MOVE-8: Separating Interface from Logic

John Campbell
White Star Software

Or,
How to get from
What we Have
To
What we Want

BackGround

Strategic Issues

- *Legacy issues used to be in the business process*
- *Now, it's the software*
- *Business can't adapt if legacy software is too hard to change*

The Goal

- *“Rewrite the software*
- *(Implicit: so it can do* ***‘anything’****)*

What is our Purpose

How to create software that is:

- *Competitive*
- *Responsive*
- *Flexible*
- *Multiple Interfaces*
- *Changeable*
- *Maintainable*
- *Functional*
- *Fast*

Problem (cont'd)

- *Market "Demands" change*
- *Sales Force*
  - *Must Have Competitive Products*
- *Functionality*
  - *Often Requires Web*
  - *GUI would add*
    - *Flexibility*
    - *Features*
    - *Power*
  - *Character is often most efficient*

Today’s Software Issues

- *Existing, functional software*
- *Interface*
  - *Character*
  - *User-friendly*
  - *Rich*
  - *Efficient*
- *Robust mechanics*

What is the REAL problem?

The software world has changed, but

Our

- *Understanding*
- *Skills*
- *Tools*

Have not

What are the Questions to Ask?

What Skills do we Have?

- *Classic Progress®*
- *GUI*
- *n-Tier*
- *Web Services*

What are the issues we face?

- Technical
  - Microsoft
  - Progress
  - Oracle
- Business
  - Sales
  - Politics
  - Ignorance

The Developer's Dilemma

- *What ever happened to:*
  - *For each customer: display customer.*

What to think about

- *Interface Objectives*
- *System Architecture*
- *Durability*
  - *Interface*
  - *Environment*
- *Maintainability*
- *Performance*

What's possible

- *Multiple Interfaces*
- *Dynamic or Static*
- *Modular code*
- *Great flexibility*
- *Good performance*

What's not Probable

- *Complete, automated rewrite*
- *Simple porting of old application*
- *Direct translation of old features*

Project Case Study

Project Background

- *Clinical Scheduling software*
- *Robust character interface*
- *Robust mechanics*
- *ASP model*

Objectives

- *Keep interface*
- *Isolate from all DB access*
- *Allow some on-line (web) access*
- *Allow users to continue using character*
- *Deliver choice of on-line, GUI or TTY*

What we did

- *Convert existing app to multi-interface*
- *Chose WebClient™ for web*
- *Character interface to be retained*
- *GUI was client of choice*
- *AppServer™ enabled*

How we did it

- *Analyzed Application*
- *Separated screens into categories*
- *Rewrote some*
- *Templates for others*

Secondary Analysis

- *Looked at code functions*
- *What could be retained*
  - *Data requests*
  - *Validation*
  - *Business Logic*

How to minimize the effort of rewriting code

- *Reproduce the interface*
- *Parse out*
  - *Data requests*
  - *Validation*
- *Extract other logic**
  - *Retain current logic as much as possible*

** Otherwise known as cut and paste*

What we Did

Overview

- *Designed templates*
- *Built tools*
- *Crafted new code*
- *Cut and paste AND automation*

Theory and Process

- *Use a repository as target for current application's information*
    - *Screen Definitions*
    - *Data retrieval*
    - *Other information (logic, etc)*

Populating the Repository

- *Use run-time tool to derive screen information*
- *Use code parser to derive data queries and some other logic*

Screen repository

- *Simple Model:*
  - *Parent table stores frame, table and query information for a screen*
  - *Child table stores primary screen object information (fill-ins) for this frame*
  - *The demo of this model is for single-table, single-record maintenance screens with fill-ins*
    - *(Full application more complex and robust)*

Frame / Table Table

```
Field-Name          Type Format
------------------- ---- ------------
ProgName            char x(20)
TableName           char x(15)
ValidateProgram     char x(20)
FrameName           char X(10)
FrameRow            inte >9
FrameCol            inte >9
FrameWidth          inte >>9
FrameHeight         inte >9
FrameTitle          char X(20)
FrameBox            logi yes/no
QueryPhrase         char X(40)
OneRecord           logi yes/no
```

