OpenEdge REST Backends, JSDO, Kendo UI & Angular
Modern technology for modern web frontends

Mike Fechner, Consultingwerk Ltd.
mike.fechner@consultingwerk.de
Consultingwerk Ltd.

- Independent IT consulting organization
- Focusing on OpenEdge and related technology
- Located in Cologne, Germany, subsidiary in UK
- Customers in Europe, North America, Australia and South Africa
- Vendor of developer tools and consulting services
- 27 years of Progress experience (V5 … OE11)
- Specialized in GUI for .NET, Angular, OO, Software Architecture, Application Integration
Sample Code on Github

- https://github.com/consultingwerk/Angular2JsdoSamples
Agenda

- **Angular / Angular 2**
  - Kendo UI Components for Angular
  - JSDO
  - OpenEdge REST Backends
  - Using the JSDO with Angular 2
  - Using the JSDO with Type Script Bindings
  - Application Infrastructure Components
Angular

- Application development **platform** for web and **mobile applications**
- Open-Source, developed primarily by **Google** and others
  - **Progress Telerik** (NativeScript), **Microsoft**
- Development framework
- Component based architecture
- Dependency injection
- Data binding
- Object oriented
Angular Development Environment

- **Angular CLI** (command line interface)
  - Create new projects and modules
  - Create new components, services, etc.
  - Execute, Test and Deploy Application

- **TypeScript** (JavaScript compatible object oriented language), developed by Anders Hejlsberg (Microsoft), author of C#, Delphi and Turbo Pascal
  - Finally a well designed language for the web
Angular Development Environment

- Choice of TypeScript and Angular aware editors
  - Microsoft Visual Studio Code (free and cross platform)
  - Sublime Text
  - Eclipse, ...

- npm – Node Package Manager
  - Repository of libraries for JavaScript and Angular Development
  - Management of peer dependencies
Angular vs. AngularJS

- Angular is not the next version of AngularJS (currently in version 1.5)
- Angular is a complete rewrite in a new language (TypeScript)
- Angular JS is just a framework
- Compatibility and conversion of Angular JS - NO goal during the development
- Angular developed with mobile as priority (performance of mobile applications important)
- Angular performs up to 5 times better
Angular vs. AngularJS

- **AngularJS controllers** no longer present in Angular
- Angular introduces **components**
Angular vs. Angular

- Angular 2 was the first version of Angular (released September 2016)
- Current Version is Angular v4.0 (released March 2017)
- Version 3 was skipped (to avoid confusion about versions …)
- Due to rapid development, the version number is no longer part of the “product” name
- Angular 2 still widely used as a synonym for Angular, ng used as the typical abbreviation
Demo

- Create new Angular CLI project
- "ng serve" to start the development server
- Generate simple component
- Insert into app.html
- Change component property in code
- Demonstrate data binding and auto-refresh
Agenda

- Angular / Angular 2
- **Kendo UI Components for Angular**
- JSDO
- OpenEdge REST Backends
- Using the JSDO with Angular 2
- Using the JSDO with Type Script Bindings
- Application Infrastructure Components
Kendo UI for Angular

- Two versions of Kendo UI available
  - New set of components for Angular 2
  - Previous set of components, now called Kendo UI for JQuery (with AngularJS bindings)
- Different implementations, complete rewrite in TypeScript
- Similar set of components
- Not all components available yet
Kendo UI for Angular Roadmap

- RC.0 shipped January 2017
- Included in Kendo UI Professional
- Scheduler, TreeView, TreeList, and Editor will be added in later releases
- Feature parity with JQuery based Kendo UI planned
Demo

- Show grid demos on http://www.telerik.com/kendo-angular-ui/components/grid/, including
  - „grouping“ (on plunker)
  - „detail template“ (on plunker)
- Inputs
- Scrollview
- Upload
- …
Data Source Support

- Kendo UI for Angular does not have its own concept of Data Sources
- **Data Source in Kendo for JQuery** (AngularJS) provides abstraction type of remote data, including full automation of Batching/Paging/Sorting/Filtering
- Kendo UI Data Source for the JSDO provided automated binding of Kendo UI components to OpenEdge Business Entities
- **No such component available for Angular 2!**
Data Source Support

- Kendo Components bound to `Array`'s (table = array of row objects)
- For paging, data should be bound to a `GridDataResult` object (provided by Kendo UI)
- Data Access to be implemented in Application source code, instead of a UI component (Kendo Data Source)
- Flexibility, but more responsibility for developer
Agenda

