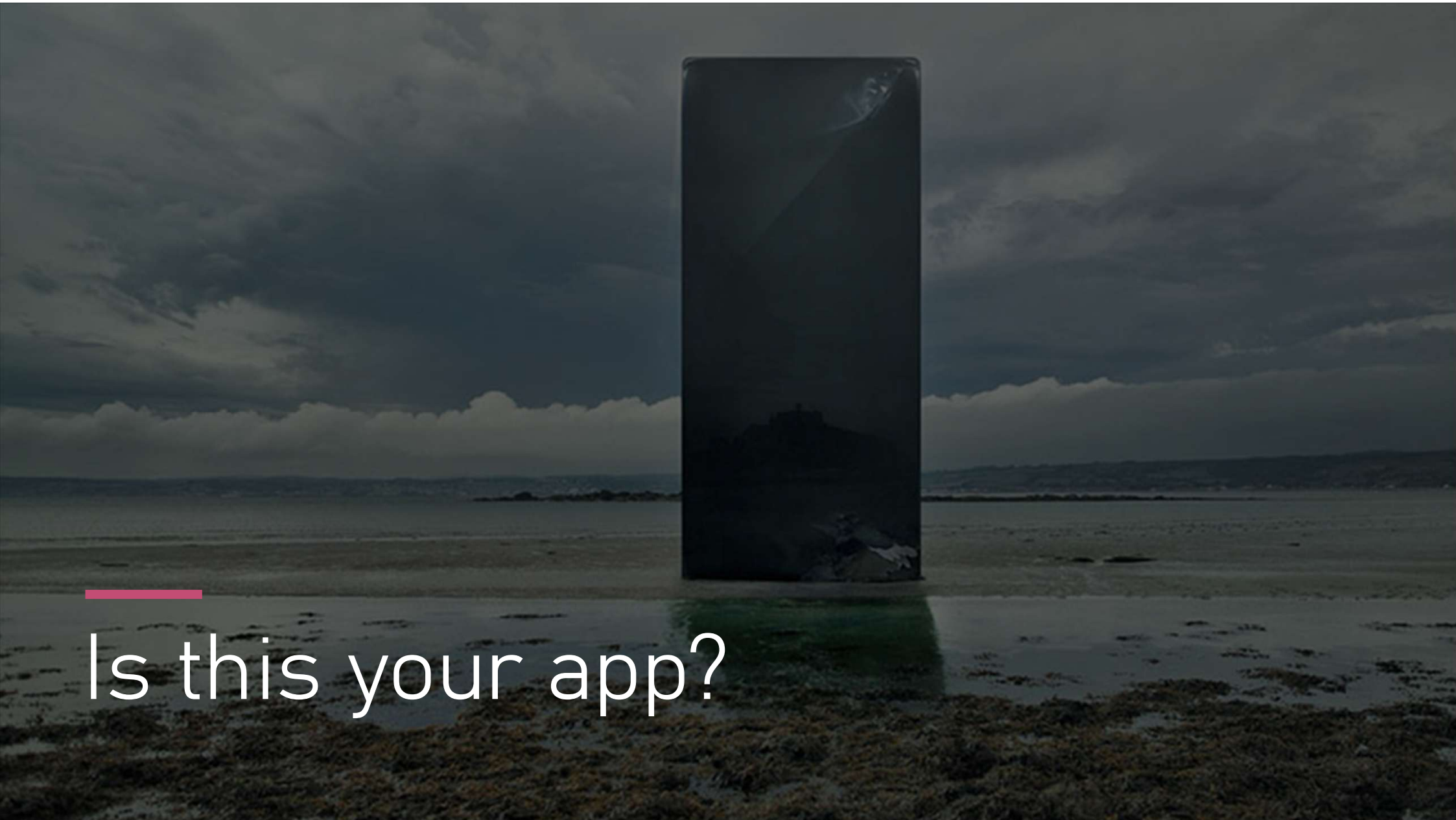


Leveraging modern programming methods in your legacy application

JAMES PALMER – VERTU MOTORS

About James

- Working with OpenEdge since 2003
- Currently Development Manager at Vertu – Automotive Retailer in the UK
- Keen interest in modernization and removal of technical debt
- Also likes staring into space and blowing his own trumpet



Is this your app?