Storing Frame into Repository

```
assign
hFrame  = self:frame
hField  = hFrame:first-child
hfield  = hfield:first-tab-item.

do while valid-handle(hField):
  if hField:table <> ? then leave.
  hField = hField:next-sibling.
end.
if valid-handle(hField) and hField:table <> ?
   then TableName = hField:table.
find first MaintScreen where MaintScreen.ProgName = vProgName
  and MaintScreen.framename = hFrame:name no-error.
if not available(MaintScreen) then do:
  create MaintScreen.
  assign ProgName     = vProgName
         TableName    = hField:table
         FrameName    = hFrame:name
         FrameRow     = hFrame:row
         FrameCol     = hFrame:column
         FrameWidth   = hFrame:width
         FrameHeight  = hFrame:height
         FrameTItle   = hframe:title
         FrameBox     = hFrame:box.
end.
```

Screen object repository

- *Screen object information*
- *Field Name Table Label*
- *Format Datatype Width*
- *Etc.*
- *Code initiated on "hotkey"*
- *Walked screen widget tree*
- *" TTY Browsers" (and other) not converted*

Prototyping New Screens

- *Using a repository allows prototyping new screens with AppBuilder and storing those screens into the repository*

Field Table

```
Field-Name          Type Format
------------------- ---- ------------
Progname            char x(20)
FrameName           char x(10)
FieldName           char x(15)
FieldRow            deci >9.99
FieldColumn         deci >9.99
FieldFormat         char x(10)
FieldWidth          deci >9.99
FieldLabel          char x(20)
ValidateString      char x(30)
ValidateMessage     char x(40)
Tooltip             char x(40)
HelpString          char x(40)
Maintain            logi yes/no
```

Storing Fields into repository

```
do while valid-handle(hField):
  if  hField:type = "fill-in" then do:
    find first MaintField
      where MaintField.ProgName  = MaintScreen.ProgName
        and MaintField.Framename = hFrame:name
        and  fieldname = hField:name no-error.
    if not available(MaintField) then do:
      create MaintField.
      assign MaintField.ProgName = vProgName.
    end.
    assign
      MaintField.FieldName   = hfield:name
      MaintField.FieldRow    = hField:row
      MaintField.Fieldcolumn = hField:column
      MaintField.FieldWidth  = hField:width
      MaintField.FieldLabel  = hField:label
      MaintField.FieldFormat = hField:format.
  end.
  hField = hField:next-tab-item.
end.
```

Interface Generation

- *Dynamic Browsers*
- *Simple Maintenance Screens*
- *Temp-tables from DB fields*

Frame Generation

```
for each MaintScreen no-lock:
  put unformatted "form " skip.
  for each MaintField no-lock
    where MaintField.progname =  MaintScreen.progname
    and MaintScreen.framename = MaintField.framename :
    put unformatted "t" FieldName " at row "
   MaintField.FieldCol " column " MaintField.FieldCol
skip.
  end.
  put unformatted
    "with " skip
    "row "    framerow skip
    "column " framecol skip
    "size "   framewidth " by " frameheight skip
    if FrameBox then "" else " no-box "  skip
    "side-label " skip
    if session:window-system <> "tty" then "three-d"
    else "" skip
    "frame " MaintScreen.framename "." skip(1)
end.
```

Interface Options

- *Static screen: code generation*
  - *This presentation*
- *Dynamic: uses code template*
  - *2005 presentation on all-dynamic*

Code Parsing

- *Tools:*
  - *Hand-built parser*
  - *JoanJu's ProParse & ProLint*

Parser Overview

- *Look for Data query stuff (for, find …)*

```
sosomt.p|for|91| for each so_mstr no-lock where so_nbr > “a"
```

- *Analyze & store to DB*

```
MaintScreen.QueryPhrase = ‘where so_nbr > “a“’
```

Query Generation

```
for each MaintScreen no-lock:
  . . .
  put unformatted "run get_" maintscreen.tablename ".p"
              "('" MaintScreen.TableName "','"
                MaintScreen.queryphrase ","
                  MaintScreen.OneRecord
         ",input-output table " Temptablename  ")."
end.
```

Methodology

- *Look for Data query stuff (for, find …)*
- *Analyze & store to DB*
- *Convert to consistent selections*
  - *Use queries*
  - *For each and find use same code*
- *Ultimate goal: drive data selections to a temp-table*

Alternatives

- *This demo uses static temp-tables*
  - *Easier to visualize and read in demo*
  - *More concrete for less abstract developers*
- *Could use ProDataSets*
  - *Smaller footprint (1 program)*
  - *Much harder to maintain*
  - *Harder to visualize*
  - *See all-dynamic – 2005 for examples*

