REST in Peace
Mastering the JSDO with a Dynamic ABL backend

Mike Fechner, Consultingwerk Ltd.
mike.fechner@consultingwerk.de
Consultingwerk Ltd.

- Independent IT consulting organization
- Focusing on OpenEdge and related technology
- Located in Cologne, Germany
- Customers in Europe, North America, Australia and South Africa
- Vendor of tools and consulting programs
- 26 years of Progress experience (V5 … OE11)
- Specialized in GUI for .NET, OO, Software Architecture, Application Integration

http://www.consultingwerk.de/
Agenda

- JSDO / Kendo UI Data Source
  - REST Web Services
  - REST Adapter for Data Object Services
  - JSDO Backend Methods
  - Dynamic REST Adapter Backend
  - WebSpeed Web Handlers (11.6)
  - Dynamic WebSpeed based Backend
<table>
<thead>
<tr>
<th>Cust Num</th>
<th>Kundennname</th>
<th>Country</th>
<th>Stadt</th>
</tr>
</thead>
<tbody>
<tr>
<td>1025</td>
<td>Athlete's Track</td>
<td>USA</td>
<td>Vienna</td>
</tr>
<tr>
<td>1030</td>
<td>Soccer Universe</td>
<td>USA</td>
<td>Oak Brook</td>
</tr>
<tr>
<td>1045</td>
<td>Play Sports</td>
<td>USA</td>
<td>Boston</td>
</tr>
<tr>
<td>182340</td>
<td>BASCO</td>
<td>USA</td>
<td>Amherst</td>
</tr>
<tr>
<td>408860</td>
<td>University Stereo</td>
<td>USA</td>
<td>Amherst</td>
</tr>
<tr>
<td>473170</td>
<td>Balanced Fortune</td>
<td>USA</td>
<td>Amherst</td>
</tr>
<tr>
<td>617690</td>
<td>Friendly Advice</td>
<td>USA</td>
<td>Mountain Green</td>
</tr>
<tr>
<td>960540</td>
<td>Express Merchant Service</td>
<td>USA</td>
<td>Amherst</td>
</tr>
</tbody>
</table>
JSDO

- JavaScript Library to provide access for JavaScript (Web Browser, Mobile, Rollbase) clients to OpenEdge Data Object Services (Business Entities)
- Introduced in OpenEdge 11.2 for OpenEdge Mobile
- Included in Telerik Platform
- Included in Rollbase
- Can be used with any JavaScript client
- Github, Apache license, royalty free
Kendo UI DataSource for JSDO

- Integrates JSDO into the Kendo UI framework
- Extends Kendo UI Data Source
- Included in the JSDO library
- Query manipulation (Kendo filters to ABL query string)
- Manages ProDataset before-image
Kendo UI DataSource

```javascript
var serviceURI = "http://localhost:8980/SmartJsdoBackendService",
    jsdoSettings = {
        serviceURI: serviceURI,
    },
    promise;

// create a new session object
jsdosession = new progress.data.JSDOSession(jsdoSettings);
promise = jsdosession.login("", "");
promise.done(function(jsdosession, result, info){
    jsdosession.addCatalog(jsdoSettings.catalogURIs)
    .done(function(jsdosession, result, details){

        dataSource = new kendo.data.DataSource( {
            type: "jsdo",
            serverPaging: true,
            serverFiltering: true,
            serverSorting: true,
            // sort: { field: "State", dir: "desc" },
            pageSize: 10,
            transport: {
                tableRef: "eCustomer",
                countFnName: "count"
            },
            error: function(e) {
                console.log ("Error: ", e);
            }
        }
    });
```
Agenda

- JSDO / Kendo UI Data Source
- **REST Web Services**
  - REST Adapter for Data Object Services
  - JSDO Backend Methods
  - Dynamic REST Adapter Backend
  - WebSpeed Web Handlers (11.6)
  - Dynamic WebSpeed based Backend
REST protocol

