Angular JS Custom Directive for Button Dropdown

(Reusable component for all the projects)

VERSION CONTROL

Prepared by:	Kumar Sahoo, Srikant
Date:	5/11/2015
Reviewed and Accepted by:	B.C.Subramanian
Date:	6/11/2015
Approved by:	Baskaran.Varadarajan, D.A.Soundararajan
Date:	29/12/2015

VERSION HISTORY

Date	Version	Author	Description
5/11/2015	0.1	Kumar Sahoo, Srikant	Initial Draft Version
9/11/2015	1.0	B.C.Subramanian	Final Reviewed
21/12/2015	1.1	S.Kalaiyarasu	Reviewed
28/12/2015	1.2	Baskaran.Varadarajan	Final Approved

Contents

1. INTRODUCTION TO ANGULAR JS CUSTOM DIRECTIVE FOR BUTTON DROPDOWN

1 INTRODUCTION

Custom directives are used in AngularJS to extend the functionality of HTML. Custom directives are defined using "directive" function. A custom directive simply replaces the element for which it is activated.

AngularJS application during bootstrap finds the matching elements and do one time activity using its compile () method of the custom directive then process the element using link () method of the custom directive based on the scope of the directive.

Directives are markers on a DOM element (such as an attribute, element name, comment or CSS class) that tell AngularJS's HTML compiler ($compile) to attach a specified behaviour to that DOM element (e.g. via event listeners), or even to transform the DOM element and its children.

- ButtonDropDown directive is used to work on multiple action through single button.

- Includes both a select-style dropdown and a menu-style dropdown. The menu-style dropdown attaches to an existing element (button, link, div, etc), whereas the select-style dropdown replaces the element it is attached to.

1.1 PURPOSE

- The ButtonDropDown is mostly used to do multiple action through single button element. So that the multiple actions will be triggered from parent button clicks.

1.2 SUPPORTING ELEMENTS BY ANGULARJS TO CREATE DIRECTIVE FOR DROPDOWN

- **Element directives** − Directive activates when a matching element is encountered.
- **Attribute** − Directive activates when a matching attribute is encountered.
- **CSS** − Directive activates when a matching css style is encountered.
- **Comment** − Directive activates when a matching comment is encountered.

1.3 INTENDED AUDIENCE

- General Users who use the application:

- ✓ General users will use the ButtonDropDown to do specific action in page.

- UI Developers:

  UI Developers who want to work on refining or adding new functionalities to the directive will be able to understand the basic logic and mechanisms very swiftly upon referring this document.

1.4 DEFINING A DIRECTIVE

This section lists simple steps to define a custom directive in an AngularJS module. First, we need to define an Angular app.

```
var myApp = angular.module('myApp', []);
```

Now, define a directive.

```
myApp.directive('myDirective', function() {
   return {
      restrict: 'E',
template: '<h1>I made a directive!</h1>'
   };
});
```

This defines a directive. restrict: 'E' means "restrict the usage of this directive to only Elements." Thus we embed this directive in the HTML page as

```
<body ng-app="myApp">
   <my-directive></my-directive>
</body>
```

This code piece is equivalent to

```
<body ng-app="myApp">
   <h1>I made a directive!</h1>
</body>
```

Note that AngularJS maps the naming conventions from HTML's **my-directive** to JavaScript"s **myDirective**

