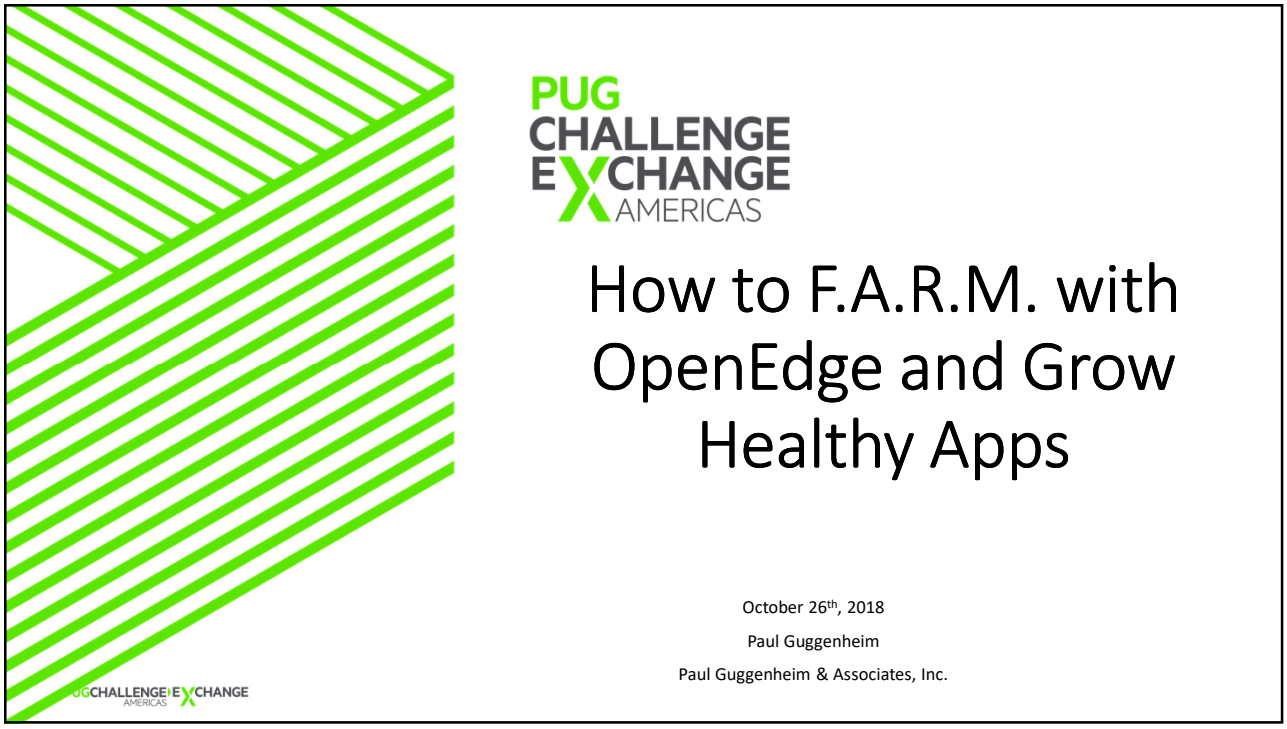


About PGA

- A Progress Evangelist since 1984, and enlightening Progress programmers since 1986
- Designed several comprehensive Progress courses covering all levels of expertise including - The Keys to OpenEdge®
- **White Star** Software Strategic Partner
- **Consultingwerk** Partner
- **TailorPro** Board Member
- **AppPro** Reseller
- Major consulting clients include Carrier Logistics, Chicago Metal Rolled Products, Eastern Municipal Water District, Foxwoods Casino, Hendrickson Trailer, Interlocal Pension Fund, International Financial Data Services, National Safety Council, and Stanley Engineering.
- Head of the Chicago Area Progress Users Group
- PUG Challenge Steering Committee Member

Presentation Objectives

- Define Terms such as node.js, npm, and FARM.
- Become acquainted with how JavaScript DataObject (JSDO) works
- Illustrate the steps needed to develop a simple JSDO FARM program for the web
- Examine the connectivity between the OpenEdge database and Progress Appserver for OpenEdge (PASOE) on the backend, to the web or mobile front end using JSDO.
- Demonstrate JSDO FARM operations using an Angular front end.