Screen and Query Generator Code Samples

Query Generation

```
for each MaintScreen no-lock:
  /* generate a program to get data for this table */
  output to value("{&dirname}get_" + MaintScreen.TableName +
   ".p").

  put unformatted "/* get_"   MaintScreen.TableName  ".p "
   skip
    "Routine to get data based on query from client */ "
     skip(1)
    chr(123)
   'get_data.i &TableName = "' MaintScreen.TableName '"}'
   skip.
  output close.
  /* end data retrieval */
```

Generated Query Routine

```
/* get_so_mstr.p
Routine to get data based on query from client */

{get_data.i &TableName = "so_mstr"}
```

Query Include - Definitions

```
/* get_data.i
Routine to get data based on query from client
*/
define temp-table t{&TableName} like {&TableName}
  field tRowid as rowid.

define input parameter pTableName   as char.
define input parameter pQueryPhrase as char.
define input parameter pOneRecord   as log.

/* note that this is a static temp-table */
define input-output parameter table for t{&TableName}.

define variable hDBQuery   as handle.
define variable hTTBuffer  as handle.
define variable hDBBuffer  as handle.
```

Query Include - Setup

```
assign
hTTBuffer = buffer t{&TableName}:handle
pQueryPhrase = "for each " + pTableName +
" no-lock where " + pqueryphrase.

/* first, create an empty DB buffer structure */
create buffer hDBBuffer for table pTableName.
create query hDBQuery.

/* point the query to the DB table */
hDBQuery:set-buffers(hDBBuffer).

/* get the query ready and open it */
hDBQuery:query-prepare(pQueryPhrase).
hDBQuery:query-open().
```

Query Include - Retrieval

```
repeat:
  hDBQuery:get-next().
  if not hDBQuery:query-off-end then do:
    /* create records in the temp table */
    hTTBuffer:buffer-create().
    /* copy the DB record to the TT */
    hTTBuffer:buffer-copy(hDBBuffer).
    /* then the rowid of the DB record */
    hTTBuffer:buffer-field("trowid"):buffer-value =
    hDBBuffer:rowid.
    if pOneRecord then leave.
  end.
  else leave.
end.
```

Screen Generation

Program Setup

```
for each MaintScreen no-lock:

  /* generate a program to display and retrieve data */
  output to value(MaintScreen.progname + ".p").
  TempTableName = "t" + MaintScreen.TableName .
  put unformatted "define temp-table " TempTableName
  " like "
    MaintScreen.TableName skip
    "field tRowid as rowid. "
  skip.

  put unformatted "." skip(1) "form " skip.
```

Screen Generation

```
put  "form " skip.
  for each maintfield
  where maintfield.progname =  MaintScreen.progname
    and MaintScreen.framename = maintfield.framename no-lock:

    put unformatted tempTableName "." FieldName " at "
        maintfield.fieldcol skip.
  end.
  put unformatted
    "with " skip
    " row "    framerow skip
    " column " framecol skip
    " size "   framewidth " by " frameheight skip
    if FrameBox then "" else " no-box "  skip
    " side-label " skip
    if session:window-system <> "tty" then " three-d" else ""
   skip
    " frame " MaintScreen.framename "." skip(1).
```