- **RE**presentational **S**tate **T**ransfer
- W3C standard
- Typically http/1.1 transport
- Simpler than SOAP web services
- Client and Server communicate about the state of an object
- State transitions as the message
- Client may request (GET) using URI
- Client may post using request content
REST „verbs“

- Additional http REQUEST_METHOD’s
- Multiple interactions on the same URI
- **GET** – client requests resource (record), should not modify the resource
- **POST** – client posts a new instance of the resource (create a record)
- **PUT** – client posts a modification of a resource (update record)
- **DELETE** – client requests deletion of a resource
- ...
JSON – JavaScript Object Notation

- The “little brother of XML“
- Origianates from JavaScript development
- JavaScript objects can be written to and read from JSON, also supported in other languages
- More lightweight, typically smaller than XML (no need for end tag), easier human readable
- OE10.2B, support for READ/WRITE-JSON of ProDataset and Temp-Table
- OE11.0, support for JSON ObjectModel Parser
- OE11.2, document format of the REST Adapter
- Mime-Type: application/json
Sample ProDataset JSON output

- {} wraps a single object
- [] wraps an array of objects
- All strings are quoted
- Data types: Number, String, Boolean, Array, Object, Null
- Everything else must be passed as a String (e.g. Date)
- No real standard for Date
JSON Catalog

- Describes capabilities of OpenEdge backend resource to JSDO
- Methods
  - create
  - read
  - update
  - delete
  - submit
  - count
  - custom operations
ProDataset Schema definition

```json
{
    "_errorString": {
        "type": "string"
    },
    "CustNum": {
        "type": "integer",
        "abiType": "INTEGER",
        "default": 0,
        "title": "Cust Num"
    },
    "Country": {
        "type": "string",
        "abiType": "CHARACTER",
        "default": "USA",
        "title": "Country"
    },
    "Name": {
        "type": "string",
        "abiType": "CHARACTER",
        "default": "",
        "title": "Kundennname"
    },
    "Address": {
        "type": "string",
        "abiType": "CHARACTER",
        "default": "",
        "title": "Address"
    },
    "Address2": {
        "type": "string",
        "abiType": "CHARACTER",
        "default": "",
        "title": "Address2"
    },
    "City": {
        "type": "string",
        "abiType": "CHARACTER",
        "default": "",
```
List of supported operations

```json

"operations": [

{

"name": "count",
"path": "/count?filter={filter}",
"useBeforeImage": false,
"type": "invoke",
"verb": "put",
"params": [

]

},
{

"path": "/",
"useBeforeImage": true,
"type": "update",
"verb": "put",
"params": [

{

"name": "dsCustomer",
"type": "REQUEST_BODY"

}

],

{

"path": "",

}

}]

```
Agenda

- JSDO / Kendo UI Data Source
- REST Web Services
- **REST Adapter for Data Object Services**
- JSDO Backend Methods
- Dynamic REST Adapter Backend
- WebSpeed Web Handlers (11.6)
- Dynamic WebSpeed based Backend
REST Adapter

- JavaServlet that translates REST messages into AppServer calls
- Similar to WSA and AIA
- Tooling integrated into Progress Developer Studio
- Not integrated into ProxyGen
- Can be deployed on standard Tomcat
- Integrated in PASOE as the REST transport
Project name: DataObjectService

Use default location

Location: C:\Work\OpenEdge116_64\DataObjectService

Project type configuration

ABL Web App

OpenEdge project specialized for one or more ABL services deployed as a single Web App to Pacific Application Server for OpenEdge.
Provide ABL Web App deploy details
Select to publish the WAR file to the ROOT web app folder incrementally.

Web Application
- Deploy as default (ROOT)
- Deploy as WebApp [DataObjectService]

Business Logic
- Module name: [DataObjectService]
- ABL Source folder: [AppServer]

Supported servers:
- [oepas1 in consultingwerk1.oepas1 (Pacific Application Server for OpenEdge 11.6ALPHA)]
REST in peace
@program FILE(name="CustomerEntity.cls", module="AppServer").
@openapi.openedge.export FILE(type="REST", executionMode="singleton", useReturnValue="false", writeDataSetBeforeImage="true").
@progress.service.resource FILE(name="CustomerEntity", URI="/CustomerEntity", schemaName="dsCustomer", schemaFile=

USING Progress.Lang.*.

