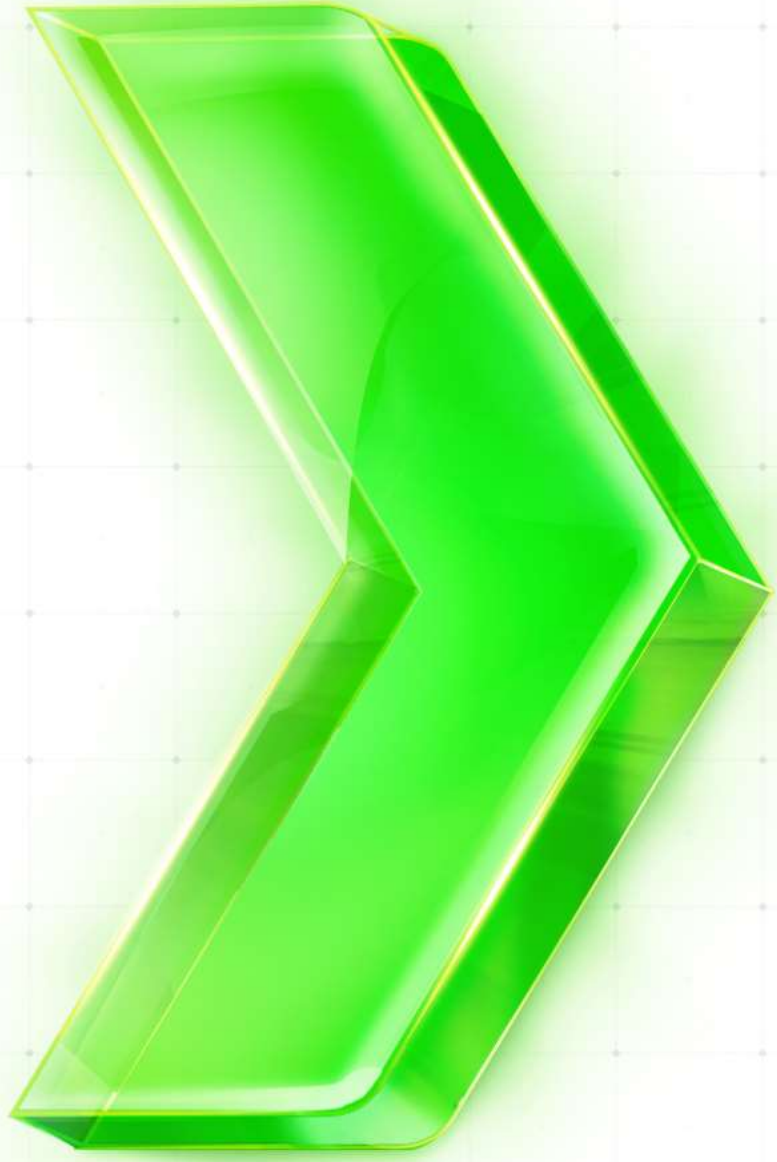
PUGCHALLENGE AMERICAS

Implementing a Custom REST API with ABL WebHandler



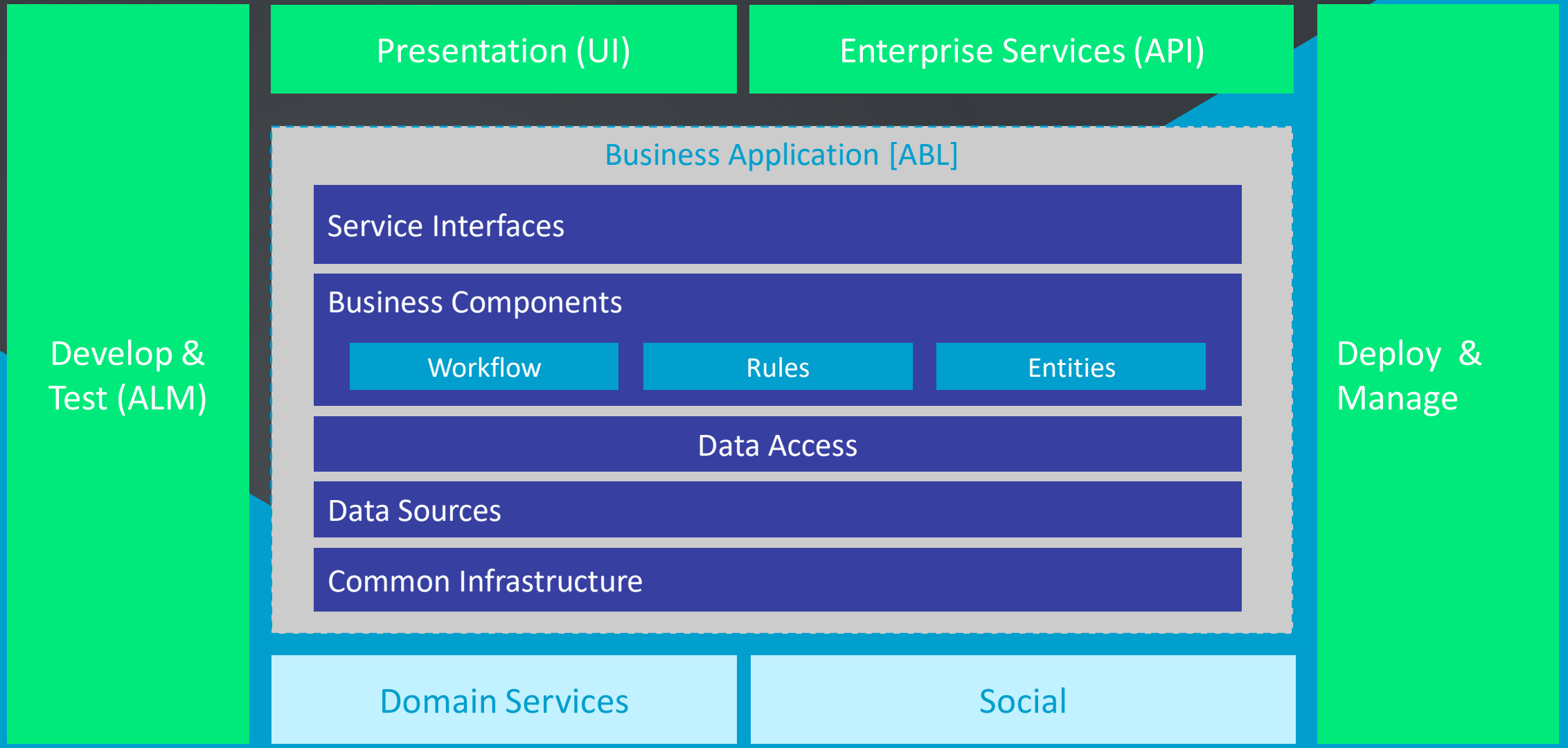
Chris Riddell
Principal Architect, Portfolio+ Inc
David Atkins
Principal Solutions Architect, Progress