Monoliths are common

- They're nothing to be ashamed about, really. Symptom of the way life was.
- Modernizing is hard
 - Technical Debt
 - No time
 - Complex business rules
 - Business logic mixed with UI, etc, etc.
- But – all is not lost. (Some) salvation is at hand

About this talk

- This talk is not...
 - An excuse to ignore your existing code
 - A talk about modernizing your app
 - A talk about OO programming
- This talk is...
 - A guide to help you use more modern approaches
 - An opportunity to learn
 - An overview of some of the newer development options in OpenEdge
- All examples written in OE 11.7.21 (Sorry! We're still stuck there!)



Get on with it!

Classes

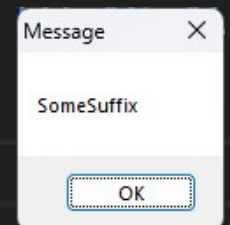
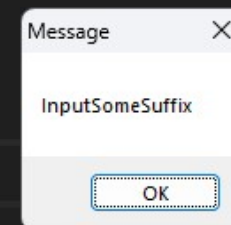
- Anyone can use classes. Not restricted to modern apps
- Classes contain methods that are very similar to procedures and functions
- Also have properties
- Usually must be instantiated or “NEWed”. Instance is held in a variable and must be disposed of
- But – much easier to handle than persistent procedures etc.

Advantages over procedural

- Compile time parameter checking
- Have an optional constructor and destructor method
- Overloading ← Super useful!
- Methods can be public, private, protected ← control who can do what
- Code completion (Dev Studio, VS Code, etc)
- With Inheritance can be used to make flexible and easily customisable/extendable code
- Much more

adhocs > ≡ CallClass.p

```
1  USING adhocs.* FROM PROPATH.  
2  
3  BLOCK-LEVEL ON ERROR UNDO, THROW.  
4  
5  DEFINE VARIABLE oMyExampleClass AS MyExampleClass NO-UNDO.  
6  
7  oMyExampleClass = NEW MyExampleClass ().  
8  
9  MESSAGE oMyExampleClass:ConcatOperation("Input") VIEW-AS ALERT-BOX.  
10 MESSAGE oMyExampleClass:ConcatOperation() VIEW-AS ALERT-BOX.  
11  
12 FINALLY:  
13     IF VALID-OBJECT (oMyExampleClass) THEN  
14         DELETE OBJECT oMyExampleClass.  
15 END FINALLY.
```



Static Classes

- Very similar to standard classes
- All elements are "STATIC"
- No need to instantiate
- But... the version instantiated will persist until the end of the session ← CAUTION!
- Very useful for repetitive tasks
- For example – cleaning up handles/objects

adhocs > TrashHelper.cls

```
1
2 BLOCK-LEVEL ON ERROR UNDO, THROW.
3
4 CLASS adhoc.TrashHelper:
5
6     CONSTRUCTOR PROTECTED TrashHelper (): //Stop accidental instantiation
7         SUPER ().
8
9     END CONSTRUCTOR.
10
11     METHOD PUBLIC STATIC VOID DeleteObject (INPUT oObject AS Progress.Lang.Object):
12         IF VALID-OBJECT (oObject) THEN
13             DELETE OBJECT oObject.
14     END METHOD.
15
16     METHOD PUBLIC STATIC VOID DeleteObject (INPUT oHandle AS HANDLE):
17         IF VALID-HANDLE (oHandle) THEN
18             DELETE OBJECT oHandle.
19     END METHOD.
20 END CLASS.
```

```
adhocs > ≡ Trashrunner.p
```

```
1  USING adhoc.* FROM PROPATH.  
2  
3  BLOCK-LEVEL ON ERROR UNDO, THROW.  
4  
5  DEFINE VARIABLE oObject AS MyExampleClass NO-UNDO.  
6  
7  oObject = NEW MyExampleClass ().  
8  
9  //Some stuff  
10  
11  FINALLY:  
12  |    TrashHelper>DeleteObject (oObject).  
13  END FINALLY.
```

ENUMs

- OO Datatype
- Designed to be used as a set of named values
- Can be used as parameters to methods
- Enforces the possible values that can be passed
- Examples include days of week, months of year or constants for external systems
- Best for lists that are (reasonably) static

```
adhocs > DaysOfWeek.cls
```

```
1  ENUM adhocs.DaysOfWeek :
```

```
2
```

```
3      DEFINE ENUM Sunday
```

```
4          Monday
```

```
5          Tuesday
```

```
6          Wednesday
```

```
7          Thursday
```

```
8          Friday
```

```
9          Saturday
```