BLOCK-LEVEL ON ERROR UNDO, THROW.

CLASS CustomerEntity INHERITS BusinessEntity:
    /*
     * Purpose:
     * Notes:
     */

{"customerentity.i"}


@openapi.openedge.export(type="REST", useReturnValue="false", writeDataSetBeforeImage="true").
@progress.service.resourceMapping(type="REST", operation="read", URI="/?filter=~{filter~}"). all

METHOD PUBLIC VOID ReadCustomerEntity(
    INPUT filter AS CHARACTER,
    OUTPUT DATASET dsCustomer):

    SUPER:ReadData(filter).

END METHOD.
REST in peace
```
{
    "version": "1.2",
    "lastModified": "Sun Jun 26 20:37:47 CEST 2016",
    "services": [
        {
            "name": "DataObjectService",
            "address": "\rest\DataObjectService",
            "useRequest": true,
            "resources": [
                {
                    "name": "CustomerEntity",
                    "path": "\CustomerEntity",
                    "autoSave": false,
                    "schema": {
                        "type": "object",
                        "additionalProperties": false,
                        "properties": {
                            "dsCustomer": {
                                "type": "object",
                                "additionalProperties": false,
                                "properties": {
                                    "ttCustomer": {
                                        "type": "array",
                                        "primaryKey": ["CustNum"],
                                        "items": {
                                            "additionalProperties": false,
                                            "properties": {
                                                "id": {"type": "string"}.
```

REST in peace
Agenda

- JSDO / Kendo UI Data Source
- REST Web Services
- REST Adapter for Data Object Services
- **JSDO Backend Methods**
  - Dynamic REST Adapter Backend
  - WebSpeed Web Handlers (11.6)
  - Dynamic WebSpeed based Backend
JSKO Backend Methods

- JSKO calls into Data Object Services, or Business Entities
- Real world scenario: Service Interface to Business Entity (see OEAA, OERA, CSS)
- Progress provides base class `OpenEdge.BusinessLogic.BusinessEntity` as a starting point for quick prototyping
  - Optional foundation for implementation
  - Suited for rapid prototyping
Read method

- GET http://localhost:8820/web/Resource/CustomerBusinessEntity?filter=...

- filter as single CHARACTER input parameter
- Can be ABL query string: “Name BEGINS ‘L’ AND City BEGINS ‘Bos’”
- Can be JSON Object
- ProDataset as Output Parameter
JFP Pattern

- JavaScript Filter Pattern
- JSON Object provided as filter parameter (CHARACTER)
- Used by the Kendo UI DataSource
- Allows more flexible, structured filter parameter

```json
filter=
{"ablFilter":"(Name BEGINS 'l' and City BEGINS 'b')",
 "skip":30,
 "top":10}
```
Count request


- Kendo UI DataSource asks for number of result records from backend
- Populate the paging buttons below the grid
- Good for UX
- May be challenging to implement with ABL
Demo

- Google Chrome Debugger
  - JSDO Read requests
  - Count Request
- F12 Developer Tools
Create / Delete / Update

- Create: POST (single record)
- Delete: DELETE (single record)
- Update: PUT (single record)

- Submit: PUT (multiple records)
Create / Delete / Update

- Request body contains JSON representation of ProDataset
- 1 record (create, delete, update)
- Multiple records (submit)
- ProDataset JSON including JSON before-image

```ruby
/*
Purpose: Update one or more records
Notes:
*/

@openapi.openedge.export(type="REST", useReturnValue="false", writeDataSetBeforeImage="true").
@progress.service.resourceMapping(type="REST", operation="update", URI="", alias="", mediaType="").
METHOD PUBLIC VOID UpdateCustomerEntity(INPUT-OUTPUT DATASET dsCustomer):
```
Agenda

- JSDO / Kendo UI Data Source
- REST Web Services
- REST Adapter for Data Object Services
- JSDO Backend Methods
  - **Dynamic REST Adapter Backend**
  - WebSpeed Web Handlers (11.6)
  - Dynamic WebSpeed based Backend
Why dynamic?