Monday 7 October 2019

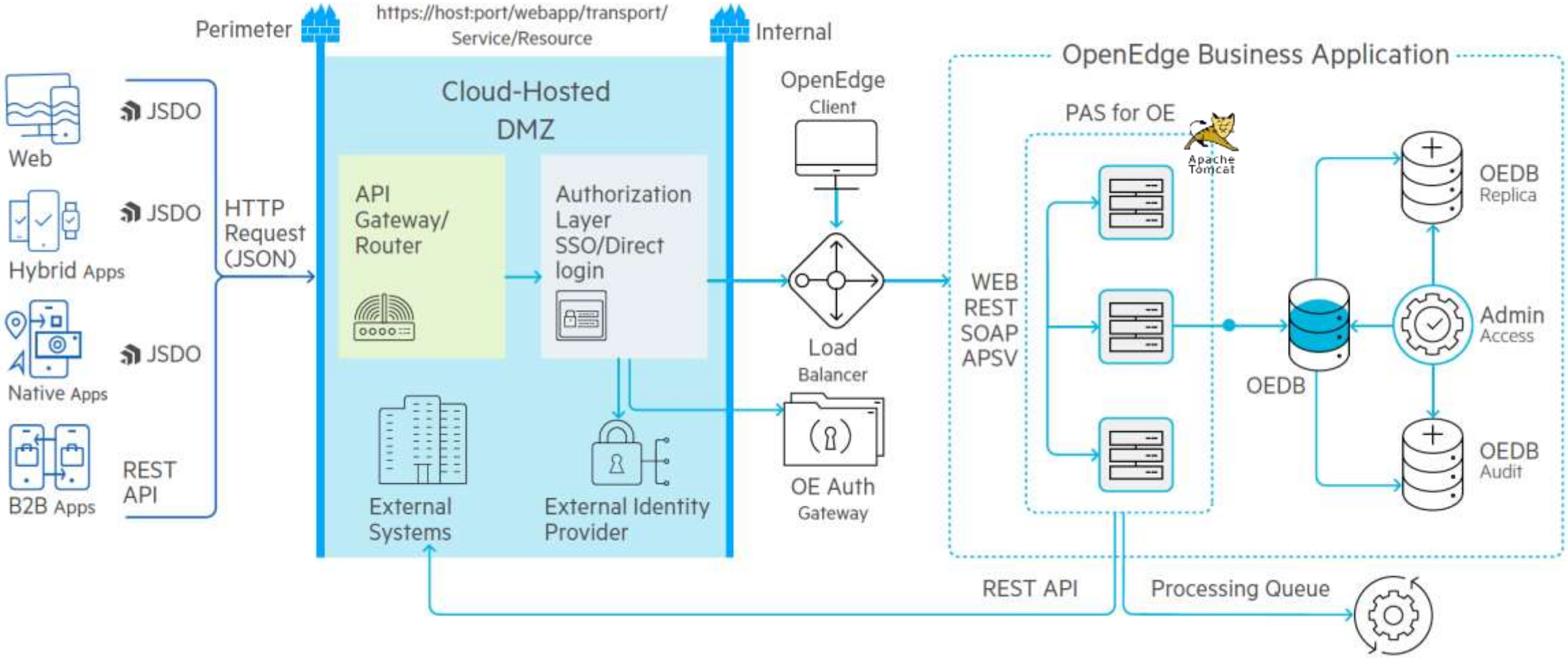
Agenda

- Architecture evolution
- Why Progress AppServer for OpenEdge?
- OpenEdge REST API best practices
- Portfolio+’s Implementation Experiences…
- Wrap up Q & A

