Component Prototype Project

Date: 01/28/19

By Mil Miketic

[mil@cogransystems.com](mailto:mil@cogransystems.com)

# Project Description

## Introduction

A prototype is required to show a functional component-based architecture using Angularjs 1.5 and Angular Material Design.

To help visualize the requirement a design mockup (not real) has been created at: <https://www.screencast.com/t/1HTkmltYQnUB>

Stage 1 Deliverable: Develop a working stand-alone system

Stage 2: Integrate with Cogran architecture and live api

Stage 3: Develop a parent child dependency component in the accordion

This document outlines the requirements and naming conventions.

## Requirement

The prototype will be named: “sample” Use the name for all your elements within the project. Later in Stage 2 it will be replaced with actual api names.

The sample prototype will perform the key functionality of CRUD [Create (Add), Read (Search), Update (Edit) and Delete] using the angularjs component architecture.

Each component module will be “lazy loaded” using oclazyload (<https://oclazyload.readme.io/>).

The components will be defined within an accordion interface (<https://github.com/LukaszWatroba/v-accordion>).

The data table or grid used is from <http://danielnagy.me/md-data-table/>

### Prototype Workflow

An application template has been provided in Azure git repo. Access details are:

GIT Access: https://cogran.visualstudio.com/DefaultCollection/CogranDev/\_git/CogranComponentPrototype

Access: [support@cogransystems.com](mailto:support@cogransystems.com)  
Pwd: XiongJin

1. Home page will consist of 2 components:  
   <cog-header> -- display text info  
   <sample> -- display a button (named Sample) that lazy loads the sample module
2. On click of the sample button the **sample** form opens a search navbar from right.  
   User inputs search criteria and clicks the Search button.   
   The controller calls the api (sample.json).  
   Any errors from the are noted as a angular material toast that displays apiErrMsg  
   If api returns no object data then display “No records found”  
   If results found then close the right navbar.
3. The search results display in the first accordion pane showing:  
   a) data filter  
   b) the Search Filter icon – on click it will load the right nav bar for search  
   b) results are displayed in a table edit grid – with select, edit, save line and delete actions
4. On click of a field within the table the field can be edited – this is part of table edit functionality
5. On click of Select – the primary key is used to load dependent child components – not in Stage 1 scope
6. On click of Edit - a dialog form displays to edit the fields
7. On click of Save – the line item is saved
8. On click of Delete – a dialog displays to confirm delete

**Note: Styling is not critical – the structure of components (best practice) is most critical**

End of Stage 1 scope

In Stage 2: a dependent controller will be created. The primary id will be used to load the dependent child data in its own accordion.

In Stage 3: The prototype data source will be replaced with a Cogran api and integrated within Cogran application. Includes application styling.

## Components

The functionality of each component is detailed below:

Index.html –load the component header <cog-header> and <cog-sample>

<cog-header> - -simple html file will display  
< sample> - - a button link to the sample module  
 -- it will lazy load the accordion container and default load the component <sample-search>

<sample-search> -- side nav to search form

<sample-grid> - - on search the table edit grid will display. On each row are actions to:

<sample-edit> -- the edit dialog form will also need to able to switch between edit and Add mode.

<sample-delete> - a dialog to confirm delete

### Json Data

The data will have one primary key and three fields and 3 records:

## sample.json

{

"formName": "sample",

"formDescription": "Sample form Description",

"apiErrMsg": null,

"formFields": [

{

"id": "1",

"field01": "Field 1-1",

"field02": "Field 1-2",

"field03": "Field 1-3"

},

{

"id": "2",

"field01": "Field 2-1",

"field02": "Field 2-2",

"field03": "Field 2-3"

},

{

"id": "3",

"field01": "Field 3-1",

"field02": "Field 3-2",

"field03": "Field 3-3"

}

]

}

## Naming Conventions

#### Style Guide

Please use either one of the following style guides. Let me know which one you selected.

## Please use: <https://github.com/toddmotto/angularjs-styleguide/blob/master/README.md> or <https://angular.io/guide/styleguide>

#### Element Naming conventions

The following table is shown the naming conventions for every element:

| **Element** | **Naming style** | **Example** | **usage** |
| --- | --- | --- | --- |
| Modules | lowerCamelCase | angularApp |  |
| Controllers | Functionality + 'Ctrl' | AdminCtrl |  |
| Directives | lowerCamelCase | userInfo |  |
| Filters | lowerCamelCase | userFilter |  |
| Services | UpperCamelCase | User | constructor |
| Factories | lowerCamelCase | dataFactory | others |

#### Cogran General Naming Requirements

For this project the Class name is: **sample**

When defining controllers use: “**controllerAs**” syntax eg controllerAs: ‘**sample**Ctrl’,

##### Cogran file naming conventions – we will use the flat structure

|── app/

│ |── components/

│ |── |── sample (module)/

│ |── |── |── sample.html

│ |── |──|── sample.module.js

│ |── |── |── sample.controller.js

│ |── |── |── sample.component.js

│ |── |── |── sample-help.dialog.html

│ |── |── |── sample-help.component.js

│ |── |── |── sample-search.html

│ |── |── |── sample-search.component.js

│ |── |── |── sample-list.html

│ |── |── |── sample-list.component.js

│ |── |── |── sample-grid.html

│ |── |── |── sample-grid.component.js

│ |── |── |── sample-edit.dialog.html

│ |── |── |── sample-edit.component.js

│ |── |── |── sample-delete.dialog.html

│ |── |── |── sample-delete.component.js

|── index.html