Working with OpenEdge Data and Business Logic in a Kendo UI Builder Application

Anil Kumar Kotha
Senior Software Engineer
Nov 17 2017

Maura Regan
Principal Software Engineer
Nov 17 2017
Disclaimer
What are we going to discuss?

- KUIB Overview
- JSDO and Business Entity
- Extending Business Entities
- Working with relational and referential data
- Customizing Kendo UI Builder Web Apps
- OpenEdge Security and Kendo UI Builder
KUIB Overview

- Single Page Application (SPA) Dev Environment
- Drag and Drop facility
- Predefined templates
- Rich set of controls
- Electron Shell container
KUIB App flow

KUIB Designer or Web App
- Module
- Data Provider
- Template
- Customization

JSDO
- Bridge between OE and KUIB
- Communication

OE AppServer
- Business Logic
- Data Processing

OpenEdge Database
- Actual Data
- Triggers
- Roles
Different Components Involved – Runtime and Design

Components
- KUIB Designer
- JSDO
- PDS OE

Web Artifacts
- Node.js and npm
- AngularJS
- Bootstrap
- Webpack

Transports
- REST RPC
- Web DOH

Servers
- Progress Application Server (PAS)
- “Classic” OpenEdge AppServer

Database
- OpenEdge DB
Agenda

- KUIB Overview
- JSDO and Business Entity
- Extending Business Entities
- Working with relational and referential data
- Customizing Kendo UI Builder Web Apps
- OpenEdge Security and Kendo UI Builder
JSDO’s role in KUIB Web App
Business Entities

- CRUD Operations
- Significance of Submit
  - Transactional operation
- Abstract Business Entity
  - OpenEdge.BusinessLogic.BusinessEntity
  - Located in DLC/tty/OpenEdge.BusinessLogic.pl
Sample Catalog

```
"schema": {
  "type": "object",
  "additionalProperties": false,
  "properties": {
    "CustomerOrders": {
      "type": "object",
      "additionalProperties": false,
      "properties": {
        "ttCustomer": {
          "type": "array",
          "primaryKey": [...
        }
    }
  }
},
{"operations": [
  {"name": "GetttCust",
    "path": "/GetttCust",
    "useBeforeImage": false,
    "type": "invoke",
    "verb": "put",
    "params": [...] // 2 items
  },
  {"name": "SubmitCustomerOrders",
    "path": "/SubmitCustomerOrders",
    "useBeforeImage": true,
    "type": "submit",
    "verb": "put",
    "params": [...] // 2 items
  }
}
```


# JSDO APIs

<table>
<thead>
<tr>
<th>JSDO Method</th>
<th>Business Entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>add() – Create</td>
<td></td>
</tr>
<tr>
<td>assign() – Update</td>
<td></td>
</tr>
<tr>
<td>remove() – Delete</td>
<td></td>
</tr>
<tr>
<td>fill() – Read</td>
<td>READ</td>
</tr>
<tr>
<td>saveChanges(false)</td>
<td>CUD</td>
</tr>
<tr>
<td>saveChanges(true)</td>
<td>Submit</td>
</tr>
<tr>
<td>invoke(“myMethod”)</td>
<td>myMethod()</td>
</tr>
</tbody>
</table>
Error handling via JSDO

- Use `getErrors()` API
  - Allows us to access all AppServer errors seamlessly
    ```javascript
    jsdo = new progress.data.JSDO({ name: 'CustOrder' });
    ...
    jsdoErrors = jsdo.eCustomer.getErrors();
    ```
- Error handler can be overridden in KUIB
Agenda

▪ KUIB Overview
▪ JSDO and Business Entity
▪ Extending Business Entities
▪ Working with relational and referential data
▪ Customizing Kendo UI Builder Web Apps
▪ OpenEdge Security and Kendo UI Builder
Modernization
Working with Business Entities

- Start from scratch
- Leverage existing code
- Convert existing code
Define Service Interface wizard

