Demystify OpenEdge REST Services

- David Atkins, Principal Solutions Architect
- Dan Mitchell, Principal Sales Engineer

25 October 2018
Agenda

▪ Why bother? How REST applies to business applications

▪ What is REST?

▪ What are the options to ‘RESTify’ OpenEdge applications

▪ Which approach(es) should I use?

▪ How to implement each approach?
  • (aka live demos that will doubtless go horribly wrong…)
Why is REST Important?
REST Use Cases

Modern Web Interfaces

Mobile Apps

Emerging Technologies

Application Integration

Modularization through Microservices

Self-Service BI and Analytics
REST Use Cases

- Modern Web Interfaces
- Mobile Apps

Service Catalog driven tooling integration
REST Use Cases

Modern Web Interfaces

Mobile Apps

Emerging Technologies

Modern languages
REST Use Cases

Modern Web Interfaces

Mobile Apps

Emerging Technologies

Application Integration

Custom & Standardized
REST Use Cases

- Modern Web Interfaces
- Mobile Apps
- Emerging Technologies
- Application Integration
- Modularization through Microservices
REST Use Cases

- Modern Web Interfaces
- Mobile Apps
- Emerging Technologies
- Application Integration
- Modularization through Microservices
- Self-Service BI and Analytics

Standardized REST
OpenEdge Reference Architecture

Presentation (UI) | Enterprise Services (API)

Service Interfaces

Business Components

Workflow | Tasks | Entities

Data Access

Data Sources

Domain Services (*aaS)

SI layer provides:
- Mapping, Translation and Auth*
- SI is separate from biz logic
- Can be multiple SIs to the same biz logic
What is REST?
What is REST?

REST = REpresentational State Transfer

REST is an architectural style for network based software that requires stateless, cacheable, client-server communication via a uniform interface between components.

“HTTP with strong constraints”

- **Resources** are named using a URL
- Supports many representations of data: **JSON**, XML, multi-part
- **Uniform interface** of HTTP Verbs: GET, PUT, POST, DELETE…etc.
- **Stateless**, no client context between requests
- Designed for **performance & scalability**, i.e. supports caching
- In practice there are ‘degrees of RESTfulness’
The Web Is Built on REST

- Browser requests are GETs:
- Type **www.progress.com/next** in your browser, and what gets sent is this:

```
GET /next HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
Accept-Language: en-US
Accept-Encoding: gzip, deflate
Host: www.progress.com
```
What Options are There to ‘RESTify’ OpenEdge?
REST Service Interface Options for OpenEdge

- Data Object Services (using REST transport)
- Data Object Services (using WEB transport)
- Mapped RPC REST Service (using REST transport)
- Custom/DIY WebHandler (using WEB transport)
- Data Object Handler WebHandler (using WEB transport)
- OData view of OpenEdge DB (using Hybrid Data Pipeline)
Data Object Services Options

- Data Object Services (using REST transport)
- Data Object Services (using WEB transport)
- Mapped RPC REST Service (using REST transport)
- Custom/DIY WebHandler (using WEB transport)
- Data Object Handler WebHandler (using WEB transport)
- OData view of OpenEdge DB (using Hybrid Data Pipeline)
PAS for OpenEdge Architecture (v11.6+)

Service interfaces

- Transform data
  - Validation
  - Request & response
- Route requests
- Error handling

REST ADAPTER
- Transport: /rest
- Tech: Java
- Config: .paar file
- Artifacts: .paar file

WEB HANDLER
- Transport: /web
- Tech: ABL
- Config: openedge.properties
- Artifacts: ABL classes

Business logic (ABL)
- CustomerData.cls
- EmployeeData.cls

PASOE Server Instance

Tomcat Web Server

Containers

Artifacts

Service Interface (ABL)
- EmployeeWebHandler.cls
Non-PAS Architectures & PASOE 11.5

Tomcat Server

Service interfaces
- Transform data
  - Validation
  - Request & response
- Route requests
- Error handling

REST ADAPTER
- Transport: /rest
- Tech: Java
- Config: .paar file
- Artifacts: .paar file

