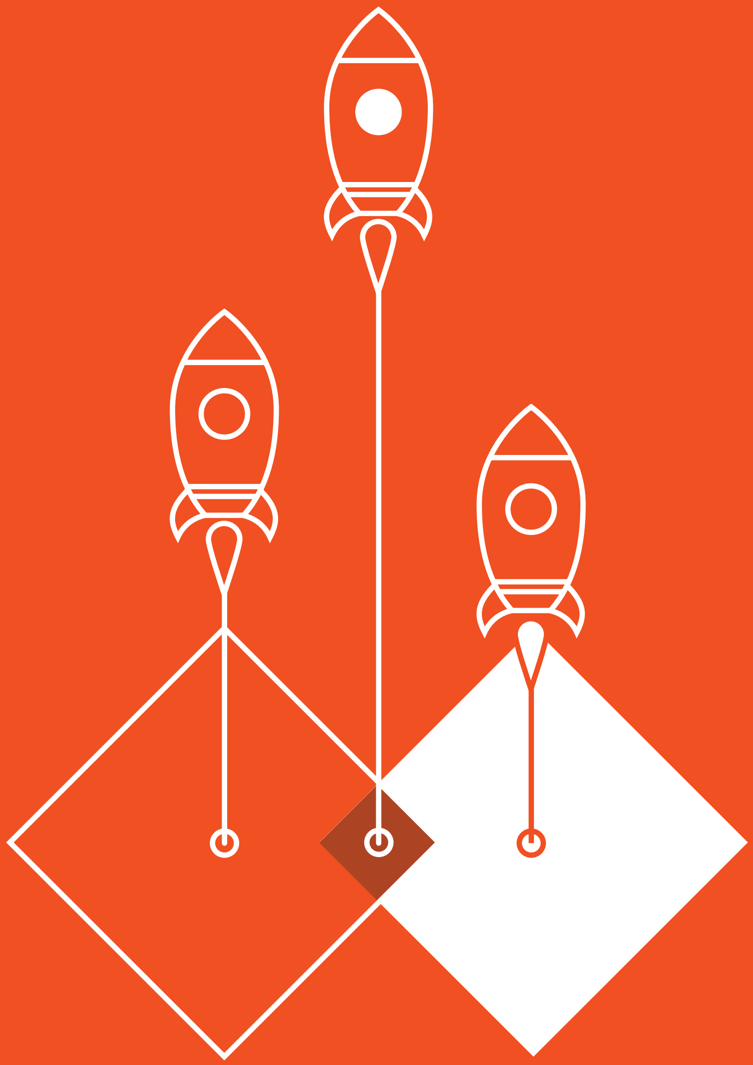
PROGRESS



USING THE JSDO WITH KENDO UI

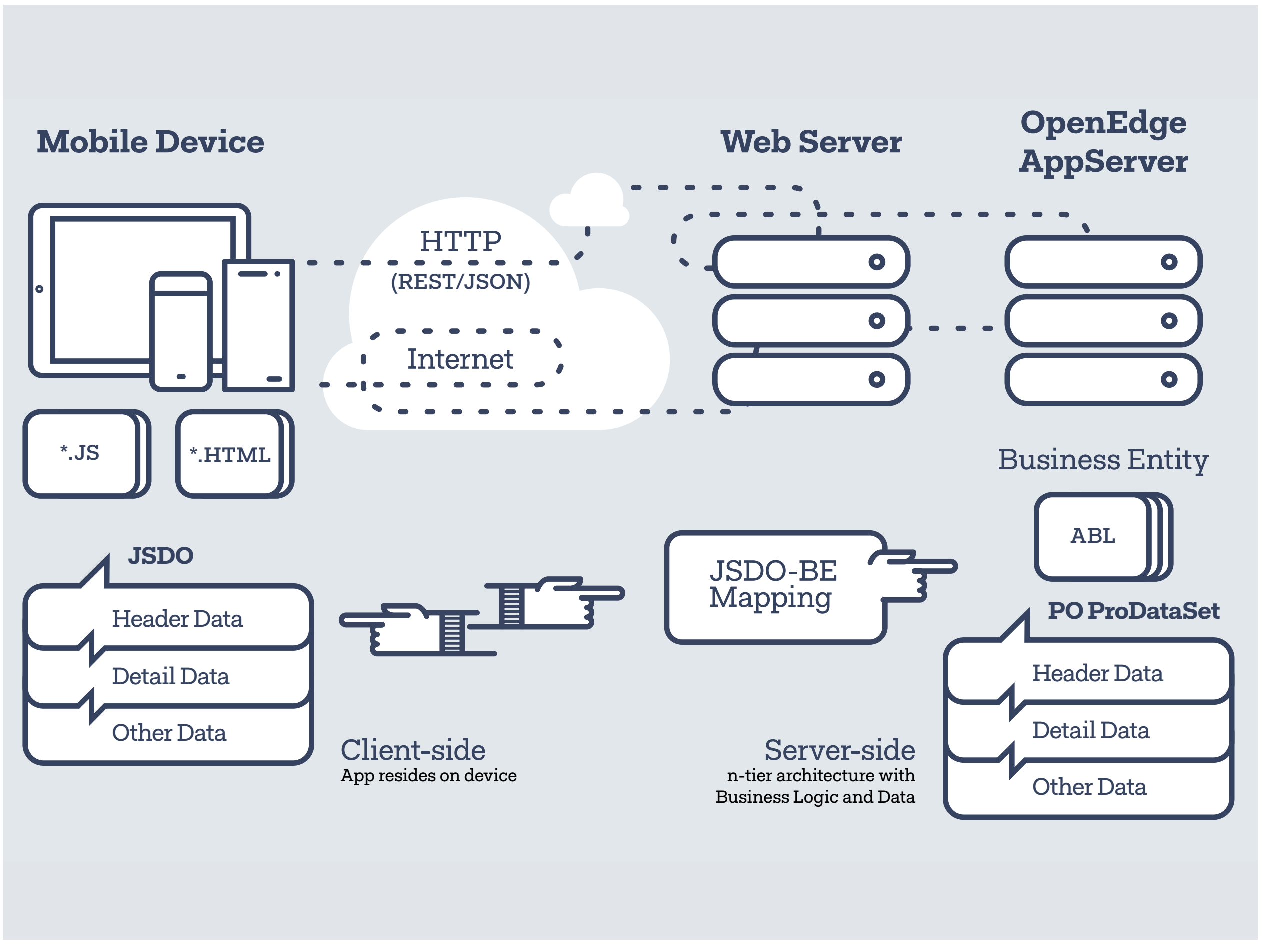
1: Introduction

Kendo UI is a JavaScript framework that can be used to build web and mobile apps using HTML5 and JavaScript. It provides a very powerful set of UI widgets that can be bound to a data source.

The Progress JavaScript Data Object (JSDO) provides support for a complex data model and API to manipulate that data while maintaining data integrity. The JSDO catalog defines the logical schema and mapping to a remote data source. The catalog also defines an API to invoke remote business logic in addition to simple CRUD operations. The JSDO is designed to work with any Web / JavaScript framework. Similarly Kendo UI is designed to provide the best UI regardless of backend provider. Naturally the two work great together. This whitepaper describes how to use the new JSDO dialect of the Kendo UI DataSource with Kendo UI to access data and business logic on an OpenEdge AppServer.

Kendo UI widgets use the Kendo UI DataSource to access local and remote data. The Kendo UI DataSource is an abstraction on top of both local data (arrays) and remote data (HTTP endpoints) that makes it much easier to request data, fill widgets with data, and then make and track changes to that data. The **transport** property in the Kendo UI DataSource configures how to perform the CRUD operations for the DataSource by defining the corresponding properties: **create**, **read**, **update**, and **destroy**. The JSDO dialect of the Kendo UI DataSoure provides the definition for these properties.

The OpenEdge backend can be accessed through the Kendo UI DataSource using the same architecture used for Mobile in OpenEdge 11.2 and greater, the JSDO. Support for serverPaging, serverFiltering, serverSorting and batching in the Kendo UI DataSource requires OpenEdge 11.4 or greater.



The JSDO manages the complex communication with an OpenEdge AppServer. It connects to the OpenEdge AppServer and executes the ABL code which accesses the data and executes the business logic. This ABL program is called a Business Entity. The JSDO only depends on JavaScript and can be integrated with many JavaScript frameworks with support for HTTP communication.

To access the OpenEdge AppServer, the CRUD operations for the Kendo UI DataSource can be configured to call the corresponding CRUD operations in the JSDO.

This document provides examples and explains how the JSDO is used with the Kendo UI components.

2: Using the JSDO from a simple HTML page

The following example shows how to access the OpenEdge backend with a simple HTML page using the JSDO:

```
<!DOCTYPE html>
<html>
<head>
    <title>Simple JSDO Usage</title>
    <script src="http://code.jquery.com/jquery-1.11.3.min.js"></script>
    <script src="http://oemobiledemo.progress.com/jsdo/progress.jsdo.min.js"></script>
</head>
<body>
    <!-- results will be written here by JavaScript -->
    <script>
          (function () {
            // this function is called after data is returned from the server
            function onAfterFillCustomers(jsdo, success, request) {
                // for each customer record returned
                jsdo.eCustomer.foreach(function (customer) {
                    // write out some of the customer data to the page
                    document.write(customer.data.CustNum + ' ' + customer.data.Name + '<br>');
                });
            }
            try {
                var serviceURI = "http://oemobiledemo.progress.com/MobilityDemoService",
                    catalogURI = serviceURI + "/static/mobile/MobilityDemoService.json",
                    jsdosession, jsdo;
                // create a new session object
                jsdosession = new progress.data.JSDOSession({
                    serviceURI: serviceURI,
                    catalogURIs: catalogURI
                });
                jsdosession.login("", "").done(function(jsdosession, result, info) {
                    // load the catalog for the session
                    jsdosession.addCatalog(catalogURI)
                        .done(function(jsdosession, result, details) {
                            // create a JSDO
                            jsdo = new progress.data.JSDO({ name: 'dsCustomer' });
                            // calling fill reads from the remote OE server
                            jsdo.fill().done(onAfterFillCustomers);
                        });
                });
            }
            catch (e) {
                alert("Error instantiating objects: " + e);
            }
        }());
    </script>
</body>
</html>
```

View Example

The JavaScript file **progress.all.4.0.min.js** is included at the top of the page, just under the title. This file contains the code for the JSDOSession and JSDO objects.

Support for the JSDO is provided by two main objects:

1. The **JSDOSession** - manages authentication and session operations
2. The **JSDO** - provides the access to the OpenEdge backend by using the information in the JSDO Catalog

Once inside the JavaScript, the `serviceURI` and `catalogURI` variables are declared. These settings are used by the JSDO to access the OpenEdge service. The `serviceURI` is the URL to the web application of the Mobile Service. The `catalogURI` is the URL to the JSDO Catalog file, a JSON file containing the definition for the resources available in the OpenEdge Service. It specifies the schema and operations for each of the resources.

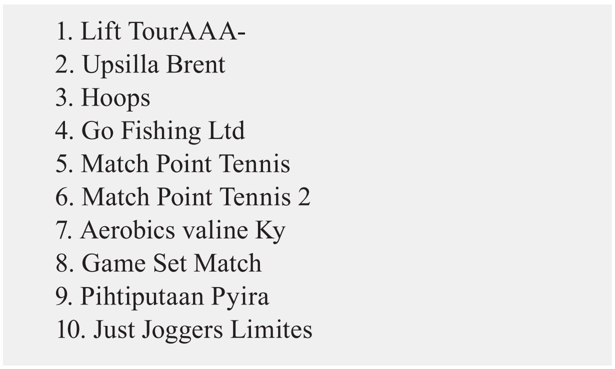
Developers do not need to specify any other URIs to access the data. For example, the developer can just call `fill()` to read the data from the backend and does not need to specify the URI that corresponds to the READ operation. This information is resolved internally based on the information in the catalog.

Operations to the server such as `fill()` (READ operation) and `saveChanges()` (CREATE, UPDATE, and DELETE operations) are asynchronous and need a callback to handle the response. The JSDO supports promises and a subscribe/unsubscribe approach to handle the response.

This example uses the the promises approach by calling the `.done()` function of the promise returned by the `fill()` method to handle the repose `.done()` function of the promise returned by the `fill()` to specify the `onAfterFillCustomers` function as the callback. The `.fail()` function of the promise is used to specify a callback to handle errors.

The `onAfterFillCustomers` function is called when the OpenEdge Service returns the data to the JSDO. In that function, the `foreach()` method is called to enumerate over the results and write them out to the page. The JSDO contains several other methods for handling data. Some of these methods are `getData()`, `find()`, `findById()`, `sort()`, and `addRecords()`.

Figure 2: Web Browser output shown records retrieved using the JSDO:



```
1. Lift TourAAA-
2. Upsilla Brent
3. Hoops
4. Go Fishing Ltd
5. Match Point Tennis
6. Match Point Tennis 2
7. Aerobics valine Ky
8. Game Set Match
9. Pihtiputaan Pyira
10. Just Joggers Limites
```

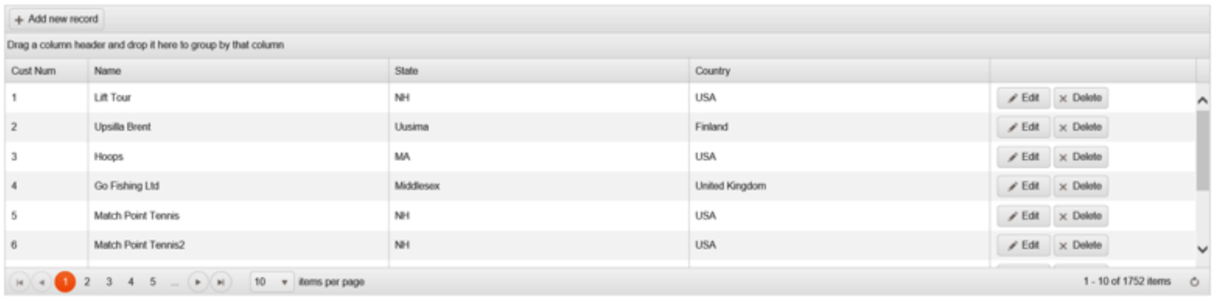
Update the `serviceURI` and the `catalogURI` to point to your own Mobile Service. The Online Documentation contains information on how to create a Mobile Service in Progress Developer Studio.

3: Using the JSDO from the Kendo UI Grid Component

The following example shows the JSDO working with the Kendo UI Grid. The Kendo UI Grid / Basic example from the Kendo UI demo page was used as a starting point. In addition to CRUD operations, right out of the box the Kendo UI Grid offers features such as paging, skipping records, column docking, adapting rendering and adjusting to screen size, and easy theming.

Using this example, we'll produce a visually appealing grid with basic functionality looking like this:

Figure 3: The Kendo UI Grid component showing records from an OpenEdge database



+ Add new record

Drag a column header and drop it here to group by that column

Cust Num	Name	State	Country	
1	Lift Tour	NH	USA	Edit × Delete
2	Upsilla Brent	Uusima	Finland	Edit × Delete
3	Hoops	MA	USA	Edit × Delete
4	Go Fishing Ltd	Middlesex	United Kingdom	Edit × Delete
5	Match Point Tennis	NH	USA	Edit × Delete
6	Match Point Tennis2	NH	USA	Edit × Delete

1 2 3 4 5 ... 10 items per page 1 - 10 of 1752 items

Connecting the Kendo UI Grid to the OpenEdge Service requires wiring up Kendo UI Transports to the JSDO.

```
<!DOCTYPE html>
<html>
<head>
    <title>JSDO / Kendo UI Grid Example</title>
    <link rel="stylesheet" href="http://cdn.kendostatic.com/2015.1.429/styles/kendo.common.min.css" />
    <link rel="stylesheet" href="http://cdn.kendostatic.com/2015.1.429/styles/kendo.default.min.css" />
    <script src="http://cdn.kendostatic.com/2015.1.429/js/jquery.min.js"></script>
    <script src="http://cdn.kendostatic.com/2015.1.429/js/kendo.all.min.js"></script>
    <script src="http://oemobiledemo.progress.com/jsdo/progress.all.min.js"></script>
    <style>
        html {
            font-size: 12px;
            font-family: Arial, Helvetica, sans-serif;
        }
    </style>
</head>
<body>
    <div id="example">
        <div id="grid"></div>
    </div>
    <script>
    $(function() {
        function createGrid() {
            $('#grid').kendoGrid({
                // define transports as the class functions
                dataSource: {
                    type: "jsdo",
                    transport: {
                        jsdo: "Customer"
                    },
                    error: function(e) {
                    }
                },
                height: 400,
                groupable: true,
                reorderable: true,
                resizable: true,
                sortable: true,
                pageable: {
                    refresh: true,
                    pageSizes: true,
                     pageSize: 10,
                    buttonCount: 5
                    },
                editable: 'inline',
                toolbar: ['create'],
                columns: [
                    { field: 'CustNum', title: 'Cust Num', width: 1000 },
                    { field: 'Name' },
                    { field: 'State' },
                    { field: 'Country' },
                    { command: ['edit', 'destroy'], title: '&nbsp;', width: '250px' }
                ]
            });
        }
        try {
            var serviceURI = "http://oemobiledemo.progress.com/CustomerService",
                jsdoSettings = {
                    serviceURI: serviceURI,
                    catalogURIs: serviceURI + "/static/mobile/CustomerService.json"
                },
                jsdosession,
                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){
                        createGrid();
                    })
                    .fail(function(jsdosession, result, details){
                        alert("Error while executing addCatalog().");
                });
            });
            promise.fail(function(jsdosession, result, info){
                alert("Error while executing login().");
            });
        }
        catch (e) {
            alert("Error instantiating objects: " + e);
        }
    });

    </script>
</body>
</html>
```

View Example

Since Kendo UI is a JavaScript framework, the only thing required to use it is to include the common CSS file, a theme CSS file (default in this case), jQuery and the Kendo UI JavaScript library. configures the dataSource property for the Kendo UI Grid to use the JSDO dialect. The type property is set to "jsdo" and the jsdo proeprty of the transport specifies the name of the resource.

This example instantiates the JSDOSession and configures the dataSource property for the Kendo UI Grid to use the JSDO dialect. **The type property is set to "jsdo" and the jsdo property of the transport specifies the name of the resource.** When the Kendo UI DataSource is instantiated an instance of the JSDO object will be created internally. All Kendo UI widgets use a DataSource to define what data they can display and manipulate. A Kendo UI DataSource can define read, update, create and destroy endpoints that are called “transports”. In this example with the new version of the JSDO, the transports are defined automatically.

4: Sharing a Kendo UI DataSource Between Widgets

While a Kendo UI DataSource can be defined in a widget configuration object (as we have seen so far), it can also be defined as a stand-alone object. This allows it to be used by more than one widget. Aside from rich data management widgets, Kendo UI also includes an extensive charting a data visualization library. These widgets use new HTML5 SVG standards to draw animated and interactive charts right in the browser. They also take care of handling older browsers (back to IE 7) where SVG is not supported.

In order to add a chart to this page which displays the same data as the grid, it is first necessary to pull the DataSource declaration out into a separate object.

```
var serviceURI = base + 'http://oemobiledemo.progress.com/CustomerService',
    catalogURI = base + '/static/mobile/CustomerService.json',
    resourceName = 'Customer';

// create a datasource that can be shared between widgets
// use previously created session
var customerDS = new kendo.data.DataSource({
         transport: {
                  jsdo: resourceName
         },
         error: function (e) {
                  console.log('Error: ', e);
         }
});

// the grid’s dataSource can now be set directly to the customerDS object
$("#grid").kendoGrid({
    dataSource: customerDS,
    height: 350,
    groupable: true,
    reorderable: true,
    resizable: true,
    sortable: true,
    pageable: {
        refresh: true,
        pageSizes: true,
        pageSize: 10,
        buttonCount: 5
    },
    editable: "inline",
    toolbar: ["create"],
    columns: [
      { field: "CustNum", title: "Cust Num", type: "int", width: 100 },
      { field: "Name" },
      { field: "State" },
      { field: "Country" },
      { command: ["edit", "destroy"], title: "&nbsp;", width: "250px" }
    ]
});
```

It is now possible to add any other Kendo UI widget or data visualization component and use the exact same DataSource.

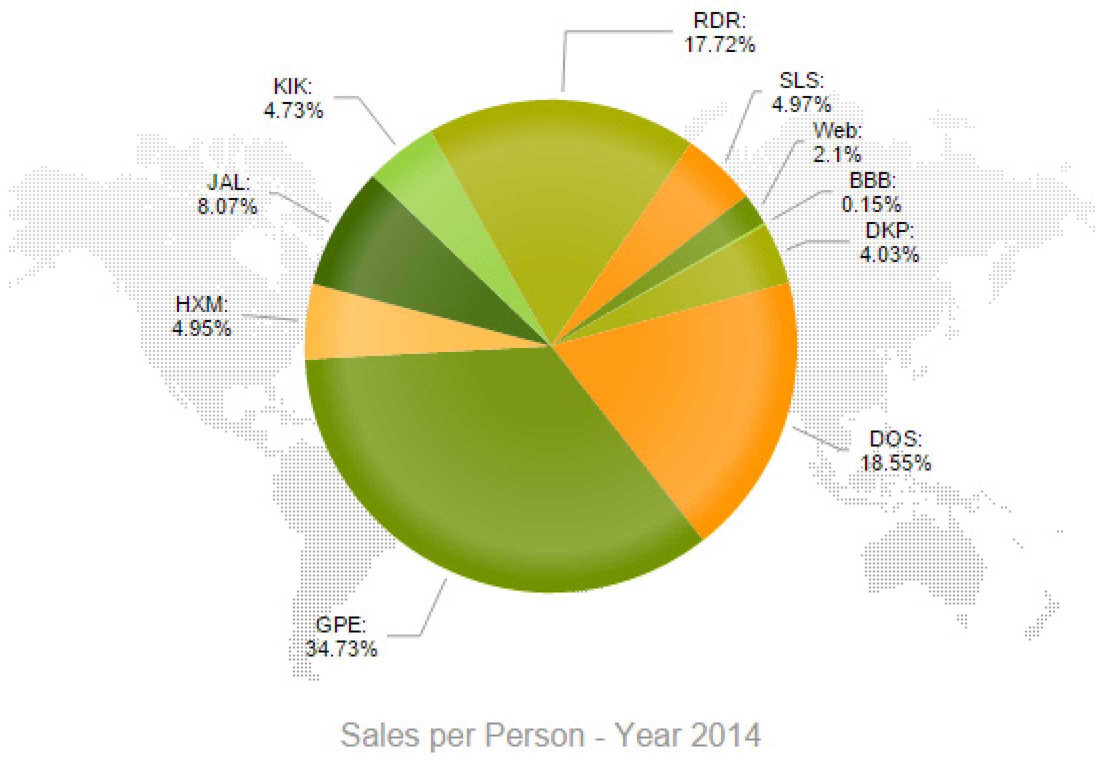
5: Using the JSDO with Kendo UI Charts

Kendo UI has an extensive charting and data visualization library. OpenEdge data retrieved using the JSDO can be displayed using Kendo UI Charts. The data retrieved by the JSDO is processed in JavaScript then passed to a Kendo UI Chart using the format that it expects. The Kendo UI DataSource is not used in these examples.

The following examples are included in the zip file:

- Pie Chart: The Sales per Person for Year 2014 chart, uses the JSDO to call an INVOKE operation called `MonthlySales()` that returns the monthly sales per `SalesRep`, the data is processed to calculate the total for the sales and produce the series that the Kendo UI Pie Chart component uses.

Figure 4: The Kendo UI Pie Chart component showing sales information from an OpenEdge database

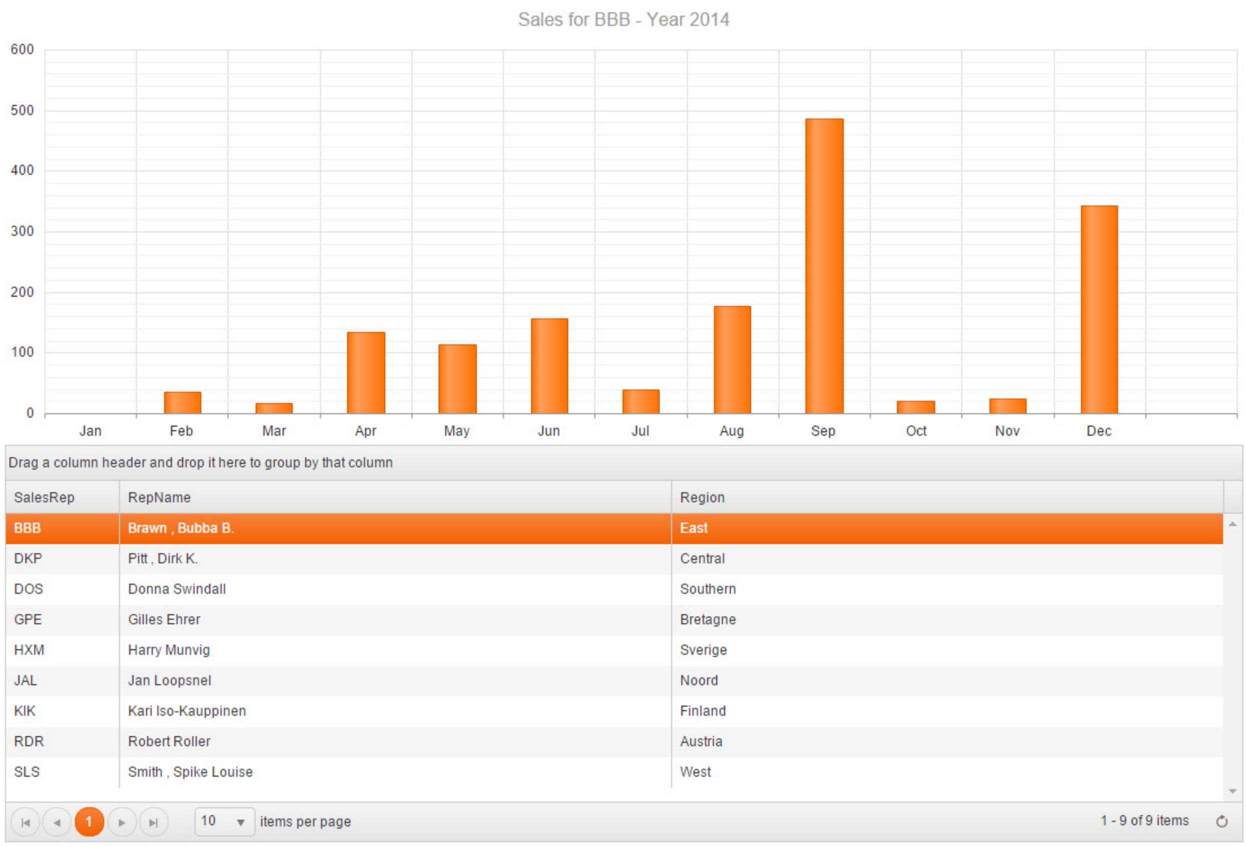


Notes
This example is available at: http://oemobiledemo.progress.com/jsdo/example015.html

The source code for these and other sample programs using Kendo UI can be downloaded from the following location: https://community.progress.com/community_groups/openedge_development/m/documents/2020.aspx

Bar Chart: The Monthly Sales for Year 2014 chart, combines a Kendo UI Grid with a Kendo UI Bar Chart. It uses the same INVOKE operation as the previous example, but processes the data to calculate the total sales per month for the `SalesRep` selected in the grid. If no `SalesRep` is selected, a total for all the `SalesRep` is shown.