- Annotations
  - File level
  - Method level (CRUD + INVOKE)
  - Field level
    - Semantic types
    - Foreign Key
    - Serialize-Name
Extending a Business Entity

- Change temp-table / dataset definition
- Customizing auto-generated CRUD+S operations code
  - Abstract Business Entity is optional
- Server side processing
- Annotations:
  - Mapping Types
  - Semantic Types
  - Foreign Key
  - Count
Server side processing

- Business Entity should be configured with JFP
  - ‘Count’ operation is optional in KUIB
  - Auto-filled if BE already has a count method
- Every request is processed in AppServer layer
- Filtering, Paging, Sorting are dependent

- Simple config in KUIB’s data source
Count Operation

- Count operation [Required for Server side processing]
  - Fetch number of records in OE database
  - Additional operation similar to INVOKE
- Count operation annotation
  - JSDO is aware of ‘count’ by default

```java
@openapi.openedge.export(type="REST", useReturnValue="false",
writeDataSetBeforeImage="false").
@progress.service.resourceMapping(type="REST", operation="count"
URI="/myCount?filter=~{filter~}", alias="", mediaType="application/json").
METHOD PUBLIC VOID myCount( INPUT filter AS CHARACTER, OUTPUT numRecs AS INTEGER):

"operations": [
    {
        "name": "myCount",
        "path": "\myCount?filter={filter}",
        "useBeforeImage": false,
        "type": "count",
        "verb": "put",
        "params": []
    }
],
```
JFP – JSON Filter Pattern

- Allows data processing at server side
- Additional annotations to ‘Read’ method
- Default Kendo UI DataSource processing is at client side
- Accessed via JSDO’s mapping type - JFP

```java
@openapi.openedge.export(type="REST", useReturnValue="false", writeDataSetBeforeImage="true").
@progress.service.resourceMapping(type="REST", operation="read", URI="?filter=~{filter~}",
  alias="", mediaType="application/json").
@openapi.openedge.method.property(name="mappingType", value="JFP").
@openapi.openedge.method.property(name="capabilities", value="ab1Filter,top,skip,id,orderBy").
METHOD PUBLIC VOID ReadCustomer(INPUT filter AS CHARACTER, OUTPUT DATASET dsCustomer):
Mappings

- Request mapping

```javascript
function registerPlugin() {
    var jfpPlugin = progress.data.PluginManager.getPlugin("JFP");
    progress.data.PluginManager.addPlugin("MYJFP", {
        requestMapping: function(jsdo, params, info) {
            var requestParams = {};

            object = {};
            params = jfpPlugin.requestMapping(jsdo, params, info);
            if (params && typeof params.filter === "string") {
                object = JSON.parse(params.filter);
            }
            object.mydata = jsdo.getProperty("mydata");
            requestParams.filter = JSON.stringify(object);
            return requestParams;
        }
    });
}
```

- Response mapping

```javascript
progress.data.PluginManager.addPlugin("myResponsePlugin", {
    responseMapping: function(jsdo, response, info) {
        var record;
        var newData = response.dsEmployee.employees;

        if (info.operation === "read") {
            for (var i = 0; i < newData.length; i++) {
                record = newData[i];
                record.VacDays = record.VacDays + 10;
            }
        }
        jsdo.setProperty("server.count", response.myTotal);
    }
// You must return the response
    return response;
});
```
Agenda

- KUIB Overview
- JSDO and Business Entity
- Extending Business Entities
- Working with relational and referential data
- Customizing Kendo UI Builder Web Apps
- OpenEdge Security and Kendo UI Builder
Foreign Key Support

- Placeholder field
- Semantic Type is ‘Lookup’
- Editor Types
  - Combo-box
  - Drop-down list
- Business Logic should be annotated as below:
  - 11.7.1 PDS OE supports tooling