- Angular / Angular 2
- Kendo UI Components for Angular
- JSDO
  - OpenEdge REST Backends
  - Using the JSDO with Angular 2
  - Using the JSDO with Type Script Bindings
  - Application Infrastructure Components
JSDO

- JavaScript Library to provide access for JavaScript (Web Browser, Mobile, Rollbase) clients to OpenEdge Data Object Services (Business Entities)
- Introduced in OpenEdge 11.2 for OpenEdge Mobile
- Included in Telerik Platform
- Included in Rollbase
- Can be used with any JavaScript client
- Github, Apache license, royalty free
JSON Catalog

- Describes capabilities of OpenEdge backend resource to JSDO
- Required to create JSDO instance
- Describes methods for
  - create, update, delete
  - read
  - submit
  - count
  - custom operations (invokable methods)
Demo

- Simple JSDO Sample (show source and explain)
  http://oemobiledemo.progress.com/jsdo/example001.html

- JSDO with JQuery grid and Kendo UI Data Source (show source, explain session, catalog and Kendo UI Grid constructor)
  http://oemobiledemo.progress.com/jsdo/example014.html
Catalog Header, Addressing
ProDataset Schema definition

```json
{
  "_errorString": {
    "type": "string"
  },
  "CustNum": {
    "type": "integer",
    "ab1Type": "INTEGER",
    "default": 0,
    "title": "Cust Num"
  },
  "Country": {
    "type": "string",
    "ab1Type": "CHARACTER",
    "default": "USA",
    "title": "Country"
  },
  "Name": {
    "type": "string",
    "ab1Type": "CHARACTER",
    "default": "",
    "title": "Kundenname"
  },
  "Address": {
    "type": "string",
    "ab1Type": "CHARACTER",
    "default": "",
    "title": "Address"
  },
  "Address2": {
    "type": "string",
    "ab1Type": "CHARACTER",
    "default": "",
    "title": "Address2"
  },
  "City": {
    "type": "string",
    "ab1Type": "CHARACTER",
    "default": "",
    "title": "City"
  }
}
```
List of supported operations
JSDO and ProDatasets

- The JSDO maps ProDatasets to JavaScript
- Provides DATA-RELATIONS
- Provides TRACKING-CHANGES support required for create, delete and update of records
- Understands validation error messages returned by OpenEdge backend services
Agenda

- Angular / Angular 2
- Kendo UI Components for Angular
- JSDO
- OpenEdge REST Backends
  - Using the JSDO with Angular 2
  - Using the JSDO with Type Script Bindings
  - Application Infrastructure Components
Sample ProDataset JSON output

- `{ }` wraps a single object
- `[ ]` wraps an array of objects
- All strings are quoted
- Data types: Number, String, Boolean, Array, Object, Null
- Everything else must be passed as a String (e.g. Date)
- No real standard for Date
REST Adapter

- JavaServlet that translates REST messages into AppServer calls
- Similar to WSA and AIA
- Tooling integrated into Progress Developer Studio
- Not integrated into ProxyGen
- Can be deployed on standard Tomcat
- Integrated in PASOE as the REST transport
@program FILE(name="CustomerEntity.cls", module="AppServer").
@openapi.openedge.export FILE(type="REST", executionMode="singleton", useReturnValue="false", writeDataSetBeforeImage="false").
@progress.service.resource FILE(name="CustomerEntity", URI="/Customer-Entity", schemaName="dsCustomer", schemaFile="DataObjectService/AppServer/customerentity.

USING Progress.Lang.*.

BLOCK-LEVEL ON ERROR UNDO, THROW.

CLASS CustomerEntity INHERITS BusinessEntity:

    Purpose:
    Notes:

    {"customerentity.i"}


@openapi.openedge.export(type="REST", useReturnValue="false", writeDataSetBeforeImage="true").
@progress.service.resourceMapping(type="REST", operation="read", URI="?filter=\{filter\}", alias="", mediaType="application/json").

METHOD PUBLIC VOID ReadCustomerEntity(
    INPUT filter AS CHARACTER,
    OUTPUT DATASET dsCustomer):

    SUPER:ReadData(filter).

END METHOD.
Create a Data Object service

This wizard allows you to edit a defined Data Object service for a set of ABL resources.