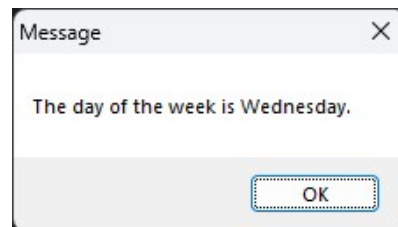
```
10         .
```

```
11
```

```
12     END ENUM.
```

```
adhocs > ☰ RunDates.p
```

```
1 USING adhoc.* FROM PROPATH.  
2  
3 BLOCK-LEVEL ON ERROR UNDO, THROW.  
4  
5 DEFINE VARIABLE oDayOfWeek AS DaysOfWeek NO-UNDO.  
6 DEFINE VARIABLE iWeekday AS INTEGER NO-UNDO.  
7  
8 iWeekday = WEEKDAY (TODAY).  
9 oDayOfWeek = DaysOfWeek:GetEnum (iWeekday).  
10  
11 MESSAGE SUBSTITUTE ("The day of the week is &1.",  
12      oDayOfWeek:ToString()) VIEW-AS ALERT-BOX.
```

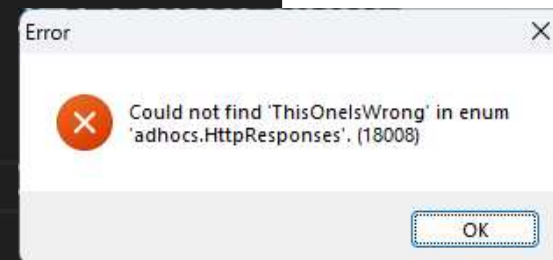
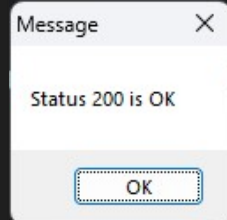


```
adhocs > HttpResponses.cls
1  ENUM adhoc.HttpResponses :
2
3      DEFINE ENUM OK           = 200
4          Created              = 201
5          Accepted              = 202
6          Unauthorized          = 401
7          Forbidden             = 403
8          .
9
10 END ENUM.
```

- Obviously, you wouldn't actually do it this way! This just shows a principle.

adhocs > ☰ LookupHttpStatus.p

```
1 USING adhocs.* FROM PROPATH.  
2  
3 BLOCK-LEVEL ON ERROR UNDO, THROW.  
4  
5 DEFINE VARIABLE oHttpResponses AS HttpResponses NO-UNDO.  
6  
7 oHttpResponses = HttpResponses:GetEnum (200).  
8  
9 MESSAGE "Status 200 is" oHttpResponses:ToString() VIEW-AS ALERT-BOX.  
10  
11 oHttpResponses = HttpResponses:GetEnum ("Unauthorized").  
12  
13 MESSAGE "Status Unauthorized is" oHttpResponses:GetValue() VIEW-AS ALERT-BOX.  
14  
15 oHttpResponses = HttpResponses:GetEnum ("ThisOneIsWrong").  
16  
17 MESSAGE "Status ThisOneIsWrong is" oHttpResponses:GetValue() VIEW-AS ALERT-BOX.  
18  
19 CATCH e AS Progress.Lang.Error:  
20     MESSAGE e:GetMessage(1) VIEW-AS ALERT-BOX ERROR.  
21 END CATCH.
```