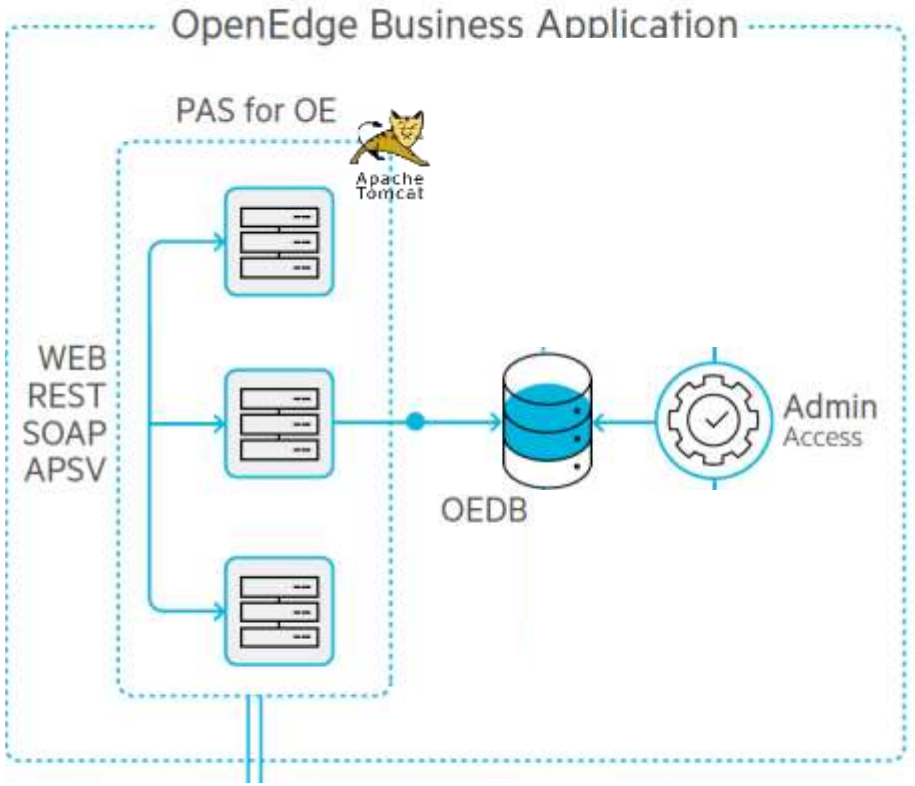
Remember this?



Reference Architecture for Cloud Deployment



Application Architecture



Why PAS for OpenEdge?

Secure: Spring security framework included

Scalability: Uses far less system resources

Simplified: Multi-session multi-mode Agent

Improved administration: Many tool options

Flexibility: REST Service Interface

Deployment flexibility: Supports containerization

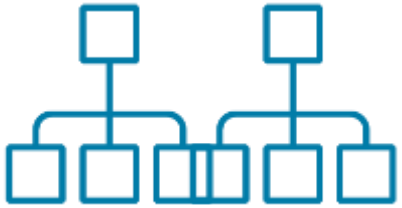
Future: Go forward AppServer for OpenEdge

REST Use Cases

Modern Web Interfaces

Mobile Apps

Emerging Technologies



Application Integration

Modularization through Microservices

Self-Service BI and Analytics

OpenEdge REST Service Interface Options

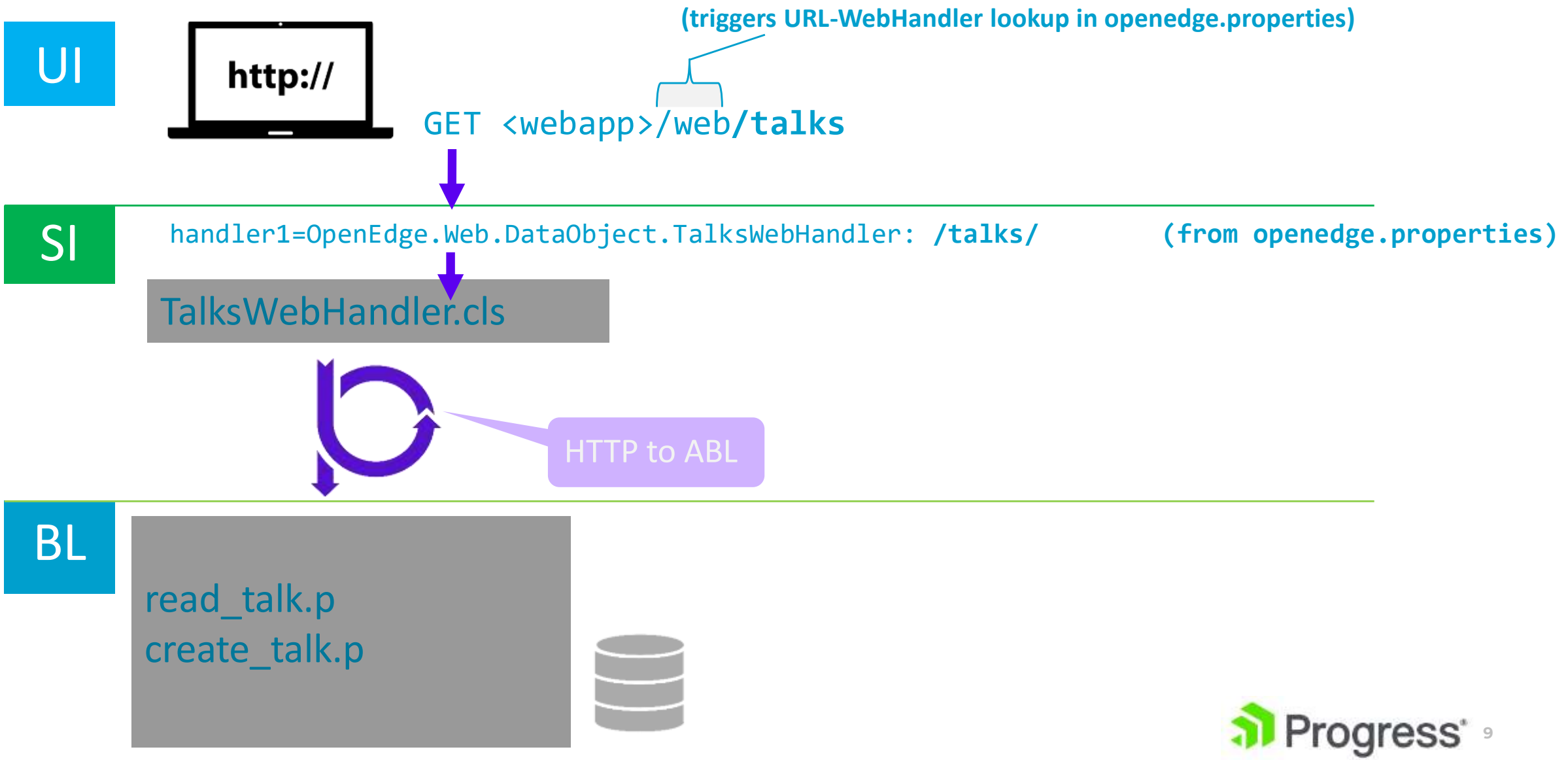
A REST Service Interface:

- Converts REST Request to ABL data
- Routes that ABL data to appropriate ABL business logic
- Converts ABL data to REST Response, & sets HTTP Status Code

Recommended PAS for OpenEdge REST Service Interface options:

- **'No/low-code' approach for prescriptive REST API**
  - **Data Object Services** auto-map standard URL schema to Business Entities classes
  - Enables use of JavaScript Data Object library – 'client-side ProDataSet' with tooling integration
- **ABL-coded approach for custom REST API**
  - **Web Handler** – maximum flexibility and transparency in REST-ABL mapping
- **JSON-configured approach for custom REST API**
  - **DataObjectHandler** – flexible no-code REST-ABL mapping

DIY WebHandler Interaction



```
38    method override protected integer HandleGet( input poRequest as OpenEdge.Web.IWebRequest ):
39        define variable talkId as character no-undo.
40        define variable resp  as WebResponse no-undo.
41        define variable msgBody as JsonObject no-undo.
42        define variable record as JsonObject no-undo.
43        define variable filter as character no-undo.
44        define variable skipRecs as integer no-undo initial 0.
45        define variable topRecs as integer no-undo initial 0.
46        define variable qryCnt as integer no-undo initial 0.
47        define variable hBusinessLogic as handle no-undo.
48
49        assign resp              = new OpenEdge.Web.WebResponse()
50               msgBody           = new JsonObject()
51               resp:ContentType = 'application/json'
52               resp:Entity       = msgBody
53               .
54        //web/talks/{talk-id} | GET | n/a | n/a |  logic/talk/read_talks.p:get_single_talk
55        if right-trim(poRequest:UriTemplate, '/':u) eq '/talks/~{talk-id}' then
56        do:
57            assign talkId = poRequest:GetPathParameter('talk-id')
58                   record = new JsonObject()
59                   .
60            run logic/talk/read_talks.p single-run set hBusinessLogic.
61            run get_single_talk in hBusinessLogic (talkId, output table ttTalk).
62
63            buffer ttTalk:write-json('JsonObject', record, true).
64
65            msgBody:Add('data',  record).
66            msgBody:Add('count', 1).
67        end.
```

Making sense of HTTP requests: content

```
POST /Twe/api/rest/calcTax/doc HTTP/1.1
User-Agent: OpenEdge-HttpClient/0.3.0 (W
Host: sstwsuat.taxware.net
Date: 2016-03-04T12:43:18.547-05:00
Content-Type: application/json
Content-Length: 1805
Authorization: TAX restuat@IGS:NXDFGebsg
Accept: application/json; charset=utf-8

{ "rsltLvl": "1",
  "isAudit": false,
  "currn": "USD",
  "txCalcTp": 1,
  "trnDocNum": "277-0",
  "docDt": "2016-02-29T11:58:00",
  "lines": [
    { "debCredIndr": 1,
      "grossAmt": 858.8,
      "custAttrbs": {
        "COMPANY": "578",
        "CUSTOMER-NUMBER": "101",
        "DIVISION": "1",
        "PRODUCT": "1-101",
        "PRODUCT-CATEGORY": "4",
        "WAREHOUSE": "main"
} } ] }
```

```
method override protected int HandlePost(
                poRequest as IWebRequest):

def var oData as JsonObject.

message poRequest:ContentType
        /* application/json */
        poRequest:ContentLength
        /* 1805 */
```

Making sense of HTTP requests: content

```
POST /Twe/api/rest/calcTax/doc HTTP/1.1
User-Agent: OpenEdge-HttpClient/0.3.0 (W
Host: sstwsuat.taxware.net
Date: 2016-03-04T12:43:18.547-05:00
Content-Type: application/json
Content-Length: 1805
Authorization: TAX restuat@IGS:NXDFGebsg
Accept: application/json; charset=utf-8

{ "rsltLvl": "1",
  "isAudit": false,
  "currn": "USD",
  "txCalcTp": 1,
  "trnDocNum": "277-0",
  "docDt": "2016-02-29T11:58:00",
  "lines": [
    { "debCredIndr": 1,
      "grossAmt": 858.8,
      "custAttrbs": {
        "COMPANY": "578",
        "CUSTOMER-NUMBER": "101",
        "DIVISION": "1",
        "PRODUCT": "1-101",
        "PRODUCT-CATEGORY": "4",
        "WAREHOUSE": "main"
} } ] }
```

```
method override protected int HandlePost(
                  poRequest as IWebRequest):
def var oData as JsonObject.
def var oWriter as MessageWriter no-undo.

oWriter = EntityWriterBuilder:Build(poRequest)
                              :Writer.
oWriter:Open().
oWriter:Write(poRequest:Entity).
oWriter:Close().

if type-of(oWriter:Entity, JsonObject) then
  oData = cast(oWriter:Entity, JsonObject).

message oData:GetLogical('isAudit')
        /* false */
```

Incoming data – OpenEdge.Web.IWebRequest

Message element	oRequest = new WebRequest()	
HTTP method ("verb")	:Method	"POST"
URL	:URI	http://localhost:8810/SportsSvc/web/Customer/catalog?filter={"ablWhere"...}
Query parameters	:URI:GetQueryNames() :URI:GetQueryValue	["filter"] Filter => " {'ablWhere':'custnum eq 42"} "
Headers	:GetHeaders() :GetHeader(<name>):Value	[HttpHeader, HttpHeader, HttpHeader] Accept => "application/json"
Cookies	:GetCookie(<name>)	
Path parameters	* URITemplate * PathParameterNames * GetPathParameter(<name>)	{resources}/catalog/{service} "resources,service" "Customer"
Entity / message body	ContentType / ContentLength Entity	application/json
Path information	* TransportPath * PathInfo * WebAppPath	/web /Customer/catalog /SportsSvc

http://pugchallenge.org/downloads2017/284_web_handlers_deep_dive.pdf

Chris Riddell

Chris.Riddell@portfolioplus.com

Software Architect

Developer and software architect with 10+ years of experience in OpenEdge.
Principal architect for +Open Banking

Portfolio+ Inc.