Resources

type filter text

AppServer

CustomerEntity.cls

Sample URI

http://<host>[:port]/rest/DataObjectServiceService/CustomerEntity
Web Handlers

- OpenEdge 11.6, PASOE
- Web Handlers provide a powerful and flexible alternative to the REST Adapter
- „The new WebSpeed“
- URL patterns mapped to ABL Classes
- Can be used to provide required REST Backend for JSDO
- See Mike Fechner‘s 2016 presentation „REST in Peace“
Demo


- Show Catalog and Resource responses for Customer and SalesRep Business Entities
Agenda

- Angular / Angular 2
- Kendo UI Components for Angular
- JSDO
- OpenEdge REST Backends

**Using the JSDO with Angular 2**

- Using the JSDO with Type Script Bindings
- Application Infrastructure Components
JSDO and Angular 2

- JSDO is a JavaScript library
- Angular supports adding JavaScript libraries as untyped code
  - No compile time checks for functions and parameters
  - No Intellisense while typing
- JavaScript libraries can be used in TypeScript
JSDO and Angular 2

- JSDO provides access to ProDatasets and represents them as
  - ProDataset: Object with temp-tables as properties
  - Temp-Table: Array of records
  - Record: JavaScript object with properties for the fields
Basic Sample on Progress Communities

- The following describes steps taken to use the JSDO with Angular 2 based on the following sample from Progress communities:

- [https://community.progress.com/community_groups/mobile/f/17/p/27655/94089](https://community.progress.com/community_groups/mobile/f/17/p/27655/94089)

Thanks to Edsel!
Adding progress.js to app Module

- The JSDO library and TypeScript declaration added to the app folder (sub folder progress)
- The progress.d.ts file declares JavaScript functions in progress.js library
- Basic version provided by Progress
Using JSDO in Angular 2

- Import JSDO in app.ts file

```typescript
// Include progress JSDO module
import { progress } from './progress/progress';
```

- Progress TypeScript declaration for the JSDO is not complete, so some types must be declared as "any"
- "any" instructs TypeScript compiler, that it is not able to perform strong type checking during compilation
- No Intellisense support when editing code
Create JSDO Session and JSDO

- Create JSDOSession instance (AppServer connect)
- Perform Login and Add Catalog
- Create JSDO instance

```javascript
// TODO: Change Session to JSDOSession
let session = new (<any>progress.data.JSDOSession)({
  serviceURI: serviceURI,
  authenticationModel: 'form'
});
session.login('demo', 'demo')
  .done(() => {
    session.addCatalog(catalogURI)
      .done(() => {
        let jsdo1 = new (<any>progress.data.JSDO)({
          tableRef: this.tableName
        });
        this.jsdo = jsdo1;
        this.jsdoLoaded.emit();
      });
});
```
Create JSDO Session and JSDO

- `session.login` and `session.addCatalog` calls are executed asynchronously.
- Best practice is to avoid blocking operations.
- JQuery promises used to „wait-for“ response of login and addCatalog call:
  - `.done (() => {
    // anonymous function
  })` provides callback for completion
  - login and addCatalog promise stacked.
Link JSDO query result to Kendo UI Grid

- (Async) JSDO operations in separate service class
- Service class injected into component descriptor, component subscribes to `jsdoLoaded` event
- This allows the component to initialize in parallel to the JSDO initialization
Demo

- Execute Web app (Demo 1)
- Review code in app.component.ts
  - CustomerJsdoDataService class
  - AppComponent Constructor
  - fetch Method of CustomerJsdoDataService
Agenda

- Angular / Angular 2
- Kendo UI Components for Angular
- JSDO
- OpenEdge REST Backends
- Using the JSDO with Angular 2
  - Using the JSDO with Type Script Bindings
- Application Infrastructure Components
TypeScript Declarations

- Web development relies heavily on JavaScript libraries
- JavaScript libraries lack strong typing (JavaScript is not a strong typed language)
- TypeScript declarations solve this discrepancy
- Declares the interfaces and types of JavaScript libraries, allows weak typed implementation to be treated as strong-typed
TypeScript Declarations

- JavaScript community active in providing TypeScript declarations
  - http://definitelytyped.org/

- Wiki Article on TypeScript declaration best-practices: https://typescript.codeplex.com/wikipage?title=Writing%20Definition%20%28.d.ts%29%20Files
DefinitelyTyped
The repository for high quality TypeScript type definitions