Data Retrieval

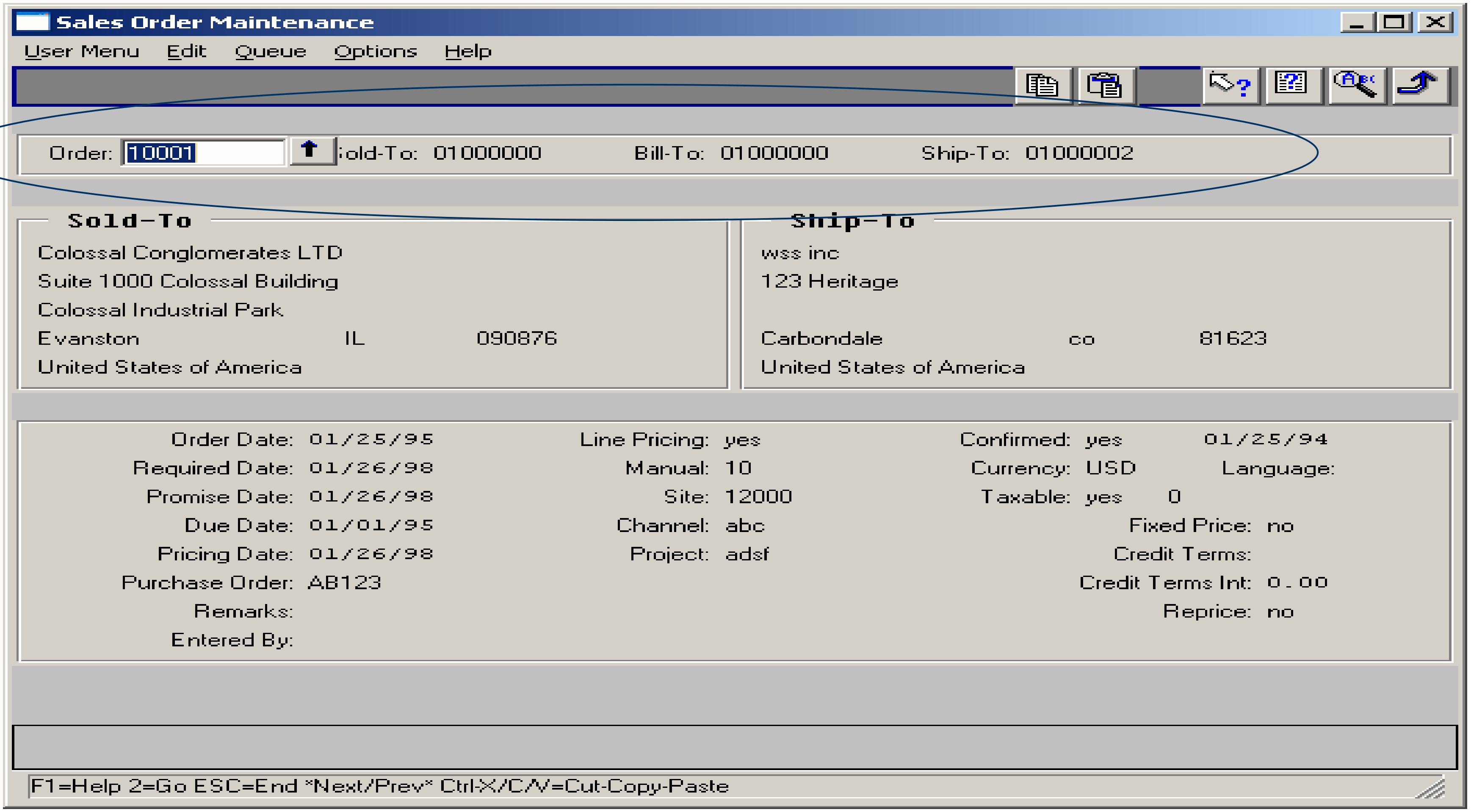
```
put unformatted "run get_" MaintScreen.TableName ".p"
   "('" MaintScreen.TableName "'," 
   MaintScreen.queryphrase ","
   MaintScreen.OneRecord
   ",input-output table " TempTableName  ")."
   skip(1)
   "find first " TempTableName "." skip(1)
   "display " skip.
 for each maintfield where maintfield.progname =
  MaintScreen.progname
   and MaintScreen.framename = maintfield.framename no-lock:
   put unformatted tempTableName "." FieldName  skip.
 end.
 put unformatted
   "with frame " skip
   MaintScreen.framename "." skip(1).
 output close.
```

Generated Application

Resulting Program

```
define temp-table tso_mstr like so_mstr field tRowid as rowid.
define variable hAppServer as handle.
form
tso_mstr.so_nbr at 7
tso_mstr.so_cust at 23.88
tso_mstr.so_bill at 39.75
tso_mstr.so_ship at 56.63
with
 row 3    column 1   size 80 by 2   no-box
 side-label  three-d frame a.

view frame a.

run get_so_mstr.p on hAppServer
  ('so_mstr',true,yes,input-output table tso_mstr).

find first tso_mstr.

display
tso_mstr.so_nbr
tso_mstr.so_cust
tso_mstr.so_bill
tso_mstr.so_ship
with frame a.
```

Original Screen



Sales Order Maintenance

User Menu Edit Queue Options Help

Order: 10001 Sold-To: 01000000 Bill-To: 01000000 Ship-To: 01000002

Sold-To

Colossal Conglomerates LTD
Suite 1000 Colossal Building
Colossal Industrial Park
Evanston IL 090876
United States of America