- Banking and financial services software products
- Publicly held company – Volaris Group under Constellation Software
- Head Quarters in Canada – Stouffville, ON
- Office in Dublin
- Ireland operations since 2007
- 6 of the 7 largest financial institutions in Canada

Agenda

Implementing Custom REST API with ABL WebHandler

1. Background
2. Architecture
3. Development
4. Documentation
5. CI/CD
6. Project Conclusions and Future

Background

portfolio

Where we Started

- Portfolio Plus is a monolithic, fat client application
  - Built using dated ABL GUI components with tight database integration
- Feature rich, Digital Core Banking as a Platform Solution
- Mission critical application for clients
  - 30 years of complex business functions

Primary Goals

- Create Open Banking capability
  - Build an industry standard, RESTful API
- Enable UI/UX modernization to meet customer expectations
- Agile Development techniques
  - Enable automated testing and deployment (CI/CD)
  - Speed to market
- Thin client model to improve performance
  - One HTTP request to retrieve data
  - Compare with many DB requests in the fat client model
  - Improved network usage

Future Strategy

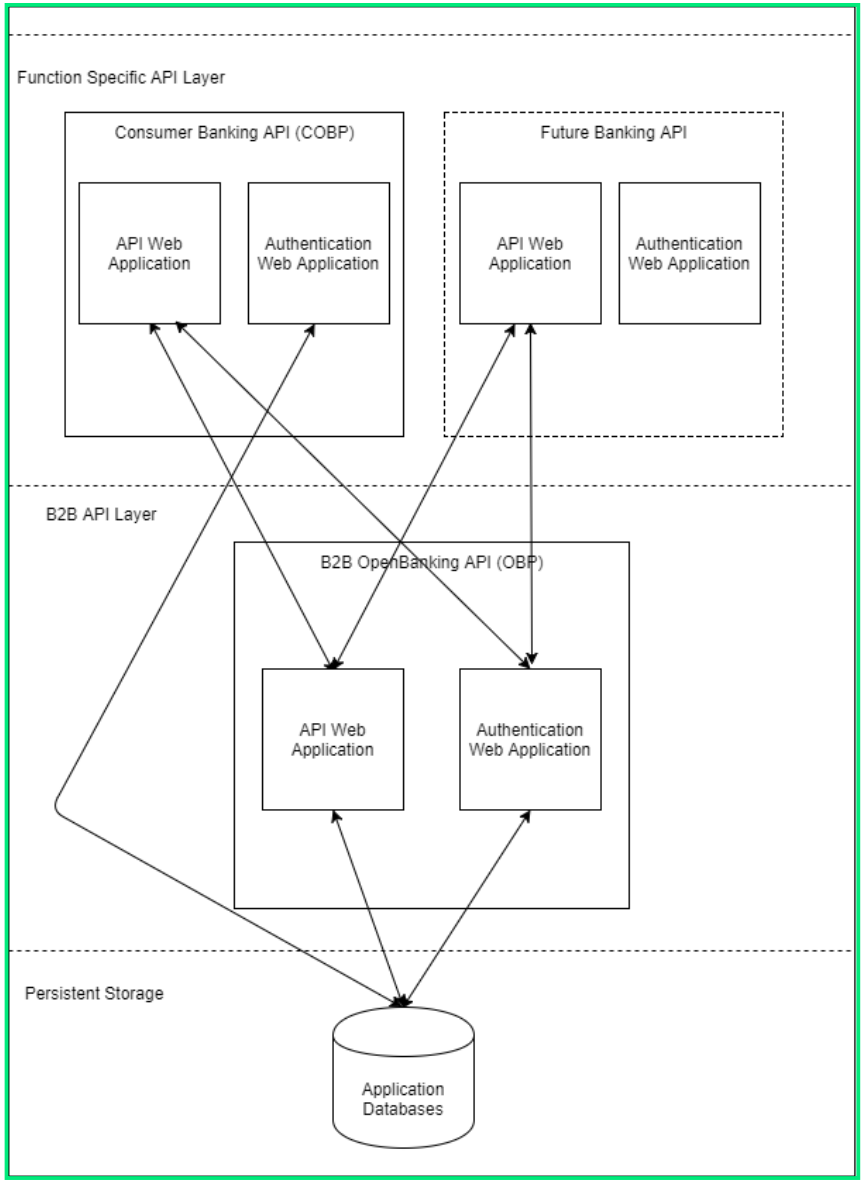
- Rapidly create new applications
- Empower clients
- Take advantage of new technologies (chatbots, etc.)
- Build an API first development culture
  - Create a larger collection of reusable components
  - Components that are very specific and easy to test

Architecture

portfolio

High Level Architecture

- Multi layered API
- B2B API provides access to core data and functions
- Application APIs provide subsets and authorization and never touch the database
- Add new APIs at the top level without necessarily making any additions to the B2B layer



portfolio

Security

- OAuth2.0
  - Available out of the box with PASOE and spring security
  - Client Credentials flow for access to B2B API
  - Resource Owner Password flow for access to application APIs
- Authentication performed through a separate web application
  - Typically deployed in the same PASOE instance as the API
  - Could be deployed separately, or centralized, in the future
- Each application API performs specific authorization