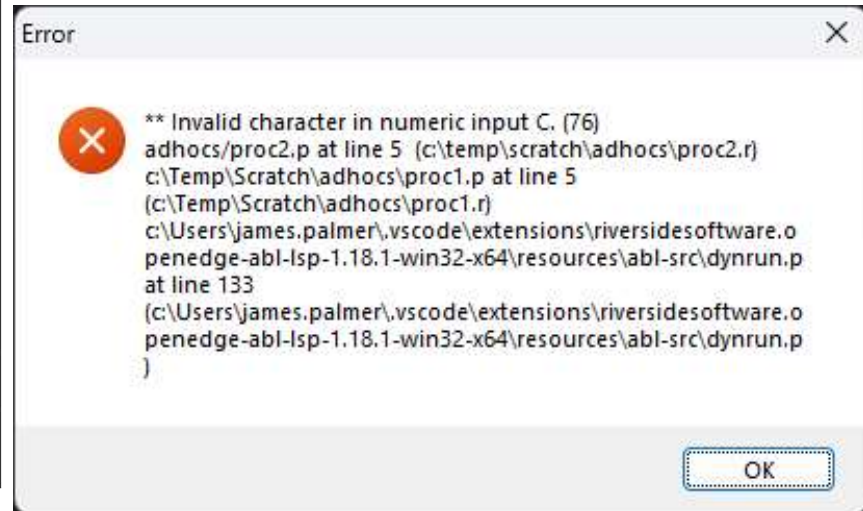


Structured Error Handling

- Been around for a while
- Much better way of handling/trapping errors
- You decide when errors appear and what they look like
- Works with classes and procedural code
- Works on any error that's not suppressed by NO-ERROR
- Not going to go into detail, but you should consider it over ERROR-STATUS:ERROR etc
- Can also define your own error types to handle different errors in different ways

```
adhocs > ≡ proc2.p
```

```
1 BLOCK-LEVEL ON ERROR UNDO, THROW.  
2  
3 DEFINE VARIABLE iNumber AS INTEGER NO-UNDO.  
4  
5 ASSIGN  
6 | iNumber = INT ("C123"). |
```



```
adhocs > ≡ proc1.p
```

```
1 BLOCK-LEVEL ON ERROR UNDO, THROW.  
2  
3 ASSIGN SESSION:ERROR-STACK-TRACE = TRUE.  
4  
5 RUN adhocs/proc2.p.  
6  
7 CATCH e AS Progress.Lang.Error:  
8 | MESSAGE e:GetMessage (1) SKIP e:CallStack VIEW-AS ALERT-BOX ERROR.  
9 END CATCH. |
```

JSON Object Model

- Have you seen or written complex code to make or parse JSON files?
- What happens when the payload changes?
- Clunky, difficult to maintain
- The JSON Object Model is Progress's answer
- Class based
- Included with your license
- Works for creating and parsing JSON files

```
adhocs > CreateJson.p
1 USING adhoc.* FROM PROPATH.
2 USING Progress.Json.ObjectModel.* FROM PROPATH.
3
4 BLOCK-LEVEL ON ERROR UNDO, THROW.
5
6 DEFINE VARIABLE cOutFile AS CHARACTER NO-UNDO.
7 DEFINE VARIABLE oJsonObject AS JsonObject NO-UNDO.
8 DEFINE VARIABLE oJSONArray AS JSONArray NO-UNDO.
9 DEFINE VARIABLE oAttendeeArray AS JSONArray NO-UNDO.
10 DEFINE VARIABLE oAttendeeObject AS JsonObject NO-UNDO.
11
12 cOutFile = "c:\temp\myjson.json".
13 oJsonObject = NEW JsonObject ().
14 oJSONArray = NEW JSONArray ().
15 oAttendeeArray = NEW JSONArray ().
16
17 //Main Details
18 oJsonObject:Add ("Event", "Progress User Group").
19 oJsonObject:Add ("Venue", "City of Manchester Stadium").
20 oJsonObject:Add ("Date", 03/20/2025).
21 (Alt+C) Duo Quick Chat
22 //Colleagues
23 oAttendeeObject = NEW JsonObject ().
24 oAttendeeObject:Add ("Name", "James Palmer").
25 oAttendeeArray:Add (oAttendeeObject).
26 oAttendeeObject = NEW JsonObject ().
27 oAttendeeObject:Add ("Name", "John Ainsworth").
28 oAttendeeArray:Add (oAttendeeObject).
29
30 oJsonObject:Add ("Attendees",oAttendeeArray).
31 oJSONArray:Add (oJsonObject).
32
33 oJSONArray:WriteFile (cOutFile).
34
35 CATCH e AS Progress.Lang.Error:
36 MESSAGE e:GetMessage(1) VIEW-AS ALERT-BOX ERROR.
37 END CATCH.
38
39 FINALLY:
40 TrashHelper>DeleteObject (oJsonObject).
41 TrashHelper>DeleteObject (oJSONArray).
42 TrashHelper>DeleteObject (oAttendeeArray).
43 TrashHelper>DeleteObject (oAttendeeObject).
44 END FINALLY.
```

```
//Main Details
oJsonObject:Add ("Event", "Progress User Group").
oJsonObject:Add ("Venue", "City of Manchester Stadium").
oJsonObject:Add ("Date", 03/20/2025).

//Colleagues
oAttendeeObject = NEW JsonObject ().
oAttendeeObject:Add ("Name", "James Palmer").
oAttendeeArray:Add (oAttendeeObject).
oAttendeeObject = NEW JsonObject ().
oAttendeeObject:Add ("Name", "John Ainsworth").
oAttendeeArray:Add (oAttendeeObject).

oJsonObject:Add ("Attendees",oAttendeeArray).
oJsonArray:Add (oJsonObject).

oJsonArray:WriteFile (cOutFile).
```