PUGCHALLENGE EXCHANGE AMERICAS

What is node.js?

"Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside of a browser."

Wikipedia

What is NPM?

"npm is a package manager for the JavaScript programming language. It is the default package manager for the JavaScript runtime environment Node.js."

Wikipedia

PUGCHALLENGE EXCHANGE AMERICAS

What is FARM?

- FARM stands for:
  - Find
  - Add
  - Remove
  - Modify
- Friendlier acronym than CRUD

What is a JavaScript Data Object?

- The purpose of the JSDO is to simplify access to relational data in a mobile application.
- JSDO is an open source application by PSC
  https://github.com/progress/JSDO
- How does it do it?
  - JSDO executes FARM operations through the DataSource Object.
  - JSDO uses an internal data store (JSDO memory) to cache data
  - JSDO handles complex data such as ProDataSets
  - Keeps a before image of the data so that it can perform rollbacks
  - It uses the Data Service Catalog to define the schema definitions

What is a DataSource Object?

"The Progress Data Source is an npm module that provides seamless integration between client apps built with the Angular framework (that is, NativeScript and Angular web apps) and a Progress Data Object Service."

The Data Source class "provides the integration layer to access the remote data resource. (Internally, it supports access to the JSDO.)"

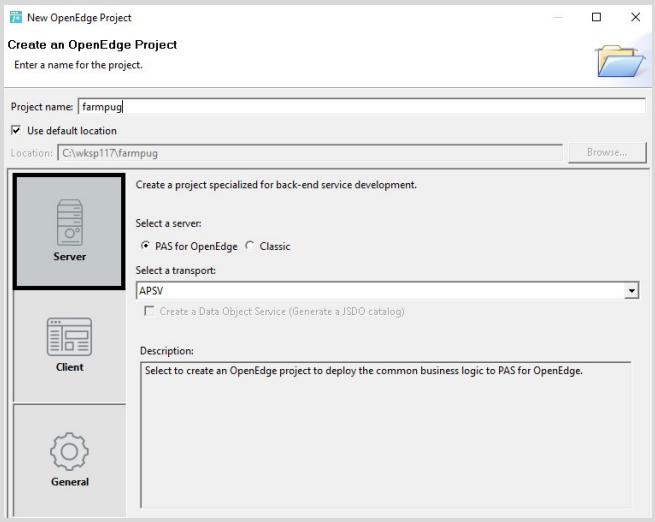
Progress Data Objects Guide and Reference

Development Steps Agenda

- Setting up the Backend
  - Create Business Entity
  - Create JSDO Service
  - Assign to a PASOE
  - Test with Catalog and Query
- Setting up the Frontend
  - Install Angular and Create Angular Project
  - Create Angular Instance
- FARM Demonstration
  - Show FARM Functions
  - Show FARM JSDO calls from Angular