Security – OAuth2 Tokens in API Requests

Request an OAuth2 token

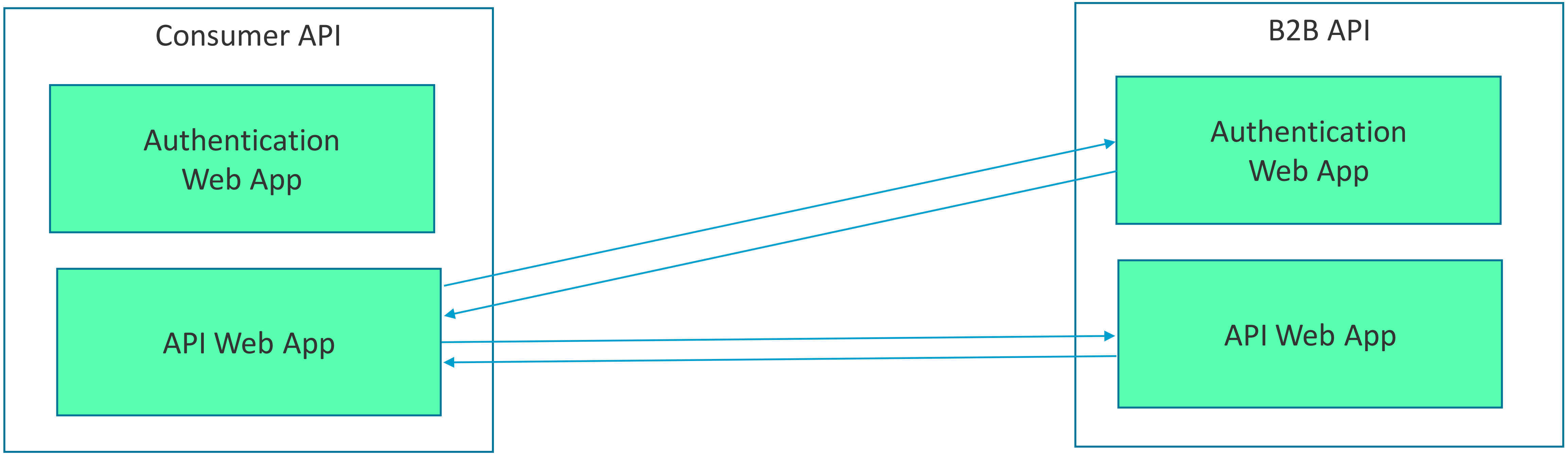
Make API Request using Bearer token

Validate API token and return API Response

Validate and return OAuth2 Token

{ "access_token": <JSON Web Token>, "token_type": "bearer", "expires_in": 600 }

--header authorization: Bearer <JSON Web Token>



Development

ABL WebHandler - Why

- Greater control over URL design with minimal configuration
  - Allowed us to use custom path parameters
  - Ex. /Retail/Accounts
    /Retail/Accounts/{AccountNumber}
- Entire HTTP request pre-parsed
  - Request components available to the WebHandler as an IWebRequest object
- Also considered DataObjectHandler
  - Required more configuration
  - Better suited to existing, re-usable modules (maybe migrating from Classic AppServer)

ABL WebHandler – What's in it?

- Base web handler inherited by API specific web handlers
  - Perform common logging, database connections, etc.
- Can re-use complex, business process procedures
- Defined OOABL patterns for any non-reusable features
  - Created concise, unit testable classes and methods

Challenge

- Legacy logic often includes shared variables and temp-tables
  - Not compatible with our OOABL approach
  - Wrapper procedures define those constructs and run legacy procedures

portfolio

ABL WebHandler - ProDataSets

- Data access architecture built around ProDataSets
- Perform business validations on temp-tables
- Database transactions are short and rarely fail
- All database queries are dynamic

Challenge

- Mindset – need to think about data access differently
  - Dynamic queries were new for many developers
  - Retrieving all data up front and committing it at the end (Stateless)

ABL WebHandler – Configuration in Development

- Must configure each relative URL and its WebHandler
  - Listed in openedge.properties configuration file
- Ordering is important
  - PASOE will run the first handler that matches the relative URL

Challenge

- Initially managed web handler configuration manually
  - Developer updates to openedge.properties, delta file
  - Ordering of web handlers is important - becomes unmanageable

portfolio

ABL WebHandler: Configuration Example

```
[instance.webapp.WEB]
   adapterEnabled=1
   defaultHandler=com.sit.obp.web.OBPDefaultWebHandler
   handler1=web.ClientWebHandler:/Clients/{ClientNumber}
   handler2=web.ClientWebHandler:/Clients
   handler3=web.RetailHandler:/Retail/Accounts/{AccountNumber}
   handler4=web.RetailHandler:/Retail/Accounts
   handler5=web.TransactionHandler:/Retail/Accounts/{AccountNumber}/Txn
```

portfolio

ABL WebHandler: Configuration Example

Request sent to:

http://hostname:port/app/web/Retail/Accounts/123456/Txn

ABL WebHandler: Configuration Example

Request sent to:

http://hostname:port/app/web/Retail/Accounts/123456/Txn

app – web application name

web – PASOE transport

Retail/Accounts/123456/Txn – relative URL

ABL WebHandler: Configuration Example

Request sent to:

http://hostname:port/app/web/Retail/Accounts/123456/Txn

Chosen WebHandler:

web.RetailHandler

ABL WebHandler: Configuration Example

Request sent to:

http://hostname:port/app/web/Retail/Accounts/123456/Txn

Chosen WebHandler:

web.RetailHandler

Why:

handler3=web.RetailHandler:/Retail/Accounts/{AccountNumber}

handler5=web.TransactionHandler:/Retail/Accounts/{AccountNumber}/Txn

portfolio

ABL WebHandler: Configuration Example

```
[instance.webapp.WEB]
   adapterEnabled=1
   defaultHandler=com.sit.obp.web.OBPDefaultWebHandler
   handler1=web.ClientWebHandler:/Clients/{ClientNumber}
   handler2=web.ClientWebHandler:/Clients
   handler3=web.TransactionHandler:/Retail/Accounts/{AccountNumber}/Txn
   handler4=web.RetailHandler:/Retail/Accounts/{AccountNumber}
   handler5=web.RetailHandler:/Retail/Accounts
```