C: > Temp > {} myjson.json > ...

```
1  [
2  {
3      "Event": "Progress User Group",
4      "Venue": "City of Manchester Stadium",
5      "Date": "2025-03-20",
6      "Attendees": [
7          {
8              "Name": "James Palmer"
9          },
10         {
11             "Name": "John Ainsworth"
12         }
13     ]
14 }
15 ]
```

adhocs > ReadJson.p

```
1  USING adhoc.* FROM PROPATH.
2  USING Progress.Json.ObjectModel.* FROM PROPATH.
3
4  BLOCK-LEVEL ON ERROR UNDO, THROW.
5
6  DEFINE VARIABLE cJsonOutput AS LONGCHAR NO-UNDO.
7  DEFINE VARIABLE cInFile AS CHARACTER NO-UNDO.
8  DEFINE VARIABLE oParser AS ObjectModelParser NO-UNDO.
9  DEFINE VARIABLE oJsonObject AS JsonObject NO-UNDO.
10 DEFINE VARIABLE oJsonArray AS JsonArray NO-UNDO.
11 DEFINE VARIABLE oAttendeeArray AS JsonArray NO-UNDO.
12 DEFINE VARIABLE oAttendeeObject AS JsonObject NO-UNDO.
13 DEFINE VARIABLE i AS INTEGER NO-UNDO.
14
15 cInFile = "c:\temp\myjson.json".
16
17 FIX-CODEPAGE (cJsonOutput) = 'utf-8'.
18
19 COPY-LOB FROM FILE cInFile TO cJsonOutput CONVERT TARGET CODEPAGE "UTF-8".
20
21 oParser = NEW ObjectModelParser().
22 oJsonArray = CAST (oParser:Parse(cJsonOutput), JsonArray).
23 oJsonObject = oJsonArray:GetJsonObject (1).
24
25 MESSAGE "Event:" oJsonObject:GetCharacter ("Event") SKIP "Venue:" oJsonObject:GetCharacter ("Venue") SKIP "Date:" oJsonObject:GetDate ("Date") VIEW-AS ALERT-BOX INFO.
26
27 oAttendeeArray = oJsonObject:GetJsonArray("Attendees").
28
29 DO i = 1 TO oAttendeeArray:Length:
30     oAttendeeObject = oAttendeeArray:GetJsonObject (i).
31     MESSAGE "Attendee" i "-" oAttendeeObject:GetCharacter("Name") VIEW-AS ALERT-BOX INFO.
32 END.
33
34 CATCH e AS Progress.Lang.Error:
35     MESSAGE e:GetMessage(1) VIEW-AS ALERT-BOX ERROR.
36 END CATCH.
37
38 FINALLY:
39     TrashHelper>DeleteObject (oJsonObject).
40     TrashHelper>DeleteObject (oJsonArray).
41     TrashHelper>DeleteObject (oParser).
42     TrashHelper>DeleteObject (oAttendeeArray).
43     TrashHelper>DeleteObject (oAttendeeObject).
44 END FINALLY. |
```

```

FIX-CODEPAGE (cJsonOutput) = 'utf-8'.

COPY-LOB FROM FILE cInFile TO cJsonOutput CONVERT TARGET CODEPAGE "UTF-8".

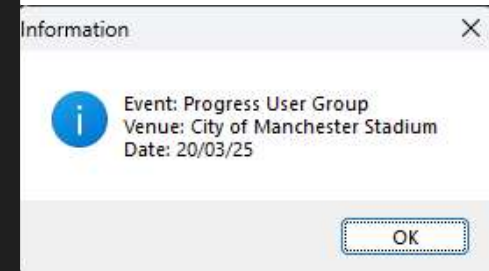
oParser      = NEW ObjectModelParser().
oJSONArray   = CAST (oParser:Parse(cJsonOutput), JSONArray).
oJsonObject  = oJSONArray:GetJsonObject (1).

MESSAGE "Event:" oJsonObject:GetCharacter ("Event") SKIP
        "Venue:" oJsonObject:GetCharacter ("Venue") SKIP
        "Date:" oJsonObject:GetDate ("Date")
        VIEW-AS ALERT-BOX INFO.

oAttendeeArray = oJsonObject:GetJsonArray("Attendees").

DO i = 1 TO oAttendeeArray:Length:
    oAttendeeObject = oAttendeeArray:GetJsonObject (i).
    MESSAGE "Attendee" i "-" oAttendeeObject:GetCharacter("Name") VIEW-AS ALERT-BOX INFO.
END.