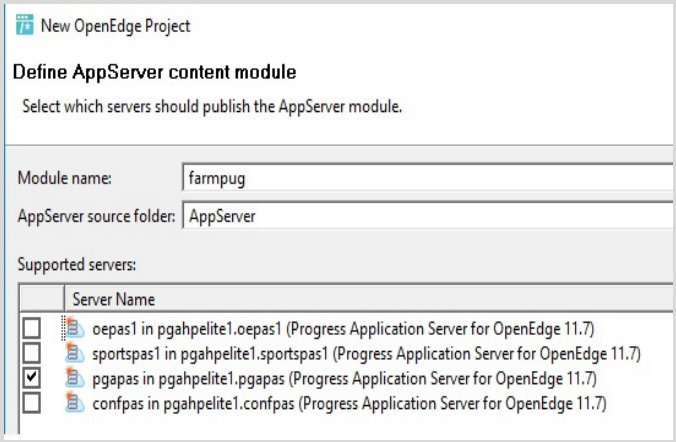
Backend Setup

Create OpenEdge server project



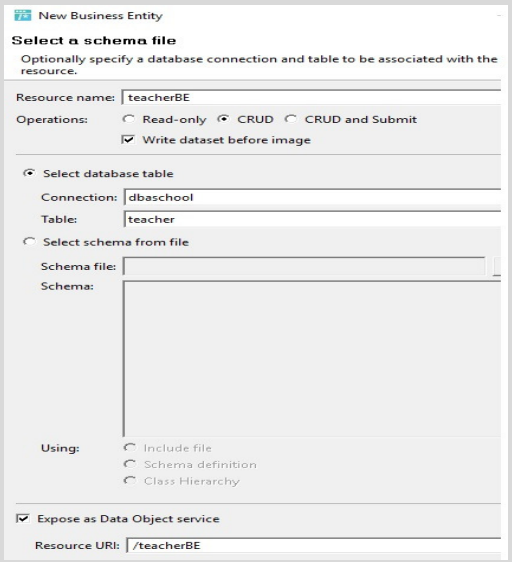
Backend Setup

Assign desired PASOE to project



Backend Setup

Create a business entity for the desired table



PUGCHALLENGE EXCHANGE AMERICAS

Backend Setup

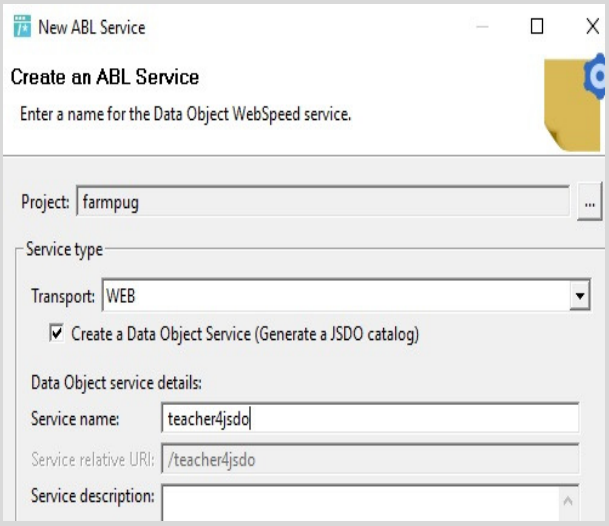
Exclude Custnum field during add record with SkipListArray

```
CONSTRUCTOR PUBLIC teacherBE():

      DEFINE VARIABLE hDataSourceArray AS HANDLE NO-UNDO
                                EXTENT 1.
        DEFINE VARIABLE cSkipListArray AS CHARACTER NO-UNDO
                                EXTENT 1.
.
.
.
 /*  Each skip-list entry is a comma-separated list of field
     names, to be ignored in create stmt */

       cSkipListArray[1] = "teacherId".


       THIS-OBJECT:ProDataSource = hDataSourceArray.
        THIS-OBJECT:SkipList = cSkipListArray.

   END CONSTRUCTOR.
```