- REST Adapter tooling in PDSOE
  - problematic for large projects
  - problematic for multi-developer environments (version control, conflict resolution, merge)
- Each time a single field in a Business Entity changes the whole service needs to be redeployed
- Redeploying REST services from PASOE not always smooth …
Why dynamic?

- Eliminate the need to redeploy REST resources when Business Entities are added or removed
- Localized catalog, generated on the fly
- Include application specific attributes in catalog
- Better control over authorization
- Don’t include catalog for Business Entities the consumer has no authorization for
- Easier to split up Catalog by Business Entity (faster load time)
Dynamic Backend

- Catalog retrieved via http GET request
- URI included in JavaScript code, not specific to the “Data Object Service” project type
- Resource data retrieved via http GET, PUT, POST, DELETE request
- URI for resource access described by data catalog, not specific to the “Data Object Service”
- All messages (in/out) are JSON messages
- Full freedom over URI format
REST Adapter based Backend

- Classic AppServer: “REST” style OpenEdge Project in Progress Developer Studio
REST Adapter based Backend

- PASOE: “ABL Web App“ style OpenEdge Project in Progress Developer Studio
- Service Type “REST (Mapped RPC)“
REST in peace
REST in peace
REST URI Mapping

- ABL class method used to handle requests to specific URI pattern and http method
- Drag and drop mapping of request parameters and URI parts to ABL method parameters
  - URI path parameter
  - Query String parameters
  - Request body, body parts
- Client (JSDO) cannot distinguish if it’s speaking to “Mapped RPC” or “Data Object Service”
Catalog Access

Catalog access

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource URI</td>
<td>/Catalog/{EntityName}</td>
</tr>
<tr>
<td>Verb='GET'</td>
<td>Consultingwerk.OREA.JsdoGenericService.Catalog..GetCatalogForBusinesEntity</td>
</tr>
</tbody>
</table>

Input Parameter Mapping

Mapping Definitions

- **Parameters**
  - (java:String)

- **Interface Parameters**
  - pcEntityName

- **Path Parameters**
  - EntityName

- **Query String Parameters**

- **Complete URL**

- **HTTP Message**
  - Method
  - Headers
    - Cookies

- **Server Contexts**
  - Servlet Request
  - Servlet Response
  - Servlet Context
  - Servlet Config
Catalog Access

- URI Pattern /rest/Catalog/{EntityName}
- Value of EntityName in URI will be passed as INPUT Parameter to ABL method

- Replacing separate entry points in the REST Adapter with fewer entry points and an additional parameter

- /Catalog/CustomerBusinessEntity -> Parameter value of “CustomerBusinessEntity”
Demo

- Code review dynamic catalog generation
Resource access

- Read
- Create/Update/Delete
- Submit
- Count
Resource Read Access

- `/rest/Resource/{EntityName}?filter={filter}`
/*
Purpose: Generic GetData (get/read) Service Interface
Notes:
@param pcEntityName The name of the Business Entity
@param pcFilter The filter parameter
@param pcNumRecords The numRecords value from the http client, see method ParseNumRecords
@param pcStopAfter The stopAfter value from the http client, see method ParseStopAfter
@param phDataset The Dataset to return to the client
*/

@openapi.openedge.export(type="REST", useReturnValue="false", writeDataSetBeforeImage="true").