```



```
adhocs > ≡ CreateJsonDataset.p
1  BLOCK-LEVEL ON ERROR UNDO, THROW.
2
3  DEFINE TEMP-TABLE ttEvent NO-UNDO
4      FIELD eventId AS INTEGER
5      FIELD eventName AS CHARACTER
6      FIELD venue AS CHARACTER
7      FIELD eventDate AS DATE
8      INDEX idxPrim IS PRIMARY UNIQUE eventId.
9
10 DEFINE TEMP-TABLE ttAttendee NO-UNDO
11     FIELD attendeeId AS INTEGER
12     FIELD eventId AS INTEGER
13     FIELD attendeeName AS CHARACTER
14     INDEX idxPrim IS PRIMARY UNIQUE attendeeId
15     INDEX idxEvent eventId.
16
17 DEFINE DATASET dsEventAttendees FOR ttEvent, ttAttendee
18     DATA-RELATION attendee FOR ttEvent, ttAttendee
19     RELATION-FIELDS (eventId, eventId).
20
21 DEFINE VARIABLE hdsAttendee AS HANDLE NO-UNDO.
22 DEFINE VARIABLE cOutFile AS CHARACTER NO-UNDO.
23
24 RUN CreateData.
25
26 ASSIGN
27     hdsAttendee = DATASET dsEventAttendees:HANDLE
28     cOutFile = "c:\temp\myjsonds.json"
29     .
30
31 hdsAttendee:WRITE-JSON("file", cOutFile, TRUE).
32 (Alt+C) Duo Quick Chat
33
34 CATCH e AS Progress.Lang.Error:
35     MESSAGE e:GetMessage(1) VIEW-AS ALERT-BOX ERROR.
36 END CATCH.
```

```
26 ASSIGN
27     hdsAttendee = DATASET dsEventAttendees:HANDLE
28     cOutFile    = "c:\temp\myjsonds.json"
29     .
30
31 hdsAttendee:WRITE-JSON("file", cOutFile, TRUE).
32 (Alt+C) Duo Quick Chat
33
```

```
C: > Temp > {} myjsonds.json > ...
1 {"dsEventAttendees": {
2   "ttEvent": [
3     {
4       "EventId": 1,
5       "EventName": "Progress User Group",
6       "Venue": "City of Manchester Stadium",
7       "EventDate": "2025-03-20"
8     }
9   ],
10  "ttAttendee": [
11    {
12      "AttendeeId": 1,
13      "EventId": 1,
14      "AttendeeName": "James Palmer"
15    },
16    {
17      "AttendeeId": 2,
18      "EventId": 1,
19      "AttendeeName": "John Ainsworth"
20    }
21  ]
22 }}
```