Classic AppServer

Business logic (ABL)
- CustomerData.cls
- EmployeeData.cls

Classic WebSpeed

- CGI for comms
- GET, POST only

WEBSPEED AGENT
- Tech: ABL
- Config: ubroker.properties
- Artifacts: ABL P-code
Service Interface Approaches

- **Data Object (REST)**
- **Data Object (WebHandler)**
- **REST (Mapped RPC)**
- **WebHandler**
- **DataObjectHandler**

- Formerly Mobile Services
- Annotate certain methods (w/ particular signatures)
- Very prescriptive
  - Programming model
  - URI paths
- Uses REST transport
- Creates Data Service Catalog as public API
# Service Interface Approaches

<table>
<thead>
<tr>
<th></th>
<th>Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Data Object (REST)</td>
</tr>
<tr>
<td>b</td>
<td><strong>Data Object (WebHandler)</strong></td>
</tr>
<tr>
<td>c</td>
<td>REST (Mapped RPC)</td>
</tr>
<tr>
<td>d</td>
<td>WebHandler</td>
</tr>
<tr>
<td>e</td>
<td>DataObjectHandler</td>
</tr>
</tbody>
</table>

- As of 11.6.3
- Annotate certain methods (w/ particular signatures)
- Quite prescriptive
- More flexibility in mapping
- Uses WEB transport
- Requires PAS for OpenEdge
- Creates Data Service Catalog as public API
Conference - ERD

A Speaker submits a Talk which is scheduled to a Timeslot in a Room and is given a Rating by an Attendee.
Demo 1: Data Object Service
Data Object Service Interaction

UI

```
http://
```

progress.jsdo.js

GET <webapp>/web/pdo/ConferenceSvc/TalksBE/talks

ConferenceSvc.json {catalog}

SI

```
handler1=OpenEdge.Web.DataObject.DataObjectHandler: /pdo/
```

DataObjectHandler

ServiceRegistry

BL

```
TalksBE.cls
ReadTalks(…)
CreateTalks(…)
```

HTTP to ABL

ConferenceSvc.gen { mapping }

© 2017 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.
Custom REST Service Interface Options

- Data Object Services (using REST transport)
- Data Object Services (using WEB transport)
- Mapped RPC REST Service (using REST transport)
- Custom/DIY WebHandler (using WEB transport)
- Data Object Handler WebHandler (using WEB transport)
- OData view of OpenEdge DB (using Hybrid Data Pipeline)
Service Interface Approaches

- **Data Object (REST)**
- **Data Object (WebHandler)**
- **REST (Mapped RPC)**
- **WebHandler**
- **DataObjectHandler**

- Formerly REST Services
- Graphical mapping tool
- Uses REST transport
- Flexible in URI paths
MappedRPC REST Graphical Mapper
# Service Interface Approaches

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
<th>Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Data Object (REST)</td>
<td>11.2.0, 11.3.0, 11.4.0</td>
</tr>
<tr>
<td>b</td>
<td>Data Object (WebHandler)</td>
<td>11.5.0, 11.6.0, 11.6.3</td>
</tr>
<tr>
<td>c</td>
<td>REST (Mapped RPC)</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>WebHandler</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>DataObjectHandler</td>
<td></td>
</tr>
</tbody>
</table>

- Associate an OOABL WebHandler class with a URI pattern
- Uses WEB transport
- Requires PAS for OpenEdge
- VERY flexible, URI is all yours
- Do whatever you want in code/ABL
- In-the-box versions
  - `OpenEdge.Web.WebHandler`
  - `OpenEdge.Web.CompatibilityHandler`
  - `OpenEdge.Web.DefaultHandler`
Demo 2: DIY WebHandler
DIY WebHandler Interaction

UI

http://

GET <webapp>/web/talks

SI

handler1=OpenEdge.Web.DataObject.TalksWebHandler: /talks/

TalksWebHandler.cls

BL

read_talk.p
create_talk.p

HTTP to ABL
### Service Interface Approaches

<table>
<thead>
<tr>
<th></th>
<th>Data Object (REST)</th>
<th>Data Object (WebHandler)</th>
<th>REST (Mapped RPC)</th>
<th>WebHandler</th>
<th>DataObjectHandler</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Pre-built generic WebHandler
- Mapping defined in JSON file
- VERY Flexible
- Requires PAS for OpenEdge
- Not fully documented YET