METHOD PUBLIC VOID GetData (pcEntityName AS CHARACTER,
                          pcFilter AS CHARACTER,
                          pcNumRecords AS CHARACTER,
                          pcStopAfter AS CHARACTER,
                          OUTPUT DATASET-HANDLE phDataset):

ServiceInterface:FetchData (pcEntityName,
                          oFetchDataRequest,
                          OUTPUT DATASET-HANDLE hFetchDataset) .
Update methods

- Update methods are implemented similar to read requests
- Dataset passed as LONGCHAR, to ensure numeric values are converted to the right ABL type

```plaintext
/*
 * Purpose: Generic UpdateData (put/update) Service Interface
 * Notes: Uses a LONGCHAR as INPUT-OUTPUT for the Dataset, to allow to map this to the actual data types of the fields in the Dataset from the business entity.
 * The default mapping for JSON number would be a DECIMAL fields, which would cause conflicts integer fields in the business entity while updating
 * @param pcEntityName The name of the Business Entity
 * @param lcDataset The JSON Representation of the Dataset to update
 */

@openapi.openedgex.export(type="REST", useReturnValue="false", writeDataSetBeforeImage="true").
METHOD PUBLIC VOID UpdateData (pcEntityName AS CHARACTER,
                               INPUT-OUTPUT lcDataset AS LONGCHAR):

   THIS-OBJECT:ProcessUpdate (pcEntityName, INPUT-OUTPUT lcDataset).
END METHOD.
```
Demo

- Walk through Resource access Service.cls
Agenda

- JSDO / Kendo UI Data Source
- REST Web Services
- REST Adapter for Data Object Services
- JSDO Backend Methods
- Dynamic REST Adapter Backend

- WebSpeed Web Handlers (11.6)
- Dynamic WebSpeed based Backend
Web handler

- Web handlers provide a very flexible way to handle web requests
- Synchronous request-response pattern
- Supports html page generation
- Supports service requests as well
- Flexible enough to provide an alternative to the REST Adapter and Web Services Adapter (SOAP)
- ABL classes, extending OpenEdge.Web.WebHandler
# OpenEdge.Web.WebHandler

## Method Summary

<table>
<thead>
<tr>
<th>Options</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HandleDelete (IWebRequest)</td>
</tr>
<tr>
<td></td>
<td>HandleGet (IWebRequest)</td>
</tr>
<tr>
<td></td>
<td>HandleHead (IWebRequest)</td>
</tr>
<tr>
<td>A</td>
<td>HandleNotAllowedMethod (IWebRequest)</td>
</tr>
<tr>
<td>A</td>
<td>HandleNotImplemented (IWebRequest)</td>
</tr>
<tr>
<td></td>
<td>HandleOptions (IWebRequest)</td>
</tr>
<tr>
<td></td>
<td>HandlePatch (IWebRequest)</td>
</tr>
<tr>
<td></td>
<td>HandlePost (IWebRequest)</td>
</tr>
<tr>
<td></td>
<td>HandlePut (IWebRequest)</td>
</tr>
<tr>
<td>#</td>
<td>HandleRequest ()</td>
</tr>
<tr>
<td></td>
<td>HandleTrace (IWebRequest)</td>
</tr>
</tbody>
</table>
Web handler

- WebSpeed in PASOE brings request handler mapping out of the box (classic Web Speed requires customization of web-disp.p for this)
- Based on configuration in openedge.properties
- New PDSOE project type ABL Web Application creates and registers a single handler
- Additional handlers can be set up in OpenEdge Management
URL Mapping

- Configuration based
- Tomcat parses request URI for patterns
  - http://localhost/web/Customer/1
- More "rest-style" URI's
- Higher ranking in search engines compared to
  - http://localhost/cgi-bin/cgiip.exe/Customer.w?CustNum=1