Ship-To

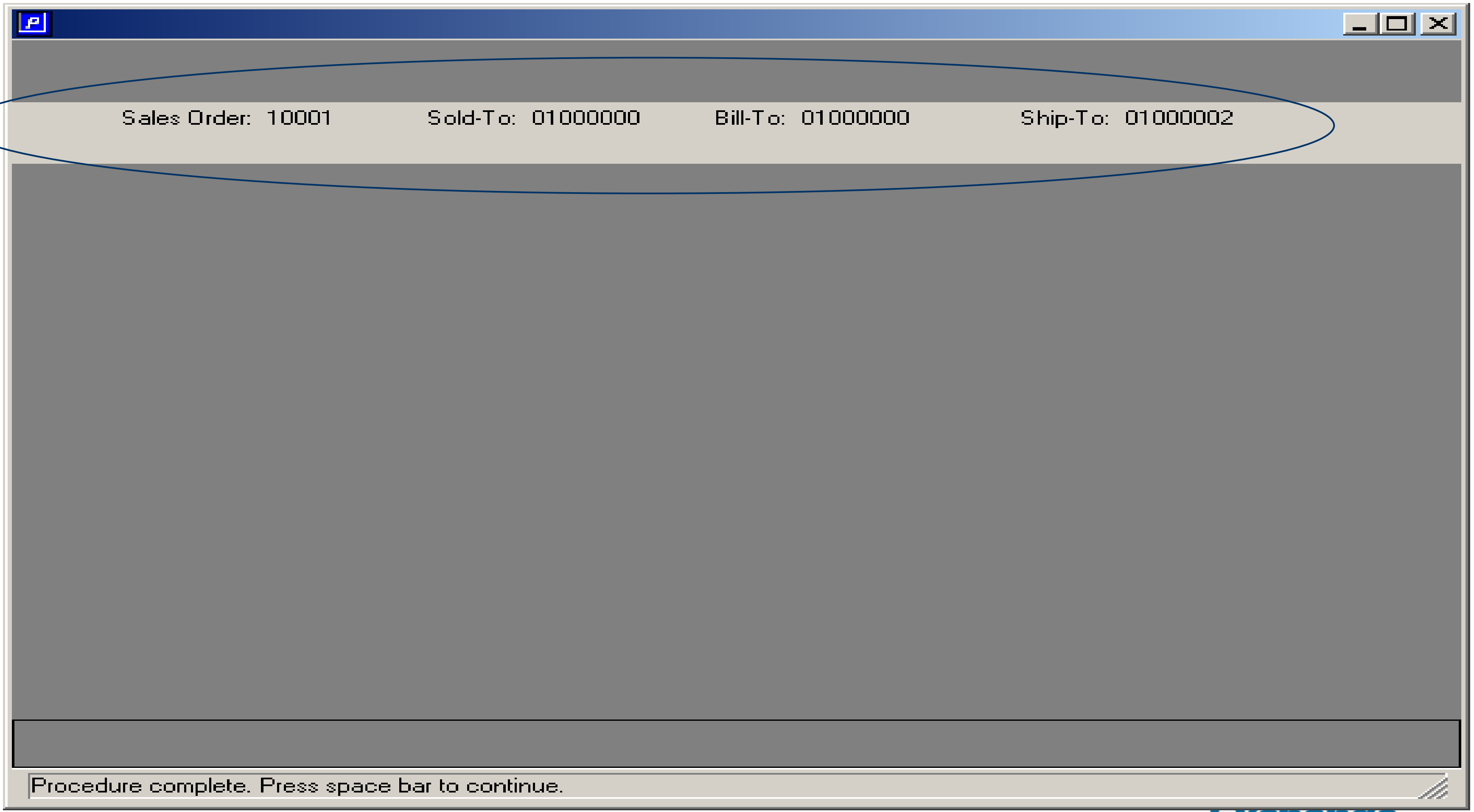
wss inc
123 Heritage

Carbondale co 81623
United States of America

Order Date:	01/25/95	Line Pricing:	yes	Confirmed:	yes 01/25/94
Required Date:	01/26/98	Manual:	10	Currency:	USD Language:
Promise Date:	01/26/98	Site:	12000	Taxable:	yes 0
Due Date:	01/01/95	Channel:	abc	Fixed Price:	no
Pricing Date:	01/26/98	Project:	adsf	Credit Terms:	
Purchase Order:	AB123			Credit Terms Int:	0.00
Remarks:				Reprice:	no
Entered By:					

F1=Help 2=Go ESC=End *Next/Prev* Ctrl-X/C/V=Cut-Copy-Paste

Resulting Frame



```
Sales Order:  10001          Sold-To:  01000000          Bill-To:  01000000          Ship-To:  01000002
```