ABL WebHandler – Configuration Solution

- Manage PASOE configuration through Progress Developer Studio
- Build .war file using Progress' Ant task
  - https://documentation.progress.com/output/ua/OpenEdge_latest/index.html#page/pdsoe/packaging-an-abl-web-app-project.html
- Easily add ABL Services
  - Naming convention to guarantee sorting
  - Automated build orders the services, alphabetically by service name

    <system>-<sequence>-<service name>

    Ex. Client-0000-ClientService

    Retail-0000-TransactionService

    Retail-0001-AccountService

Documentation

portfolio

API Documentation

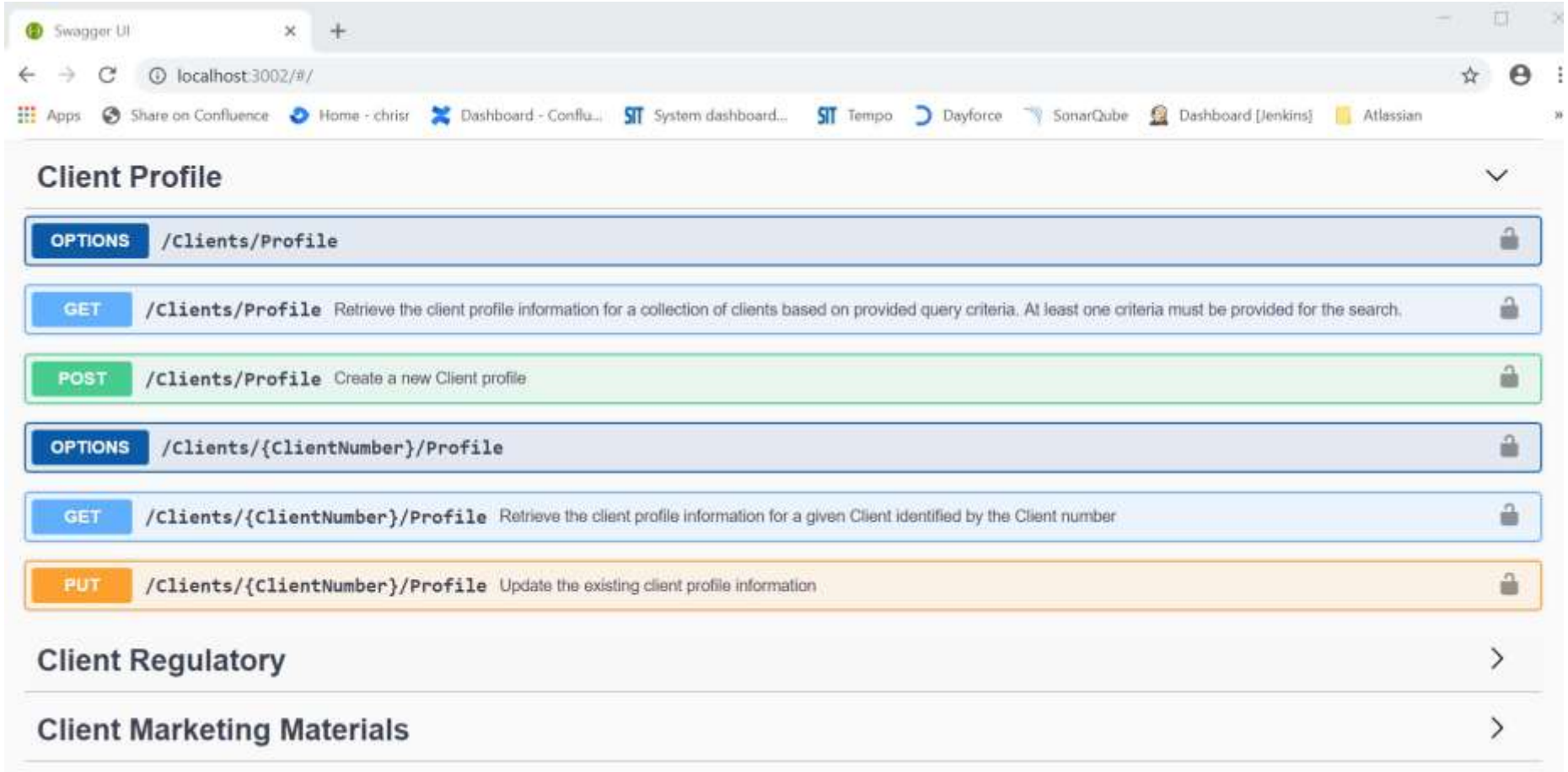
- OpenAPI 3.0 specification
  - https://swagger.io/docs/specification/about/
- Swagger Editor for internal development and testing
- Industry standard specification
- Allowed 3rd party collaboration
  - Partnership with Servoy - +Banking and +Mobile applications
  - Progress chatbot for loan applications

Challenge

- Created documentation during development (too late!)