- Request handler are specialized ABL classes
URL Mapping

```
[opas1.ROOT.WEB]
  adapterEnabled=1
  defaultCookieDomain=
  defaultCookiePath=
  defaultHandler=OpenEdge.Web.CompatibilityHandler
  handler1=JsonDataHandler: /JsonData/{BusinessEntityName}
  handler2=CustomerListWithSearchHandler: /CustomerSearch
  handler3=CustomerHandler: /Customer/{CustNum}
  handler4=DemoHandler: /Demo
  handler5=nullHandler: /AblWebAppProject
  handler6=nullHandler: /Data
  srvrDebug=0
```
Sample request handler

```c
METHOD OVERRIDE PROTECTED INTEGER HandleGet (poRequest AS OpenEdge.Web.IWebRequest):

DEFINE VARIABLE response AS OpenEdge.WebWebResponse NO-UNDO.
DEFINE VARIABLE writer AS OpenEdge.WebWebResponseWriter NO-UNDO.

DEFINE VARIABLE jsonObject AS JsonObject NO-UNDO.
DEFINE VARIABLE iCustNum AS INTEGER NO-UNDO.
DEFINE VARIABLE cCustNum AS CHARACTER NO-UNDO.

EMPTY TEMP-TABLE ttCustomer.

response = NEWWebResponse().
writer = NEWWebResponseWriter(response).

iCustNum = poRequest:GetPathParameter("CustNum") .
ASSIGN iCustNum = INTEGER (cCustNum) NO-ERROR .

jsonObject = NEWJsonObject().
jsonObject:READ(TEMP-TABLE ttCustomer:HANDLE).

response:ContentType = "application/json".

writer:Write(jsonObject:GetJsonText()).
writer:Flush().
writer:Close().

RETURN 0.
```
Agenda

- JSDO / Kendo UI Data Source
- REST Web Services
- REST Adapter for Data Object Services
- JSDO Backend Methods
- Dynamic REST Adapter Backend
- WebSpeed Web Handlers (11.6)

- Dynamic WebSpeed based Backend
REST Adapter vs. WebHandler

- REST Adapter supported since OpenEdge 11.2, JSON ProDataset before-image since OpenEdge 11.4
- Classic and PASOE AppServer
- WebHandlers available since OpenEdge 11.6 only
- PASOE AppServer only
REST Adapter vs. WebHandler

- WebHandler offer greater flexibility in handling input and output
- WebHandler provide access to full HTTP protocol without specific parameter mapping
- WebHandler can handle all content types from the same backend address
  - Eliminates CORS issues
- Use case: Angular JS application where page HTML fragments are generated on server
- 100% ABL source code and openedge.properties
openedge.properties

```
[smartpas.ROOT.WEB]
adapterEnabled=1
srvrAppMode=development
wsRoot=/static/webspeed
srvrDebug=1
defaultCookieDomain=
defaultCookiePath=
defaultHandler=OpenEdge.Web.CompatibilityHandler
handler1=Consultingwerk.OERA.JsdoGenericService.WebHandler.CatalogWebHandler: /Catalog/{EntityName}
handler4=Consultingwerk.OERA.JsdoGenericService.WebHandler.InvokeMethodWebHandler: /Resource/{EntityName}
handler6=Consultingwerk.Web2.WebHandler.HistoryMenuWebHandler: /SmartMenu/{MenuStructureId}
handler7=Consultingwerk.Web2.Services.SmartViewsHandler.SmartGridWebHandler: /SmartViews/Grid/{EntityName}
handler8=Consultingwerk.Web2.Services.SmartViewsHandler.SmartViewerWebHandler: /SmartViewer/Viewer/{EntityName}
```
WebHandler based Backend

- WebHandler URI mapping allows to setup same structure as Data Object Service with REST Adapter
- WebHandler allow easy mixing of JSDO Resource requests with other REST requests
- Programming model around WebHandlers provides data as JSON “Entity” (request body)
Dynamic JSDO Backend implementation

- Our WebHandler are providing an interface to the same Service.cls class that serves REST requests.
- Input/output to actual worker methods are JsonObject’s retrieved from or returned to WebRequest/WebRequest as the “Entity” (update) or Dataset-Handle (read).
/*
Purpose: Default handler for the HTTP GET method. The request being
serviced and an optional status code is returned. A zero or
null value means this method will deal with all errors.

