How to represent hierarchical data

using JSDO

Author : Anil Kumar

Version : 1.1

Last Updated : 7th Nov 2016

Contents

[Introduction 3](#_Toc466305329)

[Approaches for representing hierarchical data 4](#_Toc466305330)

[Individual properties information 4](#_Toc466305331)

[Supported Configuration 5](#_Toc466305332)

[Procedure and Ideal setup for hierarchical data representation 6](#_Toc466305333)

[Telerik Mobile Sample 10](#_Toc466305334)

[Unsupported Usecase(s): 11](#_Toc466305335)

[Future 11](#_Toc466305336)

# Introduction

In order to support Kendo UI DataSource (for communication with Kendo UI Grid or any datasource related controls/views pertain to Kendo UI framework) from JSDO (and thus to any of OpenEdge or Rollbase backend environments) we have introduced new feature by name JSDO Datasource (aka JSDO Dialect) in release 4.0 of JSDO.

There are bunch of properties that JSDO Datasource implicitly supports. The two properties namely ***readLocal*** and ***autoSave*** are applicable at the JSDO Datasource level. Primarily these flags have impact when nested grids or hierarchical representation of data being used in the frontend (say mobile or web) application (via ‘Kendo UI’ grid or Kendo UI listView or any control which uses Kendo UI DataSource). This can be triggered by providing ***jsdo*** as the transport type in the 'Kendo UI' datasource as below:

dataSource: {

type: "jsdo",

transport: {

jsdo: jsdoInstance,

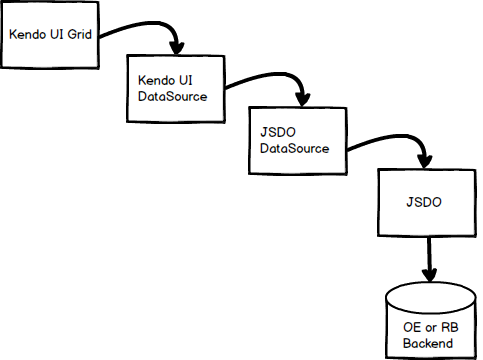
tableRef: "eOrder",

readLocal: true

autoSave: false

}

**Control Flow**:



# Approaches for representing hierarchical data

As Progress Data Object is a client side artifact, one can embed or use this as an intermediate layer for both mobile and web application(s).

1. Hierarchical data representation via **listView** control in a mobile app built using ***Progress Data Service*** template through Telerik Platform environment
2. Hierarchical data representation via **listView** control in a mobile app built using ***Telerik Views Service*** and ***Progress Data Provider*** through Telerik Platform environment
3. Representing parent-child relationship via **Kendo UI Grid** control in a web application

This document aims at building mobile application using Progress Data Service template (approach 1 in above set).

# Individual properties information

**What is readLocal Property:**

Inorder not to have multiple copies of data across Kendo local storage and JSDO memory and also to reduce server calls to OE/RB backend, we have introduced this functionality of ***readLocal*** in which case, Kendo grid (or any widget which uses Kendo datasource) can read from the JSDO memory. Also, this is applicable only for the hierarchical tables/grids and does not have major significance with single level.

However, this usecase is possible or will have impact only when single/same JSDO is used by both parent and child tables. When we have individual JSDO’s that are being used by parent and child, then readLocal property will not have its significance as the datasources operate on individual JSDO’s of their own.

**What is autoSave Property and its Usage:**

Kendo UI Datasource has a property called ***autoSync*** and ***sync()*** method. i.e., when autoSync property is set to TRUE, implicitly sync() method will be triggered/invoked. This sync() method will in turn make a call to JSDO’s saveChanges() method to send all changes to the backend server.

However, there may be a usecase where invoking sync should not trigger saveChanges(). To support this scenario, we have introduced new property called **autoSave**. i.e., setting this property will not let the sync() method to call saveChanges(). i.e., changes are not sent to backend server. Developer need to explicitly call jsdo.saveChanges() whenever required.

This **autoSave** property is ***TRUE*** by default at the JSDO DataSource level

Property **readLocal** primarily deals with the read () operation and not more on the CUD (Create, Update and Delete) operations. This specific property (autoSave) deals with more on the **CUD** operations from mobile application or the Kendo grid application.

# Supported Configuration

As part of hierarchical data representation we have two datasources (Parent and Child) for representation of relevant data on UI. Having said that these properties (readLocal and autoSave) are applicable at individual JSDO datasources (parent and child), we will be ending up in dissimilar combinations of these flags at parent and child datasources. However, following table depicts the supported configuration of these elements for representing data in hierarchical manner.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Parent DataSource** | **Child DataSource** | **Comments** |
| Read Local Property  (readLocal) | https://wiki.progress.com/images/icons/emoticons/error.png | Done | This is applicable only at the child datasource level |
| Auto Save Property  (autoSave) | https://wiki.progress.com/images/icons/emoticons/error.png | Done |  |
| Server Flags  (serverFiltering, serverSorting, serverPaging) | Done | https://wiki.progress.com/images/icons/emoticons/error.png | All of the Kendo UI DataSource server flags are applicable at the parent datasource level only |

**Note**:

The **useRelationships** property which is used to handle relationships in the JSDO API is ignored when working with a hierarchical grid. This because:

* + The preferred model to work with hierarchical grids is to work as native as possible with Kendo UI.
  + The useRelationships requires a working record on the parent table. The interaction with the Kendo UI listView does not set the working record.

**Hierarchical data support in Kendo UI listView with respect to JSDO can be achieved in following ways:**

1. **Usage of common/single JSDO by both parent and child data sources of Kendo UI Grid**:

In this scenario, a resource with multiple tables (say Customer and Order) will be generated as a single dataset in catalog file. In the hierarchical grid support both the Kendo UI DataSources (parent and child) will be using a single instance of the JSDO for communication with the OpenEdge or Rollbase backend servers. The Kendo UI Grid server flags (serverFiltering, serverSorting and serverPaging) can be set to true at the parent level of Kendo UI DataSource (and optionally at the child data source) and the significance of **readLocal** can be seen by using this property (true/false) at the child level Kendo UI DataSource.

1. **Usage of different JSDOs for both parent and child data sources (whose datasource in turn binds to a Kendo UI listView)**:

In this use case, each resource will have individual JSDOs, i.e., in a Customer-Order application, Customer table will have its own JSDO followed by Order table possessing another JSDO. In this approach sorting, filtering and paging functionality can be achieved via the JSON Filter Pattern.

All the server flags of Kendo UI grid (serverFiltering, serverSorting and serverPaging) can be set to true at both parent and child level of the Kendo UI DataSource.

**Technical details**: The JSON Filter Pattern currently works on top level table elements or in other words, JFP approach currently do not have a mechanism of conveying table name in the filter parameters that are sent to server. As we are using two different JSDOs on two different Kendo UI Grids (parent and child) sending READ request with a filter parameter by specifying JFP mechanism we can achieve sorting, paging and filtering mechanism on both the parent and/or the child table.

**Note**: This mechanism always works on server and there is ***no need for the readLocal*** property.

This document aims at building mobile application using common JSDO instance for both parent and child datasources

# Procedure and Ideal setup for hierarchical data representation

As noted in above sections, we can customize the properties at both parent and child datasources level. However, inorder to achieve successful parent-child or hierarchical data representation using any of the Kendo UI datasource controls, following shows ideal setup of JSDO dialect and Kendo UI datasource properties.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Parent DataSource** | **Child DataSource** | **Comments** |
| Read Local Property  (readLocal) | FALSE (Default value) | TRUE | Both readLocal and autoSave of parent datasource are default values which JSDO by default provides. We can leave them unspecified in the datasource layer |
| Auto Save Property  (autoSave) | TRUE (Default value) | FALSE |
| JSDO | Instance1 | Instance1 | Uses same JSDO |

**Steps**:

1. Launch Telerik Platform in a HTML5 compatible browser
2. Create a new mobile app by selecting Progress Data Service option/template from ***Advanced*** section
3. Create individual HTML and JavaScript files for representing UI elements and the relevant business logic with Kendo UI control set. Few of the key functions are as below:
   1. Initialize
   2. Create JSDO Datasource
   3. Submit operation for committing changes to backend environment.

**Glimpse of JSDO DataSource Code Snippets of Parent and child**

**Parent DataSource**

createJSDODataSource: function () {

try {

// create JSDO

if (jsdoSettings && jsdoSettings.resourceName) {

this.jsdoModel = new progress.data.JSDO({ name : jsdoSettings.resourceName,

autoFill : false, events : {

'afterFill’: [ {scope: this, fn : function (jsdo, success, request) {

}

} ],

'beforeFill’: [ {scope: this, fn : function (jsdo, success, request) {

}

} ]

}

});

this.jsdoDataSource = new kendo.data.DataSource({

type: "jsdo",

batch: dataViewModel.useSubmit,

transport: {

jsdo: this.jsdoModel,

tableRef: jsdoSettings.tableName

},

error: function(e) {

console.log("Error: ", e);

}

});

jsdo\_for\_orders = this.jsdoModel;

this.resourceName = jsdoSettings.resourceName;

}

else {

console.log("Warning: jsdoSettings.resourceName not specified");

}

}

catch(ex) {

app.viewModels.dataViewModel.createDataSourceErrorFn({errorObject: ex});

}

},

**Child DataSource**

createJSDODataSource: function() {

try {

// create JSDO

if (jsdoSettings && jsdoSettings.resourceName) {

this.jsdoModel = jsdo\_for\_orders;

if (this.jsdoDataSource == undefined) {

this.jsdoDataSource = new kendo.data.DataSource({

type: "jsdo",

batch: Orders\_of\_Cust\_dataViewModel.useSubmit,

filter: { field: "CustNum", operator: "eq", value: selectedCustNum },

transport: {

jsdo: jsdo\_for\_orders,

tableRef: jsdoSettings.tableName1,

readLocal: true,

autoSave: false

},

error: function(e) {

console.log("Error: ", e);

},

change: function(e){

if (e.action === "itemchange") {

console.log("Order Details got modified ...")

}

}

});

}

else {

app.viewModels.Orders\_of\_Cust\_dataViewModel.jsdoDataSource.filter({field: "CustNum", operator: "eq", value: selectedCustNum});

this.jsdoDataSource.transport.readLocal = true;

this.jsdoDataSource.transport.autoSave=false;

var mylistView = $("#orderofcustomerView").data("kendoMobileListView");

$('#orderofcustomerView').getKendoMobileListView().refresh();

$('#orderofcustomerView').getKendoMobileListView().setDataSource(app.viewModels.Orders\_of\_Cust\_dataViewModel.jsdoDataSource);

}

this.resourceName = jsdoSettings.resourceName;

}

else {

console.log("Warning: jsdoSettings.resourceName not specified");

}

}

catch(ex) {

app.viewModels.Orders\_of\_Cust\_dataViewModel.createDataSourceErrorFn({errorObject: ex});

}

},

**Following table represent information on individual elements of code snippet**:

|  |  |  |
| --- | --- | --- |
|  | **Description** | **Comments** |
| **jsdo\_for\_orders** | We are assigning the JSDO instance of the parent datasource to a global variable which can be used by the child datasource | Both parent and child datasources are using single JSDO instance |
| **App** | Represents Kendo mobile application object |  |
| **viewModels** | Place holder for all view models available in mobile app |  |
| **dataViewModel** | View model of parent screen |  |
| **Orders\_of\_Cust\_dataViewModel** | View model of child screen |  |
| tableRef: jsdoSettings.tableName | tableRef property of parent datasource represents parent table | These two values are declared in the jsdoSettings.js file |
| tableRef: jsdoSettings.tableName1 | tableRef property of child datasource represents child table |

# Telerik Mobile Sample

A sample mobile application has been developed for depicting the usage of the ***readLocal*** and ***autoSave*** properties to achieve hierarchical data representation in Telerik mobile app which uses Progress Data Service template and this application has been developed using Teelrik Platform environment.

* This sample is built on top of Customer (parent table), Order (child) tables of Sports2000 database. i.e., resource is built on multiple tables.
* Sample uses single JSDO instance for both parent and child datasources
* Possess CRUD operations
* Does not make use of Views Service(s) available in Telerik Platform.

Please find the sample in following **GitHub** location:

<https://github.com/CloudDataObject/sample-Hierarchical_PDS_Template>

# Unsupported Usecase(s):

1. When autoSave property against child table datasource is set to FALSE, parent table’s autoSave property is set to TRUE (Default value) then performing SUBMIT (commit changes to server) with some changes to child table data and no changes to parent table records do not result in successful commit to backend server. As a workaround one need to use one of the below:
   1. Create a record in parent table or modify any record such that the parent datasource is dirty (or contents are modified)
   2. Invoke JSDO’s saveChanges method explicitly in the SUBMIT operation. Following is example code snippet (part of submit operation)

if (jsdoDataSource.hasChanges()){

promise = jsdoDataSource.sync();

}

else {

promise = jsdoDataSource.sync();

<Mobile App DataView Model>.jsdoModel. saveChanges(true);

}

1. Server flags of Kendo UI DataSource are independent by themselves. i.e., we can set them at the datasource level without any constraints. However, setting server flags at the child table (or datasource) level in a hierarchical grid is currently (4.2) not supported
   1. This is not supported because of JFP (JSON Filter Pattern) Technical Limitation. i.e., currently we don’t have a mechanism to send the table name to backend server from client side using JSON Filter Pattern approach and it always operates on the parent table.
2. When **autoSave** property against child table datasource is set to True and parent table’s **autoSave** property is set to False, then ideally the changes of the child table should be honored and should be sent to OpenEdge backend server. However, the current behavior will lose all changes made to child table when any CUD operation is performed on parent table record.

# Future

As noted above, mobile apps can be created using Telerik Platform ***views service*** approach which internally uses ***Progress data provider*** for the interaction with OpenEdge backend environment and thus we can represent data in hierarchical manner using this mechanism/approach as well. However, current version of JSDO (4.3) is not completely verified against views service (Telerik Platform 3.5). This will be fully supported in future releases of JSDO and Telerik Platform.