PUGCHALLENGE EXCHANGE AMERICAS

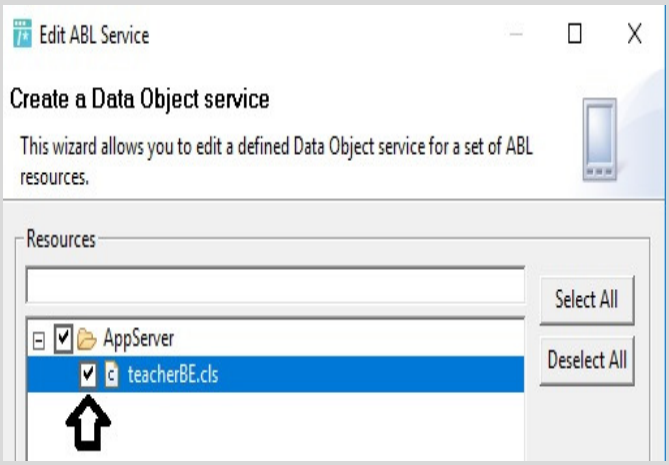
Backend Setup

Create an ABL Service



Backend Setup

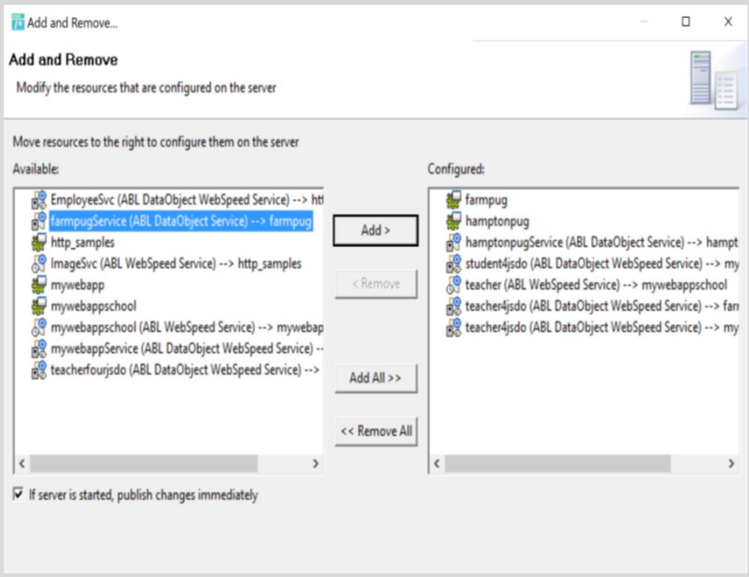
Select the created business entity to assign to this Data Object service



PUGCHALLENGE EXCHANGE AMERICAS

Backend Setup

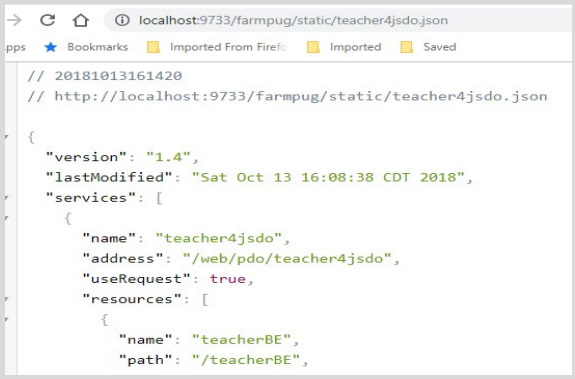
Add Data Object service to desired PASOE



Test Backend Setup

Use the following URL to access the JSDO Catalog:

http://localhost:<PASOEPort>/<projectName>/static/<serviceName>.json



- Please note the case of the fields presented in the catalog

Test Backend Setup – Extract Data

- Create a test query using the JSDO to display data in a browser

- There are 4 Test Query required components
  1. Import jQuery and JSDO libraries
  2. Set serviceURI and catalogURI variables
  3. Create function to execute after data is filled in the JSDO
  4. Execute the getSession method and pass the appropriate parameters

Import jQuery and JSDO libraries

```
<script src="https://kendo.cdn.telerik.com/2018.2.620/js/jquery.min.js"></script>

<script src="https://oemobiledemo.progress.com/jsdo/progress.jsdo.js"></script>
```

Set serviceURI and catalogURI variables

```
var serviceURI = "http://localhost:9733/farmpug";

var catalogURI = "http://localhost:9733/farmpug/static/teacher4jsdo.json";
```

PUGCHALLENGE EXCHANGE AMERICAS

Create after JSDO data filled function

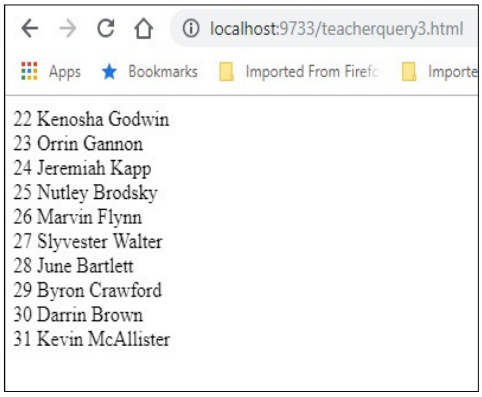
```
function onAfterFill(jsdo, success, request) {
            jsdo.ttteacher.foreach(function(teacher) {
                  document.write(teacher.data.teacherId
                  + ' `
                  + teacher.data.tfirstName
                  + ' `
                  + teacher.data.tlastName + '<br>');
            });
      }
```

Execute the getSession method

```
progress.data.getSession({
        serviceURI: serviceURI,
        catalogURI: catalogURI,
        authenticationModel: "anonymous"
}).done(function (jsdosession) {
        jsdo = new progress.data.JSDO({ name: 'teacherBE' });
        jsdo.fill("teacherid > 21 and teacherid < 32")
                .done(onAfterFill)
                .fail(function () {
                        console.log("Error while reading records.");
                });
}).fail(function () {
        console.log("Error while creating session.");
});
```