Notes:
@param poRequest The IWebRequest instance representing the call
@return StatusCode of the response sent to the client
*/

METHOD OVERRIDE PROTECTED INTEGER HandleGet (poRequest AS IWebRequest):

DEFINE VARIABLE oResponse AS IHttpResponse NO-UNDO.
DEFINE VARIABLE oParamDictionary AS CharacterDictionary NO-UNDO.
DEFINE VARIABLE oService AS Service NO-UNDO.
DEFINE VARIABLE cEntityName AS CHARACTER NO-UNDO.
DEFINE VARIABLE cFilter AS CHARACTER NO-UNDO.
DEFINE VARIABLE cNumRecords AS CHARACTER NO-UNDO INIT ? .
DEFINE VARIABLE cStopAfter AS CHARACTER NO-UNDO INIT ? .
DEFINE VARIABLE hDataset AS HANDLE NO-UNDO.

ASSIGN
    oResponse = NEW WebResponse ()
    /* HTTP messages require a content type */
    oResponse:ContentType = 'application/json':U


/ *  
Purpose: Generic GetData (get/read) Service Interface  
Notes:  
@param pcEntityName The name of the Business Entity  
@param pcFilter The filter parameter  
@param pcNumRecords The numRecords value from the http client, see method ParseNumRecords  
@param pcStopAfter The stopAfter value from the http client, see method ParseStopAfter  
@param phDataset The Dataset to return to the client  
* /

@openapi.opendge.export(type="REST", useReturnValue="false", writeDataSetBeforeImage="true").

METHOD PUBLIC VOID GetData (pcEntityName AS CHARACTER,
                          pcFilter AS CHARACTER,
                          pcNumRecords AS CHARACTER,
                          pcStopAfter AS CHARACTER,
                          OUTPUT DATASET-HANDLE phDataset):

IF pcFilter BEGINS "~{":U THEN
    THIS-OBJECT:FetchDataRequestFromFilter (oFetchDataRequest, pcFilter, cBufferName, OUTPUT cOrderBy)
Purpose: Assigns the properties of the given FetchDataRequest instance from a character string representing an ABL Filter

Notes:
@param poFetchDataRequest The FetchDataRequest instance to assign values to
@param pcFilterName The character string representing the ABL Filter instance
@param pcBufferName The Buffer name for the query string
@param pcOrderBy OUTPUT The order by value

METHOD PROTECTED VOID FetchDataRequestFromFilter (poFetchDataRequest AS FetchDataRequest, 
cFilter AS CHARACTER, 
pcBufferName AS CHARACTER, 
OUTPUT pcOrderBy AS CHARACTER):

DEFINE VARIABLE oFilterParameter AS FilterParameter NO-UNDO.
DEFINE VARIABLE oQueryParser AS QueryParser NO-UNDO.
DEFINE VARIABLE lcFilter AS LONGCHAR NO-UNDO.
DEFINE VARIABLE i AS INTEGER NO-UNDO.
DEFINE VARIABLE cFilterTable AS CHARACTER NO-UNDO.
DEFINE VARIABLE oJsonObject AS JsonObject NO-UNDO.
DEFINE VARIABLE oObjectModel AS ObjectModelParser NO-UNDO.
DEFINE VARIABLE oFormat AS NumericFormat NO-UNDO.
DEFINE VARIABLE iEntry AS INTEGER NO-UNDO.
DEFINE VARIABLE cNames AS CHARACTER NO-UNDO.


/* SCL-415: Perform JSON Serialization/Deserialization with AMERICAN numeric format to avoid issues with decimal point interpretation */
oFormat = SessionHelper:GetNumericFormat() .
SessionHelper:SetDefaultNumericFormat().

lcFilter = pcFilter .

oObjectModel = NEW ObjectModelParser () .
oJsonObject = CAST (oObjectModel:Parse (lcFilter), JsonObject).

REST in peace
Demo

- Walk through Resource access WebHandler
Don’t miss our other presentations

- Monday 11:00: **CCS Deep Dive** (Mike)
- Tuesday 11:00: **OO-Oh** (Mike)
- Tuesday 13:00: **Application Modernization using the SmartComponent Library** (Mike and Marko)
- Tuesday 16:45: **REST in peace** (Mike)
- Wednesday 11:00: **CCS BoF** (all CCS)
- Wednesday 11:00: **Angular JS for OpenEdge programmers** (Marko)
Questions