Figure 5: A sample app combining the Kendo UI Bar Chart and Grid components



SalesRep	RepName	Region
BBB	Brawn , Bubba B.	East
DKP	Pitt , Dirk K.	Central
DOS	Donna Swindall	Southern
GPE	Gilles Ehrer	Bretagne
HXM	Harry Munvig	Sverige
JAL	Jan Loopsnel	Noord
KIK	Kari Iso-Kauppinen	Finland
RDR	Robert Roller	Austria
SLS	Smith , Spike Louise	West

Notes
This example is available at: http://oemobiledemo.progress.com/jsdo/example016.html

6: Next Steps

Here are some ideas to further your knowledge on using the JSDO with Kendo UI:

- Download the zip file "Examples using the JSDO v4.0 with Kendo UI" from Progress Communities and run the examples in your web browser: https://community.progress.com/community_groups/openedge_development/m/documents/2020.aspx

- Visit the JSDO open source content on GitHub: https://github.com/CloudDataObject
  The JSDO is the JavaScript implementation of the Cloud Data Object (CDO). On this site, you will find the specification for the CDO, source code and libraries for the JSDO (a JavaScript implementation of the CDO) and additional sample projects using the JSDO.

- Visit the demo site for Kendo UI: http://demos.telerik.com/kendo-ui
  On this site, you will find examples for the Kendo UI components including the Grid, ListView and DataSource components.

- Create a Mobile app using Telerik Platform (free trial is available at www.telerik.com) and use the JSDO to access data in the OpenEdge or Rollbase backend:

  1. Create a new AppBuilder Hybrid project (in a Hybrid app). Select the "Progress Data Service" template.
  2. Use the info in the README.txt file to update the scripts/jsdosettings.js file to point to an existing OpenEdge Mobile Service you have created or use the jsdoSettings in README.txt that refer to a demo service.
  3. Use the Run menu to run your app in a Simulator.

  To run the app on a Mobile device:

  1. Download the Telerik AppBuilder companion app from the app store for your mobile device.
  2. Build your app for the target device.
  3. Scan the supplied QR Code to launch the app on the device. You can use the QR Code feature in the Telerik AppBuilder companion app.

- Review the OpenEdge Mobile documentation to learn how to enable support for serverPaging, serverFiltering, serverSorting and batching in your Business Entities using Progress Developer Studio 11.4 or greater:
  https://documentation.progress.com/output/oemobile1151

7: References

The following links point to the documentation for Kendo UI and for OpenEdge related to this white paper:

- http://docs.telerik.com/kendo-ui/framework/datasource/overview
- http://docs.telerik.com/kendo-ui/api/javascript/data/datasource
- http://demos.telerik.com/kendo-ui/datasource/index
- http://demos.telerik.com/kendo-ui/grid/index
- http://demos.telerik.com/kendo-ui/grid/hierarchy
- http://demos.telerik.com/kendo-ui/pie-charts/index
- http://documentation.progress.com/output/OpenEdge114/openedge114/#page/dvmad/OpenEdge.049.html#
- http://documentation.progress.com/output/OpenEdge114/openedge114/#page/pdsoe/OpenEdge.0287.html#wwconnect_header
- http://documentation.progress.com/output/OpenEdge115/openedge115/#page/pdsoe/building-mobile-applications.html#wwconnect_header
- https://documentation.progress.com/output/oemobile1151
- https://community.progress.com/community_groups/openedge_development/m/documents/2078.aspx