Run Command to Check Data Output

http://localhost:9733/teacherquery3.html



Setting up the Frontend

Please see the wiki page for additional installation instructions:

https://github.com/progress/JSDO/wiki/Using-the-JSDO-and-DataSource-components-in-an-Angular-web-app

Here are the steps:

- If not done previously, install node.js from https://nodejs.org/en/.
- Install the TLS version which also installs npm
- npm -g install @angular/cli@1.6.5 (Installs Angular on the machine)
- ng new <yourProjectName> (Creates Angular Project)
- cd <yourProjectName> (go into the project name directory)
- npm install @progress/jsdo-angular (Installs JSDO Angular module)
- Install Visual Studio Code using the link: https://code.visualstudio.com

PUGCHALLENGE AMERICAS EXCHANGE

Setting up the Frontend

Once Visual Studio Code is installed, do the following:

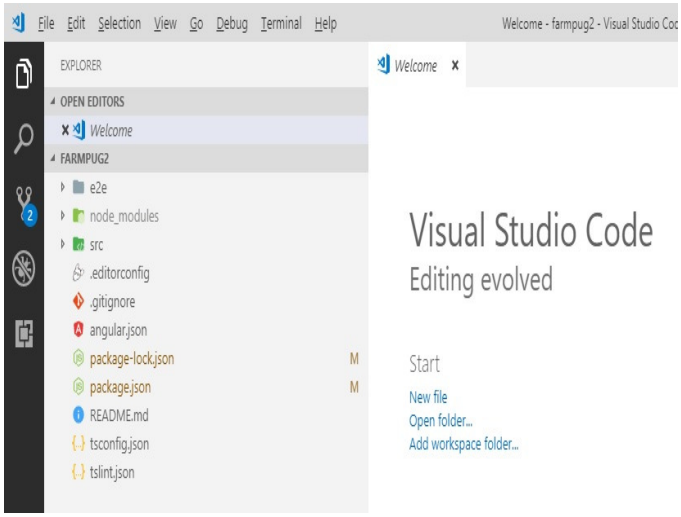
- Go into the project name directory

(if not there already)

- Type: `code .`

(Where "." specifies the current directory)

The Visual Studio Code app will appear.



PUGCHALLENGE EXCHANGE AMERICAS

Setting up the Frontend

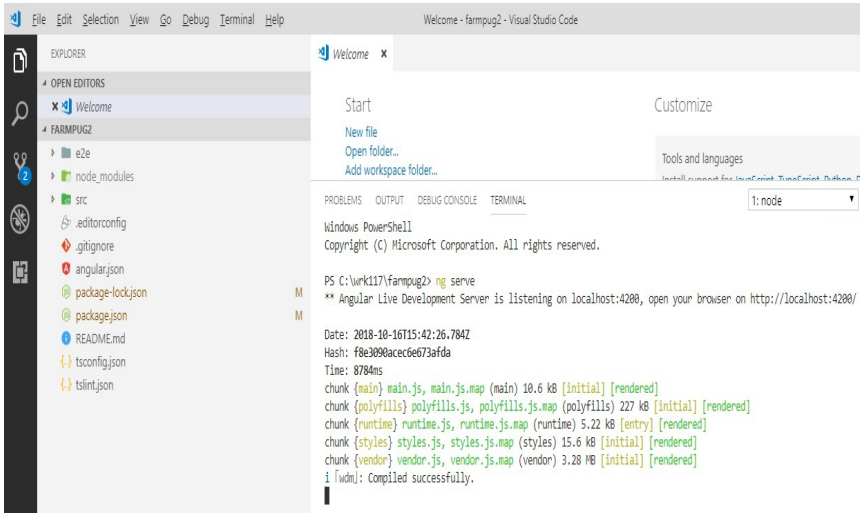
Once Visual Studio Code is open, create a new terminal window by typing the shortcut:

`CTRL-Shift-`` `

Then type:

```
ng serve
```

To start the Angular Web Server



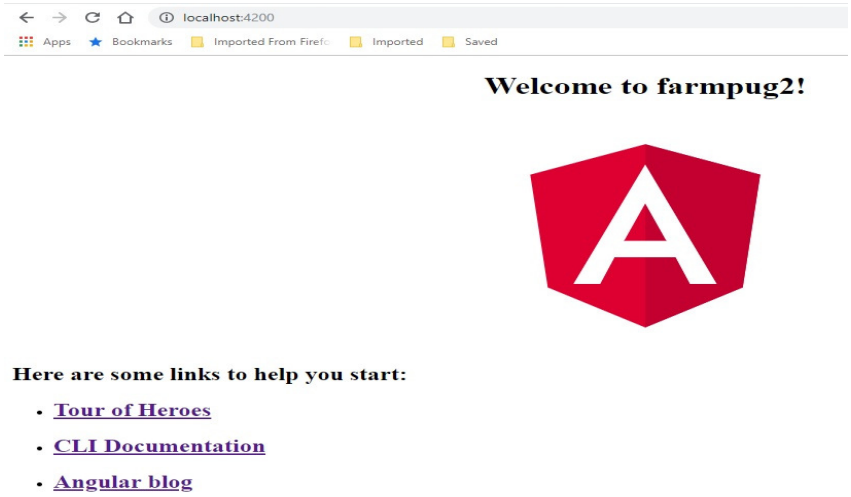
PUGCHALLENGE EXCHANGE AMERICAS

Setting up the Frontend

Access the Angular app by typing:

http://localhost:4200

We will be replacing this HTML and Javascript code with our jsdo code.



Setting up the Frontend

From the github link where it is titled:

Using the JSDO and DataSource components in an Angular web app

Paste the code to the right:

Code for app.component.css

```
li:focus {
    background: lightgreen;
}

.active {
    background-color: lightgreen;
}
```

PUGCHALLENGE EXCHANGE AMERICAS

Setting up the Frontend

From the github link where it is titled:

Using the JSDO and DataSource components in an Angular web app

Paste the code to the right for the app.component.html file:

(full code is in the link)

```
<h2>{{title}}</h2>

<div>
    <div>
        <button (click)="newItem()">New</button>
        <button (click)="saveItem()">Save</button>
        <button (click)="deleteItem()">Delete</button>
    </div>
    <div style="background-color: lightcoral;">{{appMessage}}</div>
    <div style="width: 300px; float: left;">
        <ul style="list-style-type:none;">
            <li *ngFor="let item of items; let i = index"
                [class.active]="isSelected(i)"
                [attr.data-index]="i"
                tabindex="1" (click)="selectItem(i)">{{item.Name}}</li>
        </ul>
    </div>
    <div style="margin-left: 300px;">
        <div style="width: 50%; padding: 10px;">
            <label>CustNum</label>
            <input [value]="customer.CustNum" disabled="false" />
        </div>
.
.
.
```

PUGCHALLENGE EXCHANGE AMERICAS

import { Component } from '@angular/core'; import { FormsModule } from '@angular/forms'; import { progress } from '@progress/jsdo-core'; import { DataResult, DataSource, DataSourceOptions } from '@progress/jsdo-angular'; const serviceURI = 'https://oemo

Setting up the Frontend

From the github link where it is titled:

Using the JSDO and DataSource components in an Angular web app

Paste the code to the right for the app.component.ts file:

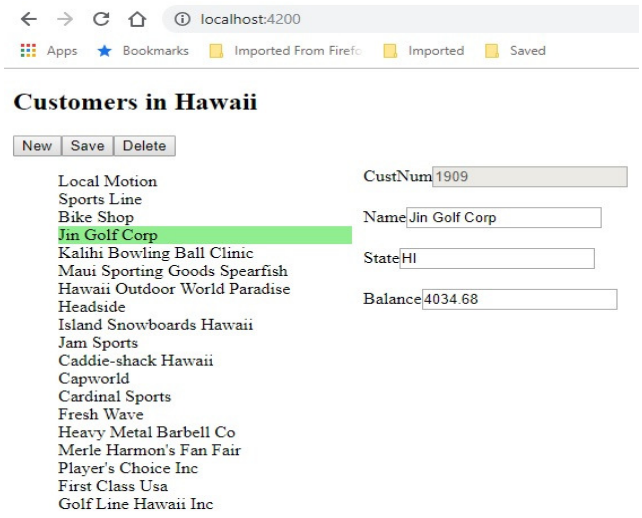
(full code is in the link)

```
import { Component } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { progress } from '@progress/jsdo-core';
import { DataResult,
         DataSource,
         DataSourceOptions } from '@progress/jsdo-angular';

const serviceURI = 'https://oemobiledemo.progress.com/OEMobileDemoServices';
const catalogURI =
'https://oemobiledemo.progress.com/OEMobileDemoServices/static/SportsService.j
son';

const CUSTOMER_TEMPLATE = {
       CustNum: '',
       Name: '(new customer)',
       State: 'HI',
       Balance: 0 };
.
.
.
```