© 2017 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.
DataObjectHandler WebHandler Interaction

UI

http://
GET <webapp>/web/conf/talks

SI

handler1=OpenEdge.Web.DataObject.DataObjectHandler: /conf/

DataObjectHandler.cls

ServiceRegistry

BL

read_talk.p
create_talk.p

conf.map { mapping }
DataObjectHandler: What can I map?

```
"/talks/{talk-id}": {
  "POST": {
    "contentType": "application/json",
    "statusCode": 201,
    "options": {
      "responseEnvelope": true
    },
    "entity": {
      "name": "logic/talk/new_talk.p",
      "function": "add_talk",
      "arg": [{
        "ablName": "ttTalk",
        "ioMode": "INPUT",
        "ablType": "table",
        "msgElem": { "type": "body", "name": null } }
    }, {
      "ablName": "pcChar",
      "ioMode": "output",
      "ablType": "character",
      "msgElem": { "type": "header", "name": "location" }
    }
  }
},
```

URI mapping
```
/talks
/{service}/data/{resource}
/{collection}//{coll-id}
```

Status codes
```
202 / Accepted
418 / I'm a teapot
```

Envelopes
```
requestEnvelope : "input"
errorEnvelope : "oops"
```

IO Modes
```
"input" "output" "input-output" "return"
```

ABL data types (also extent variants)
```
"character", "longchar", "integer", "int64", "decimal", "logical", "rowid", "recid", "date", "datetime", "datetime-tz", "raw", "memptr", "dataset", "temp-table",
"class <ooabl.type.name>"
```

HTTP Message elements
```
Request-only "path", "query", "httpMethod", "request", "constant"
Response-only "none", "statusCode", "statusReason"
Both "cookie", "header", "field", "body"
```
Standardized REST Service Interface Options

- Data Object Services (using REST transport)
- Data Object Services (using WEB transport)
- Mapped RPC REST Service (using REST transport)
- Custom/DIY WebHandler (using WEB transport)
- Data Object Handler WebHandler (using WEB transport)
- OData view of OpenEdge DB (using Hybrid Data Pipeline)
Standardized REST: OData

An open protocol to allow the creation and consumption of queryable and interoperable RESTful APIs in a simple and standard way.

OASIS Standard REST API (“SQL for the web”)
• Uniform URL conventions
• Standard Query String operations
• Surface metadata in standard way
• Operations built on REST principles

Started in 2007 by Microsoft
OASIS Standard since Feb 2014
ISO ratified in Feb 2017

Progress DataDirect the first member to join OData Technical Committee
OData access to OpenEdge – Direct to Database

All connections can be encrypted

Cloud-host application (i.e. Salesforce / Dynamics)

On-Premises

OpenEdge

JDBC

OE

DB

OData (HTTPS)

© 2017 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.
Which option(s) should I choose?
If you are building Web or Mobile UIs…

Then we recommend…

- Data Object Services (using WEB transport) & PAS for OpenEdge
- Data Object Services (using REST transport) if Classic AS

- Data Object Handler WebHandler (WEB) & PAS for OpenEdge
- DIY WebHandler (using WEB transport) & PAS for OpenEdge
- Mapped RPC REST Service (using REST transport)

- OData view of OpenEdge DB (using Hybrid Data Pipeline)
If you are building a custom B2B REST API?

- Data Object Services (using WEB transport) & PAS for OpenEdge
- Data Object Services (using REST transport) if Classic AS

Then we recommend...

- DIY WebHandler (using WEB transport) & PAS for OpenEdge
- DataObjectHandler WebHandler (WEB) & PAS for OpenEdge
- Mapped RPC REST Service (using REST transport) if Classic AS
- OData view of OpenEdge DB (using Hybrid Data Pipeline)
If you need to expose ‘standardized’ REST?

- Data Object Services (using WEB transport) & PASOE
- Data Object Services (using REST transport) if Classic AS

- DataObjectHandler WebHandler (WEB) & PAS for OpenEdge
- DIY WebHandler (using WEB transport) & PAS for OpenEdge
- Mapped RPC REST Service (using REST transport) if Classic AS

- OData view of OpenEdge DB (using Hybrid Data Pipeline)