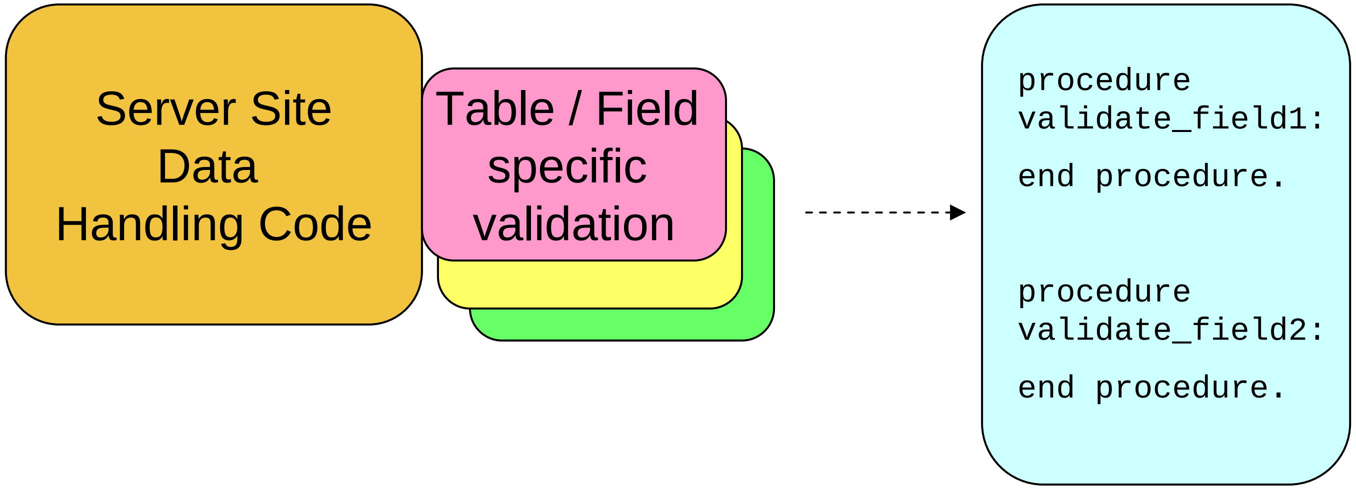
```
Procedure complete. Press space bar to continue.
```

Application Considerations

New Code Logic

- *Re-usable libraries*
- *Consistently used code*
- *Put common code in one module*

Server-side validation



```
procedure
validate_field1:
end procedure.


procedure
validate_field2:
end procedure.
```

Multiple persistent procedures

- *Memory vs. speed*
- *Progress is pretty efficient*
- *Memory is cheaper*

Re-usable queries

- *Queries by table*
- *One program or many*
- *Dynamic queries*
- *Flexible*
- *Hard to read / maintain*

Keep what works!

- *Don't redo every line of code*
- *Much functionality is robust*
- *Redo what needs help*
- *Many parts of existing code were slapped together*
- *Others are added to, and added to…*

The Big Picture

- *If your application works, it's "right"*
- *Complete rewrites are extremely difficult*
  - *Many companies have failed to accomplish them*
- *You have one chance at this, so do it right!*

The Big Picture

- *Many parts of your existing code were slapped together*
  - *(prototype becomes production)*
- *Others were added to, and added to…*
- *Streamline bad code when feasible*
- *Keep old code when reasonable*

Move simple validation toward the client

- *Load static data to temp-tables*
  - *At startup*
  - *Only when needed*
- *Consider local flat files*
- *Keep database accesses distinct from logic*
- *Consider using distinct modules to populate temp tables*
- *Benchmark efficiency*

Code Architecture

- *Re-structure but don't over-structure*
- *Use the right technology for the environment*
- *Super-procedures*
- *Persistent procedures*

Don't over-engineer

- *Simpler is always better*
- *Be sparing of*
  - *Publish-subscribe*
  - *Dynamic objects*
  - *Examples of why*

What's Possible

- *Single-platform*
- *Host-based TTY*
- *GUI Client-server*
- *Not much different from web based*
- *App-server based*
- *Web Client gives rich interface*
- *.NET possible, but beware*

Generating Code

- *A repository is key to*
  - *Consistency*
  - *Changeability*

Summary

- *Application Migration is*
  - *Everybody’s Goal*
  - *Not Simple*
  - *Not Impossible*

Summary

- *Look at Models*
  - *DWP*
  - *OpenEdge® Reference Architecture*
  - *Other Vendors / developers*

Questions?

Thanks

John Campbell

White Star Software