROGRESS