



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

June 6th 2017

Anil Kumar Kotha

Edsel Garcia

Disclaimer

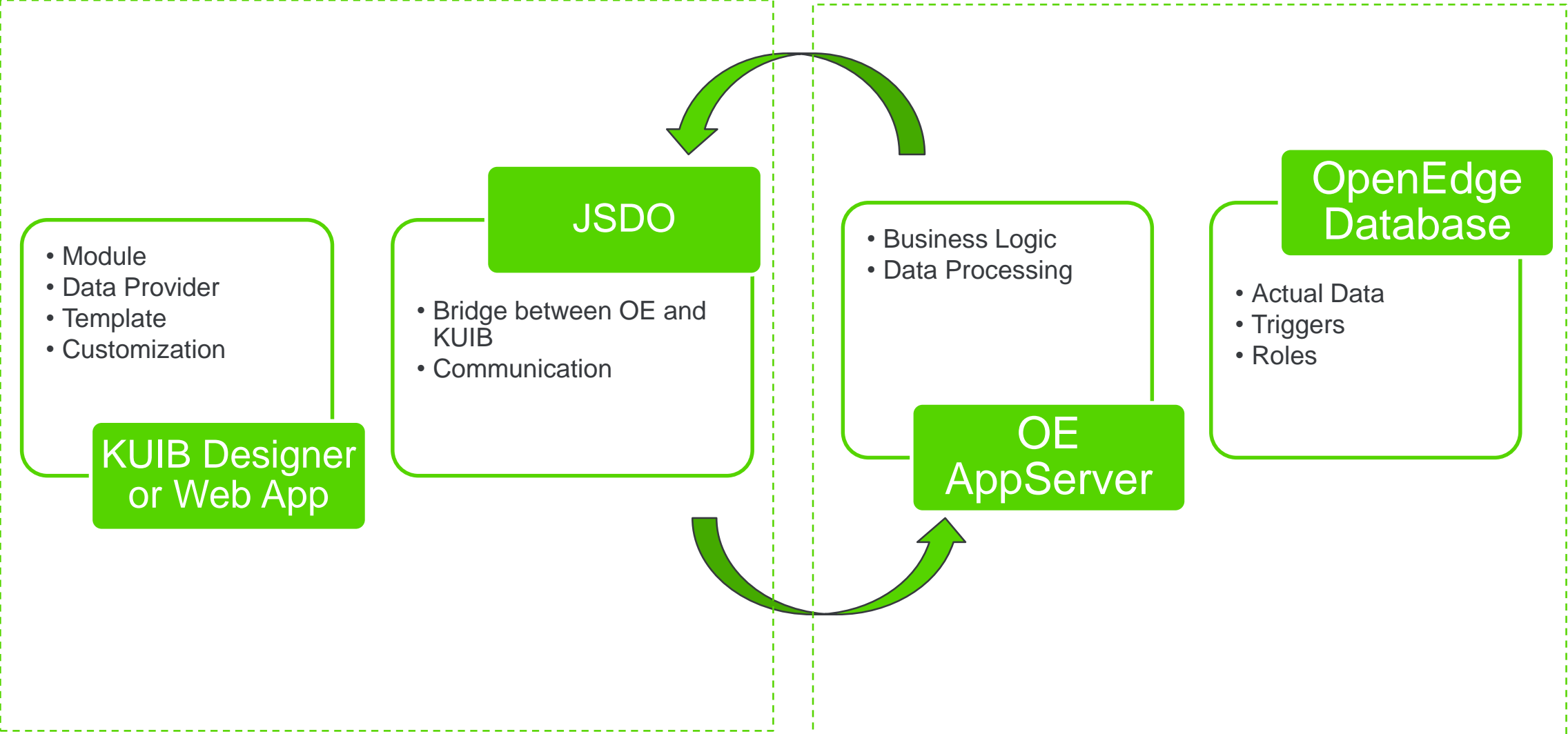
Agenda

- Introduction
- JSDO – JavaScript Data Object
- Business Entity and Extensions
- Working with relational and referential data
- Customizing Kendo UI Builder Web Apps
- OpenEdge Security and Kendo UI Builder

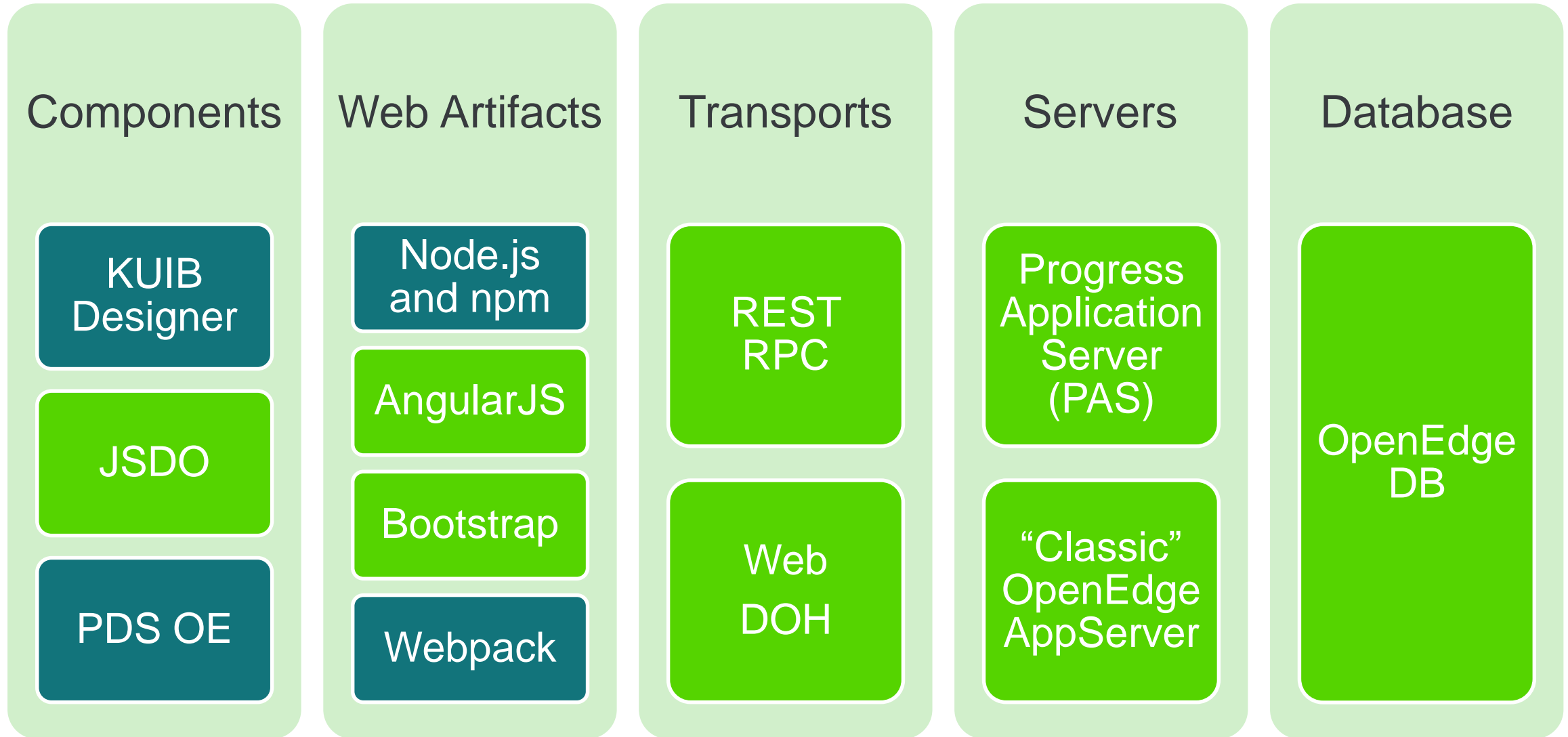
Introduction

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

KUIB App flow



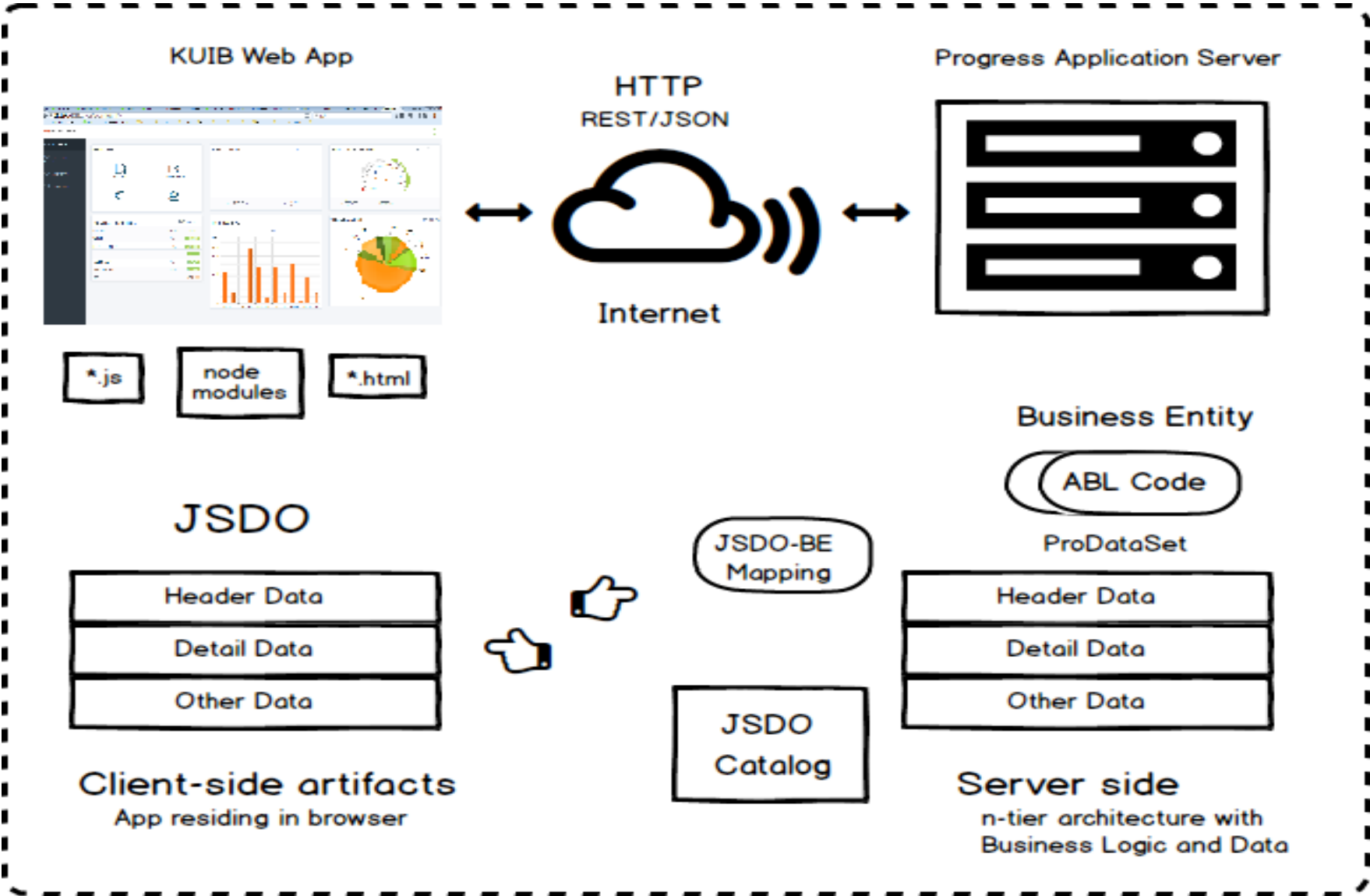
Different Components Involved – Runtime and Design



Agenda

- Introduction
- JSDO – JavaScript Data Object
- Business Entity and Extensions
- 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



JSDO APIs

JSDO Method	Business Entity
add() – Create assign() – Update remove() – Delete	
fill() – Read	READ
saveChanges(false)	CUD
saveChanges(true)	Submit
invoke("myMethod")	myMethod()

Error handling via JSDO

- Use `getErrors()` API
 - Allows us to access all AppServer errors seamlessly

```
jsdo = new progress.data.JSDO({ name: 'CustOrder' });
```

```
...
```

```
jsdoErrors = jsdo.eCustomer.getErrors();
```

- Error handler can be overridden in KUIB

Agenda

- Introduction
- JSDO – JavaScript Data Object
- Business Entity and Extensions
- 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



Business Entities

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

New Business Entity

Select a schema file

Optionally specify a database connection and table to be associated with the resource.

Resource name:

Operations: Read-only CRUD **CRUD and Submit**

Write dataset before image

Select database table

Connection:

Table:

Select schema from file

Schema file:

Schema:

Using: Include file
 Schema definition
 Class Hierarchy

Expose as Data Object service

Resource URI:

Define Service Interface wizard

■ Annotations

- File level
- Method level (CRUD + INVOKE)
- Field level
 - Semantic types
 - Foreign Key

The screenshot shows the 'Define Service Interface' wizard in the 'Edit Annotation' step. The window title is 'Define Service Interface'. The main heading is 'Edit Annotation' with the subtitle 'Edit annotations for the selected files.'.

Select a file: A list of files is shown, with 'Sports/AppServer/AdvCusto' selected.

Annotation details for the selected file: The 'Main Annotation' tab is active. It includes a checked 'Enable Main Annotation' checkbox. Under 'REST annotation details', the 'Execution mode' is set to 'singleton', and 'Return value' and 'Before-image' are unchecked. The 'Resource name' is 'AdvCustomer', the 'Resource URI' is '/AdvCustomer', and the 'Schema file' is 'Sports/AppServer/AdvCustomer.cls'. The 'Schema' section shows 'ttCustomer' and 'dsCustomer' as available options, with 'dsCustomer' selected. The 'Schema name' is 'dsCustomer'. The 'Main annotation for the above selected file:' text area contains the following code: `@openapi.openapi.export FILE(type="REST", executionMode="singleton", useR
@progress.service.resource FILE(name="AdvCustomer", URI="/AdvCustomer", sc`.

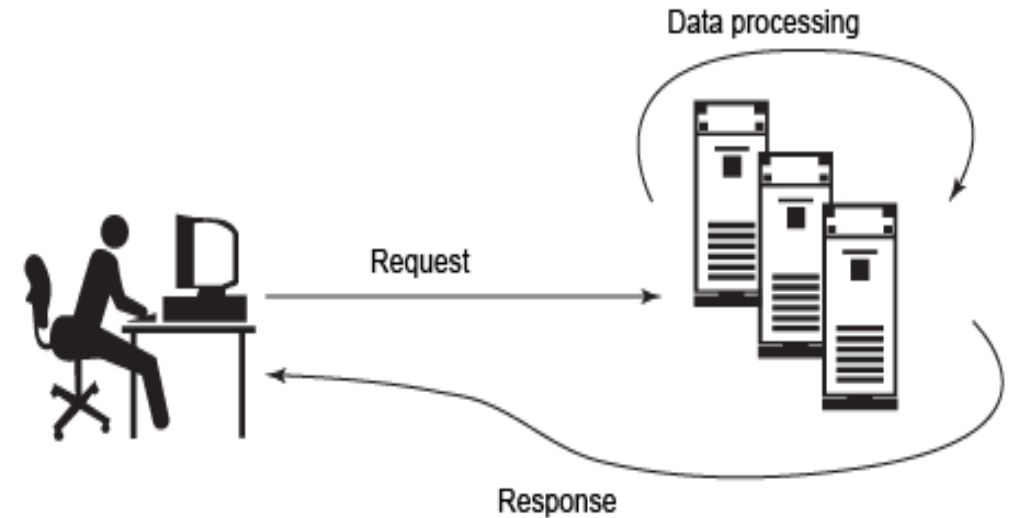
Navigation buttons at the bottom include '< Back', 'Next >', 'Finish', and 'Cancel'.