adhocs > ReadJsonDataset.p

```
1 USING adhocs.* FROM PROPATH.
2 BLOCK-LEVEL ON ERROR UNDO, THROW.
3
4 DEFINE VARIABLE hEventAttendee AS HANDLE NO-UNDO.
5 DEFINE VARIABLE cInFile AS CHARACTER NO-UNDO.
6 DEFINE VARIABLE hQuery AS HANDLE NO-UNDO.
7 DEFINE VARIABLE cRecord AS CHARACTER NO-UNDO.
8 DEFINE VARIABLE iBuffer AS INTEGER NO-UNDO.
9 DEFINE VARIABLE hBuffer AS HANDLE NO-UNDO.
10 DEFINE VARIABLE iLoop AS INTEGER NO-UNDO.
11
12 cInFile = "c:\temp\myjsonds.json".
13
14 CREATE DATASET hEventAttendee.
15
16 hEventAttendee:READ-JSON ("file", cInFile, "empty").
17
18 CREATE QUERY hQuery.
19
20 DO iBuffer = 1 TO hEventAttendee:NUM-BUFFERS:
21     hBuffer = hEventAttendee:GET-BUFFER-HANDLE(iBuffer).
22
23     hQuery:SET-BUFFERS(hBuffer).
24
25     hQuery:QUERY-PREPARE("FOR EACH " + hBuffer:NAME).
26     hQuery:QUERY-OPEN().
27
28     DO WHILE hQuery:GET-NEXT ():
29
30         DO iLoop = 1 TO hBuffer:NUM-FIELDS:
31             cRecord = cRecord + STRING(hBuffer:BUFFER-FIELD(iLoop):NAME) + ": " + STRING(hBuffer:BUFFER-FIELD(iLoop):BUFFER-VALUE) + CHR(13).
32         END.
33         MESSAGE "Temp-Table : " hBuffer:NAME SKIP cRecord
34             VIEW-AS ALERT-BOX.
35         cRecord = "".
36     END.
37     cRecord = "".
38 END.
39
40 hQuery:QUERY-CLOSE().
```

```

CREATE DATASET hEventAttendee.

hEventAttendee:READ-JSON ("file", cInFile, "empty").

CREATE QUERY hQuery.

DO iBuffer = 1 TO hEventAttendee:NUM-BUFFERS:
  hBuffer = hEventAttendee:GET-BUFFER-HANDLE(iBuffer).

  hQuery:SET-BUFFERS(hBuffer).

  hQuery:QUERY-PREPARE("FOR EACH " + hBuffer:NAME).
  hQuery:QUERY-OPEN().

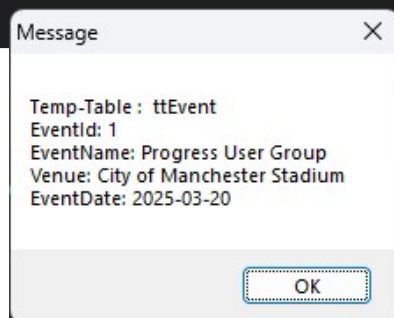
  DO WHILE hQuery:GET-NEXT ():

    DO iLoop = 1 TO hBuffer:NUM-FIELDS:
      cRecord = cRecord + STRING(hBuffer:BUFFER-FIELD(iLoop):NAME) + ": " + STRING(hBuffer:BUFFER-FIELD(iLoop):BUFFER-VALUE) + CHR(13).
    END.
    MESSAGE "Temp-Table : " hBuffer:NAME SKIP cRecord
      VIEW-AS ALERT-BOX.
    cRecord = "".

  END.
  cRecord = "".

END.

```



HTTP Requests

- Do you shell out to cURL or something similar for REST calls?
- Did you know Progress can natively call REST APIs?
- Deals with most situations easily
- Error handling much easier to achieve
- Add `${DLC}/gui/netlib/OpenEdge.net.pl` to the Propath
- Plenty of documentation and examples online
- Note: Can be slow for bigger payloads

```

adhocs > ApiCall.p
1  USING Progress.Json.ObjectModel.* FROM PROPATH.
2  USING adhocs.* FROM PROPATH.
3  USING OpenEdge.Net.HTTP.*      FROM PROPATH.
4
5  BLOCK-LEVEL ON ERROR UNDO, THROW.
6
7  DEFINE VARIABLE cApiUri AS CHARACTER NO-UNDO INITIAL "https://restcountries.com/v3.1/name/".
8  DEFINE VARIABLE cCountry AS CHARACTER NO-UNDO INITIAL "germany".
9
10 DEFINE VARIABLE oRequest AS IHttpRequest NO-UNDO.
11 DEFINE VARIABLE oResponse AS IHttpResponse NO-UNDO.
12 DEFINE VARIABLE oResponseObject AS JsonObject NO-UNDO.
13 DEFINE VARIABLE oResponseArray AS JsonArray NO-UNDO.
14 DEFINE VARIABLE oNameObject AS JsonObject NO-UNDO.
15
16 oRequest = RequestBuilder:GET(cApiUri + cCountry)
17           :REQUEST.
18
19 oResponse = ClientBuilder:Build():Client:Execute(oRequest).
20
21 IF oResponse:StatusCode <> 200 THEN
22     MESSAGE "Code:" oResponse:StatusCode SKIP "Reason:" oResponse:StatusReason VIEW-AS ALERT-BOX ERROR.
23 ELSE
24     DO:
25         ASSIGN
26             oResponseArray = CAST(oResponse:Entity, JsonArray)
27             oResponseObject = oResponseArray:GetJsonObject(1)
28             oNameObject = oResponseObject:GetJsonObject("name")
29             .
30
31         MESSAGE
32             "Name:" oNameObject:GetCharacter("common") SKIP
33             "Official Name:" oNameObject:GetCharacter("official") SKIP
34             "UN Member:" oResponseObject:GetLogical("unMember") SKIP
35             "Independent:" oResponseObject:GetLogical("independent")
36             VIEW-AS ALERT-BOX INFO.
37     END.
38
39 CATCH e AS Progress.Lang.Error:
40     MESSAGE e:GetMessage(1) VIEW-AS ALERT-BOX ERROR.
41 END CATCH.
42
43 FINALLY:
44     TrashHelper:DeleteObject(oRequest).
45     TrashHelper:DeleteObject(oResponse).
46     TrashHelper:DeleteObject(oResponseObject).
47     TrashHelper:DeleteObject(oResponseArray).
48     TrashHelper:DeleteObject(oNameObject).
49 END FINALLY.