```sql
@openapi.opendge.entity.foreignkey (name="ttSalesrepFK", fields="SalesRepID", parent="Salesrep.ttSalesrep", parentFields="SalesRep")
DEFINE TEMP-TABLE ttOrder NO-UNDO BEFORE-TABLE bttOrder
FIELD Ordernum AS INTEGER INITIAL 0 LABEL "Order Num"
FIELD SalesrepID AS CHARACTER LABEL "Sales Rep"
```
Hierarchical and Stacked Data Grids

- Parent/Child data represented in different forms in KUIB webapp
- Supported editing modes:
  - Inline
  - Popup
  - Incell
- Allows CRUD operations on child table data
- Both parent and child tables should be in single resource
Agenda

- KUIB Overview
- JSDO and Business Entity
- Extending Business Entities
- Working with relational and referential data
- Customizing Kendo UI Builder Web Apps
- OpenEdge Security and Kendo UI Builder
KUIB and JSDO code under the hood

- **JSDO Catalog**
  - Resources (DataSets and Temp-tables)
  - Operations
- **Data Source definitions**
- **Arrays representation**
- **Metadata**
- **Generated code uses:**
  - JSDO Dialect for Kendo UI DataSource
Customizing KUIB Code

- Custom Sections
- View Factory
- Public Controller
- Other Assets
- Customizing Templates
  - Create your own templates
  - Extend an existing template
Recommendations

- Encapsulate functionality into high level API methods
  - AngularJS code
  - Kendo UI components
  - Kendo UI DataSource
Agenda

- KUIB Overview
- JSDO and Business Entity
- Extending Business Entities
- Working with relational and referential data
- Customizing Kendo UI Builder Web Apps
- OpenEdge Security and Kendo UI Builder
Authentication and Authorization

- Supported models
  - Anonymous
  - Basic
  - FORM
  - SSO (in pipeline)

- User Roles

- JSDO Specific:
  - Use `progress.data.JSDOSession`
  - `progress.data.Session` (deprecating in future)
Enabling Authentication at PASOE layer

- Modifications to `oeablSecurity.properties` file [new in OpenEdge 11.7]
  - Located in `{DLCWork}/<oepas_instance>/webapps/<webapp>/WEB-INF`

- Change `client.login.model`

- User’s information
  - `users.properties`
  - LDAP
  - OERealm

```
### The HTTP client Authentication model to use ###

#### This property controls which HTTP client authentication model to use. The allowed names are:

<table>
<thead>
<tr>
<th>name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>anonymous</td>
<td>No user login - all clients have public access</td>
</tr>
<tr>
<td>basic</td>
<td>Users authenticate using the HTTP BASIC standard</td>
</tr>
<tr>
<td>form</td>
<td>Users authenticate using a HTTP POST message &amp; form data</td>
</tr>
<tr>
<td>container</td>
<td>Users authenticate via Tomcat realm services and authorize URL access via Spring Security</td>
</tr>
<tr>
<td>OERealm</td>
<td>OpenEdge Single Sign-on using ClientPrincipal access tokens</td>
</tr>
</tbody>
</table>

```

```plain
client.login.model=form
```

```
### HTTP BASIC Realm name for All Transports ###

#### Set the BASIC realm name used by browsers to prompt the user for a user-id/password.

```
http.all.realm=OpenEdge
```
Authentication – KUIB

- At Data Provider (resource) level
  - All Data sources (tables) will use same authentication
- Login screen is shown upon Preview
  - First module’s resource loaded upon successful login
Summary

- Seamless integration with JSDO
- Leverage your Business Logic
- Flexibility with views in KUIB
  - Pre-defined views
  - User-defined (Blank) view
- Access to large set of Kendo UI components

KUIB Examples

- oemobiledemo.progress.com
- GitHub - https://github.com/CloudDataObject/kendo-ui-builder-samples
- Progress Communities KUIB wiki page
Happy Developing !!!

KUIB is the ‘Key’ to Modernization