PUGCHALLENGE EXCHANGE AMERICAS

import { Component } from '@angular/core'; import { FormsModule } from '@angular/forms'; import { progress } from '@progress/jsdo-core'; import { DataResult, DataSource, DataSourceOptions } from '@progress/jsdo-angular'; const serviceURI = 'https://oemo

Setting up the Frontend



PUGCHALLENGE EXCHANGE AMERICAS

getSession Method

- The getSession method returns a progress.data.JSDOSession instance with a specified JSDO login session already established and a specified Data Service Catalog already loaded.
- It requires 3 parameters:
  - ServiceURI
  - CatalogURI
  - AuthenticationModel

getSession Method

- If the getSession method is successful, then a DataSource object can be instantiated.
- The DataSource object is bound to a jsdo object.
- A jsdo object contains the name of the business entity.
  - Make sure the spelling and the case of the letters of the business entity match what's in the JSDO Catalog!

DataSource FARM Operations

- The DataSource object contains the following functions:
  - create
  - update
  - remove (delete)
  - saveChanges
- The create, update, and remove methods change data in local jsdo memory. Think of this like a temp-table in the ABL.
- The saveChanges function sends those changes back to the server where the records can be updated and committed. This is similar to the SAVE-ROW-CHANGES() method on dataset buffers.

PUGCHALLENGE EXCHANGE AMERICAS

DataSource FARM Operations

- The saveItem function checks to see if the customer already exists in JSDO memory.
  - If it does, then there exists an _id value and the update function is called.
  - If there isn't an _id value, then the create function is called.
  - In either of these cases, the saveChanges must be executed in order to update the records on the server.
- For the deleteItem function, the dataSource remove function is called, which removes the record from local JSDO memory.
  - The saveChanges function is then run to remove the record on the server.

DataSource FARM Operations

- The readCustomer function performs the DataSource read function.
- This consists of a filter where the state field equals HI for Hawaii.
- Another way to specify the filter is:
  - filter: “state eq ‘HI’”

JSDO Properties and Methods

- The JSDO is very similar to a ProDataSet.
- Here are some of the properties and methods that are available.

Property	Method
autoSort	acceptChanges acceptRowChanges
caseSensitive	addRecords
name	Find, foreach, getData, getId, getSchema
Record	rejectChanges rejectRowChanges
useRelationships	saveChanges
autoApplyChanges	Subscribe, unsubscribe

PUGCHALLENGE EXCHANGE AMERICAS

Summary

- Setup the backend first since that creates a json catalog file which is used by the frontend JSDO logic.
- The JSDO incorporates ProDataSet functionality on the Web and Mobile FrontEnd allowing developers to build robust applications.
- FARM sounds better than CRUD.
- A very special thanks to Edsel Garcia for all his help and encouragement on making this presentation possible.
- Thanks to Will Griesemer at App Technologies for help on Javascript.

Summary – Helpful Links

- JSDO - https://github.com/progress/JSDO/wiki
- Demo – https://oemobiledemo.progress.com/jsdo/example001.html
- Node.js - https://nodejs.org/en/
- Visual Studio Code – https://code.visualstudio.com
- Angular Get Started - https://angular.io/guide/quickstart
- NativeScript Install Guide https://docs.nativescript.org/start/quick-setup
- Progress Angular-JSDO Get Started:

https://github.com/progress/JSDO/wiki/Using-the-JSDO-component-with-plain-HTML-and-JavaScript

PUG CHALLENGE EXCHANGE AMERICAS

Questions?

PUGCHALLENGE EXCHANGE AMERICAS