Usage
Include a line like this:

```/// <reference path="jquery/jquery.d.ts" />```
TypeScript Declaration for the JSDO

- Complete declaration available at https://github.com/consultingwerk/JSDO
export module progress {
    export module data {
        export class Session {
            constructor(options?: SessionOptions);
            static AUTH_TYPE_ANON : string;
            static AUTH_TYPE_BASIC : string;
            static AUTH_TYPE_FORM : string;
            login(serviceURI: string, username: string, password: string): void;
            addCatalog(catalogURI: string): void;
            subscribe(eventName: string, callback: Function, scope?: any): void;
            unsubscribe(eventName: string, callback: Function, scope?: any): void;
            unsubscribeAll(eventName: string): void;
        }
    }
}

export class JSDOSSession {
    constructor(options: JSDOSSessionOptions);
    login(username: string, password: string): JQueryPromise;
    addCatalog(catalogURI: string): JQueryPromise;
    subscribe(eventName: string, callback: Function, scope?: any): void;
    unsubscribe(eventName: string, callback: Function, scope?: any): void;
    unsubscribeAll(eventName: string): void;
}
npm repositories

- Node Package Manager
- Structured way of managing dependencies for (web) development projects
- Angular CLI build process pulls libraries from NPM repositories
- Supports updating the local copy of the library when the library provided in the repository is updated
- Alternatively enforce a certain version/range
http://esd.consultingwerkcloud.com:4873
Demo

- Create new Angular CLI project
- Add JSDO from Consultingwerk NPM repository
- Review strong typed access to JSDO in sample application (Demo 2)
Agenda

- Angular / Angular 2
- Kendo UI Components for Angular
- JSDO
- OpenEdge REST Backends
- Using the JSDO with Angular 2
- Using the JSDO with Type Script Bindings

Application Infrastructure Components
Real application requirements

- Avoid duplicating code for communicating with the application backend
- Focus on business logic or specific client side code, not on infrastructure
- Integrated session management
- Security incl. authentication, authorization & menu
- Simple reuse of Data Sources between Components
- Localization
- Hide any complexity caused by the above
Kendo UI and JSDO … the Smart way

- **NgModule** extension
  - Handles JSDOSession configuration
  - Manages Authentication (display Login Dialog, handle session time out, authorization issues)
  - Manages active Data Sources
  - Manages communication between components (links)
Smart-Data-Source

- Encapsulates JSDO instance
- Rich set of configuration options
  - Business Entity Name
  - Table(s), Support for joining child tables in resultset
  - Batch size
- Communication with one or multiple visual components
- Support for parent/child filtering
Smart-Data-Source parent/child mode

```xml
<smart-data-source
    smart-filter-source='salesrepFilter'
    smart-object-name='salesrepDataSource'
    smart-entity-table='eSalesrep'
    smart-navigation-source='salesrepToolbar'>
</smart-data-source>

<smart-data-source
    smart-object-name='customerDataSource'
    smart-data-source='salesrepDataSource'
    smart-foreign-fields='SalesRep,SalesRep'
    smart-entity-table='eCustomer'
    smart-entity-view='eSalesrep'
    smart-navigation-source='customerToolbar'
    smart-filter-source='customerFilter'>
</smart-data-source>
```

smart-data-source and foreign-fields define parent/child relation
Smart-Data-Source UI Binding

- Filter (control selection)
- Navigation Toolbar (control)
- Grids (display, navigation, update)
- Viewer (display, update)
- Simple components (every Angular 2 component can display and update fields)
- Smart-Lookup or Auto-Complete component (complex components with own linked Smart-Data-Source)
Smart-Grid

- Encapsulates Kendo UI Grid
- Manages communication (display, update, navigation, multi-row-selection) with Smart-Data-Source
- Layout either in html code or provided by backend

```html
<div class="col-xs-12" style="margin-top: 1em;">
    <smart-grid
        (selection-changed)="onCustomerGridSelectionChanged($event)"
        smart-object-name='customerGrid'
        smart-data-source='customerDataSource'
</div>
```
Demo

- Demo3 ... review
  - app.module.ts (NgModule)
  - smart-data-source-binding.html

- Demo4 ... review
  - smart-data-source-binding with simple html input tags
  - Execute in browser
Smart-Viewer component

- Container for Input components (and similar components)
- Central data-binding to Smart-Data-Source
- Layout options similar to grid
- Manages update state with Smart-Data-Source
  - during update, disallows navigation form Grid and Navigation Toolbar
SmartComponents NG2 Demo

- Review component definitions in `getting-started-form.html`
- Parent Child relation between SalesRep and Customer
- Filter component
- Smart Viewer
- Smart Lookup
Questions