```

```
DEFINE VARIABLE cApiUri AS CHARACTER NO-UNDO INITIAL "https://restcountries.com/v3.1/name/".  
DEFINE VARIABLE cCountry AS CHARACTER NO-UNDO INITIAL "germany".
```

```
oRequest = RequestBuilder:GET(cApiUri + cCountry)  
:REQUEST.
```

```
oResponse = ClientBuilder:Build():Client:Execute(oRequest).
```

```
IF oResponse:StatusCode <> 200 THEN
```

```
MESSAGE "Code:" oResponse:StatusCode SKIP "Reason:" oResponse:StatusReason VIEW-AS ALERT-BOX ERROR.
```

```
ELSE
```

```
DO:
```

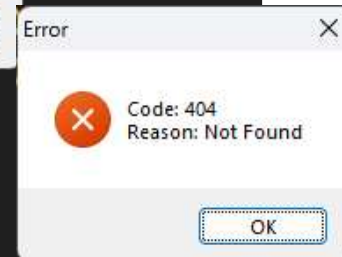
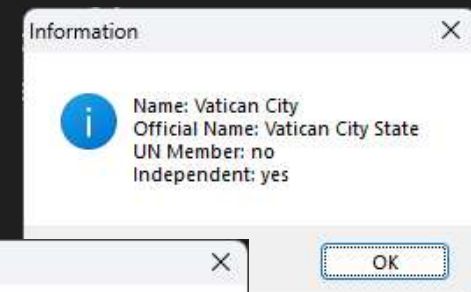
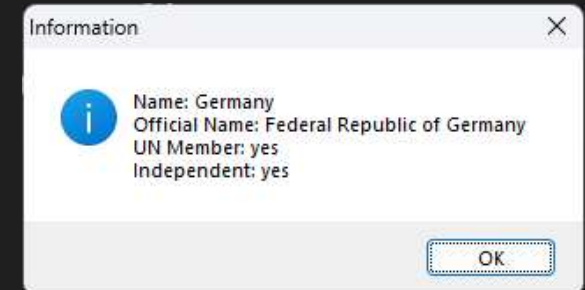
```
ASSIGN
```

```
oResponseArray = CAST(oResponse:Entity, jsonArray)  
oResponseObject = oResponseArray:GetJsonObject(1)  
oNameObject = oResponseObject:GetJsonObject("name")  
.
```

```
MESSAGE
```

```
"Name:" oNameObject:GetCharacter("common") SKIP  
"Official Name:" oNameObject:GetCharacter("official") SKIP  
"UN Member:" oResponseObject:GetLogical("unMember") SKIP  
"Independent:" oResponseObject:GetLogical("independent")  
VIEW-AS ALERT-BOX INFO.
```

```
END.
```



Conclusion

- You don't have to have a modern app to write modern code
- Integrate it into the existing code base
- Do powerful things more simply
- A legacy app is not an excuse to not keep up to date
- This stuff is fun, less error prone and can reduce technical debt introduced



Questions?