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

Client-side Processing
Count Function ⓘ

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

```
@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="ablFilter,top,skip,id,orderBy").
METHOD PUBLIC VOID ReadCustomer(INPUT filter AS CHARACTER, OUTPUT DATASET dsCustomer):
```

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

```
@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": []
  },
  ,
```

Mappings

- Request mapping

```
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

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



DEMO

Agenda

- Introduction
- Business Entity and Extensions
- JSDO – JavaScript Data Object
- 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

The image shows a form with several fields. On the left, there are fields for 'Customer' (with a dropdown menu showing 'Game Set Match'), 'Address' (a large grey placeholder), and 'Shipping Policy' (an empty text box). On the right, there is a 'Sales Person' dropdown menu. The dropdown is open, showing a search bar with a magnifying glass icon and a list of names: 'Brawn , Bubba B.' (highlighted in green), 'Pitt , Dirk K.s', 'Donna Swindall', 'Gilles Ehrers', 'Harry Munvigs', and 'Jan Loopsnel'. The 'Sales Person' label and the dropdown menu itself are enclosed in a green rectangular box.

```
@openapi.openedge.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
- Supports Inline, Popup, Incell editing modes
- Allows CRUD operations on child table data
- Single relation or multi-relation(s) among tables
- Both parent and child tables should be in single resource

Agenda

- Introduction
- JSDO – JavaScript Data Object
- Business Entity and Extensions
- 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

Edit Data Source

Name: Applicant

Search: ApplicantWeb, Applicant

Properties:

Label	Email Address
Editor Type	email-input

Excluded fields: (empty)

Included fields: ApplicantID (Number), Name (RichText), Skills_1 (Text), Skills_2 (Text), Skills_3 (Text), Email (Email), Phone (PhoneNumber)

Include All / Exclude All

Client-side Processing

Save Cancel Need Help?

Customizing KUIB Code

- Custom Sections
- View Factory
- Public Controller
- Other Assets
- Custom Templates

Recommendations

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



DEMO

Agenda

- Introduction
- Business Entity and Extensions
- JSDO – JavaScript Data Object
- Working with relational and referential data
- Customizing Kendo UI Builder Web Apps
- OpenEdge Security and Kendo UI Builder

Authentication

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

- JSDO Specific:
 - Use **progress.data.JSDOSession**
 - progress.data.Session (plans to deprecate in future)



Enabling Authentication at PASOE layer

- Modifications to *oeab/Security.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:
##
##      name                Description
##      =====
##      anonymous           No user login - all clients have public access
##      basic              Users authenticate using the HTTP BASIC standard
##      form               Users authenticate using a HTTP POST message &
##                        form data
##      container          Users authenticate via Tomcat realm services and
##                        authorize URL access via Spring Security
##      sso                OpenEdge Single Sign-on using ClientPrincipal
##                        access tokens
##
##
```

```
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

Edit Data Provider

Name ⓘ

Service URI ⓘ

Catalog URI ⓘ

Authentication Model ⓘ

Anonymous ▼

Anonymous

Basic

Form

Summary

- Custom Business Entities to leverage business logic
- Flexibility with custom views in KUIB
 - Pre-defined views
 - User-defined (Blank) view
- Access to large set of Kendo UI components via KUIB
- Use API's to improve maintainability of code

Happy Developing !!!

KUIB is the 'Key' to
Modernization



PUGCHALLENGE  **EXCHANGE**
AMERICAS