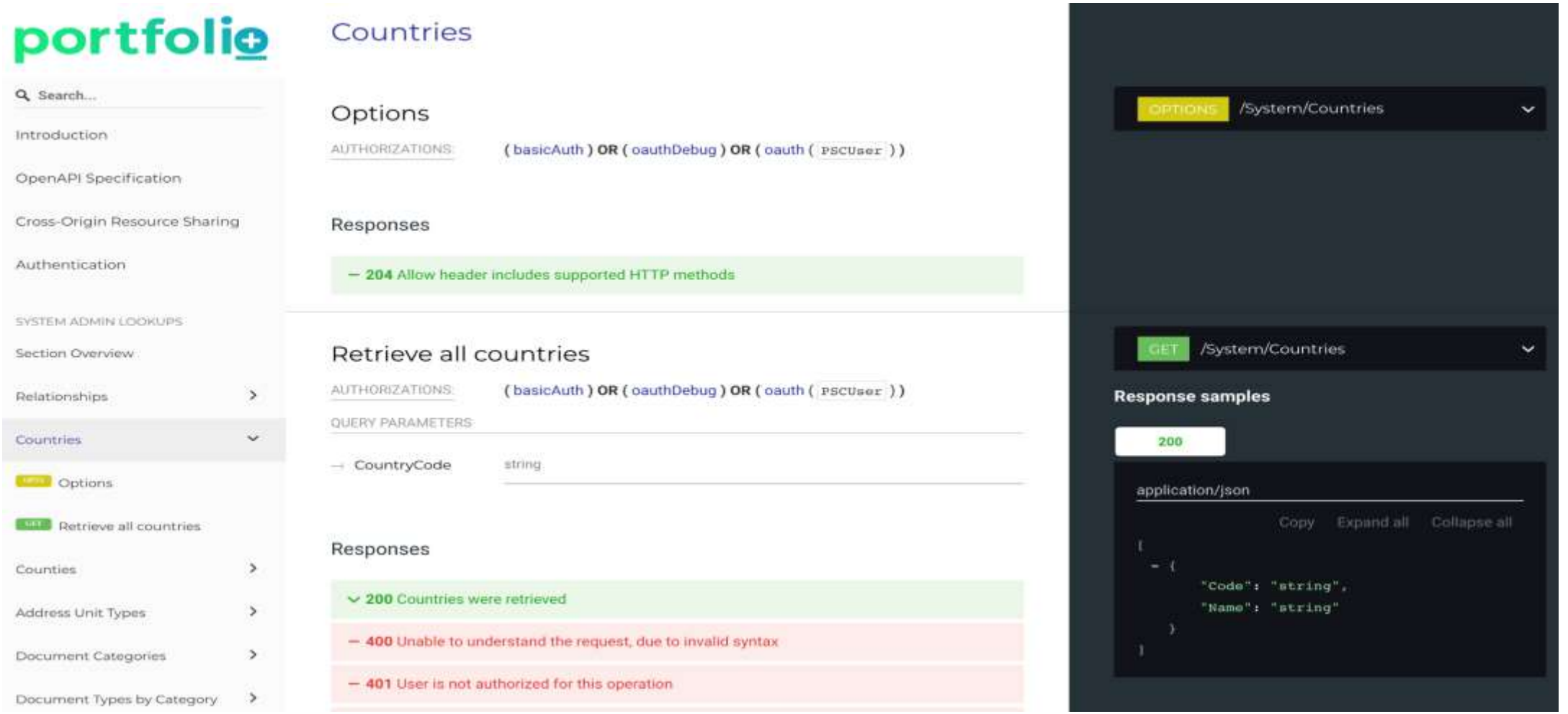
portfolio

Swagger Example



portfolio

ReDoc Example



Continuous Integration & Delivery

portfolio

portfolio

Tools Overview

- Source control/indexing: Apache Subversion, Atlassian Fisheye
- Artifact repository: Apache Archiva
- Linting and code coverage: SonarQube (with Riverside ABL rules)
- Build configuration: Apache Maven
- Automated builds: Jenkins
- Ticketing/bug tracking : Atlassian JIRA
- Code reviews: Atlassian Crucible
- Documentation: Atlassian Confluence

What We Do

- Automate compiles (on Jenkins) every commit (SVN)
  - Uses Maven dependency management
- Push deployable software to an artifact repository (Archiva)
  - Track released versions (which can be included as Maven dependencies)
- Jenkins pipelines capable of provisioning VMs
  - Deploy legacy and web, API applications as well as databases
  - In house scripts tested regularly and reusable for client deployments
  - *TIP: With pasman/tcman use pasoestart instead of stop and start*

Project Conclusions & Future

portfolio

Where We Are Now

- ~300 API endpoints created
  - Running ~1000 unit tests
- +Banking and +Mobile applications
  - Fully utilizing the REST API
- Some UI components of the legacy application now call the REST API

Future

- Expand our application API layer
  - Expand the API's authorization capabilities
  - Implement API documentation up front to enable Test Driven Development
  - Adopt OEAG for authentication and SSO
- Improve continuous deployment with Docker
  - PASOE applications in all environments
  - Database containers for short lived, internal environments
- Expanding our market to clients to build their own UI
  - Startups… not just financial institutions
  - Providing a banking engine, not just a GUI application

portfolio

Thank You!

Chris Riddell
Chris.Riddell@portfolioplus.com

David Atkins
datkins@progress.com

You might also like to attend

376: “REST API Documentation using Swagger”
Martyn Kemp, Consultingwerk – 13:00 Monday

215: “Doing More With the Spring Framework in Progress Application Server for OpenEdge”
Chad Thomson, Progress Software, Inc. – 09:45 Tuesday

224: “OpenApi (Swagger) to ABL”
Martyn Kemp, Consultingwerk – 09:45Tuesday

340: “Patterns for Migrating Fat Client GUI Applications to N-Tier, Web Applications”
Mike Fechner, Consultingwerk – 11:00 Tuesday