1.5 SAMPLE CODE

1.5.1 ANGULAR DIRECTIVE CODE

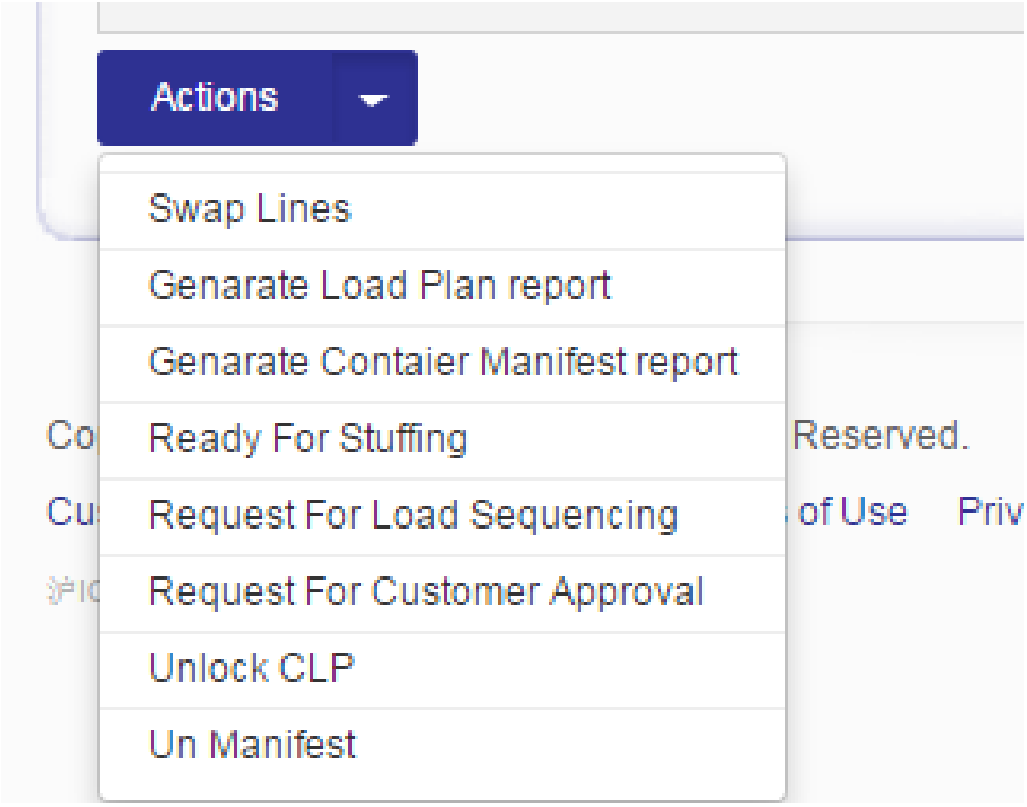
```
(function () {
    angular.module('projectNameUI.common.projectNameForm').directive('projectNameDynamicAttrButtonDropdown', projectNameDynamicAttrButtonDropdown);

    /* @ngInject */
    function projectNameDynamicAttrButtonDropdown($compile){

        var directive = {
            restrict: 'E',
            templateUrl:'app_modules/common/projectNameForm/directives/views/projectNameButtonDropdown.html',
            scope:{
                configJSON:'=configjson',
                changeFunction:'&change'
            },

            link: function (scope, element, attrs, ctrl, $log) {
                }
        };
        return directive;
    }
})();
```

1.5.2 HTML VIEW CODE

```
<button id="split-button" type="button" class="btn btn_primary" style="margin-right: 0px;">{{itemInfo.label}}
</button>
    <button type="button" class="btn btn_primary" uib-dropdown-toggle ng-disabled="disabled"><span
            class="caret"></span></button>
    <ul class="btn_dropmenu_options uib-dropdown-menu uib-dropdown ul" style="width: auto !important;border: 1px solid #DCDBDB;border-top: none;list-style:
    none;overflow: hidden;border-radius: 0px 0px 5px 5px;" id="test" role="menu" aria-labelledby="single-button">
        <li class="divider" ng-hide="select.show === false" ng-repeat="select in itemInfo.Val.options">
            <a style="width: auto !important;padding: 3px 15px !important;font-size : 12px;"
                    ng-click="changeFunction()(select.Link)">{{select.Text}}</a>
        </li>
        </ul>
```

2. BUTTON DROPDOWN INFORMATION USING ANGULAR JS

2.1 VIEW (SAMPLE DATA)



2.2 GENERAL INFORMATION ABOUT THE DIRECTIVE

- **View**

- **Directive Name:**
  <projectName>-dynamic-attr-button-dropdown

- **Directive Type:**
  Button Dropdown directive

- **Attributes to add:**
  Two attributes needs to add **configjson="configJSON" and change="Action"** at the time of implementation .

  Ex:-
  <projectName-dynamic-attr-button-dropdown field="Action_ButtonDropdown" configjson="configJSON" change="Action"/>

- **Location:**

  Directive: Directive:
  <ProjectName>_CustApps_Web_UI_Client\app\app_modules\common\projectNameForm\directives\projectNameButtonDropdown.js

  View:
  <ProjectName>_CustApps_Web_UI_Client\app\app_modules\common\projectNameForm\directives\views\projectNameButtonDropdown.html

2.3 DB JSON OBJECT STRUCTURE

```
"Action_ButtonDropdown" : {
   "type" : "btn_dropdown",
   "label" : "Actions",
   "name" : "Action",
   "class" : "btn_primary",
   "show" : true,
   "projectNameVal" : {
        "options" : [
                  {
       "Text" : "Edit",
       "Link" : {mode:"Edit", url:"/COB/RP/updateCustomerConsignee"}
   },

   {
       "Text" : "Delete",
       "Link" : {mode:"Delete", url:"/COB/deleteCustomerConsignee"}
   }
      ]
  }
}
```

2.4 JSON OBJECT VARIABLE DETAILS

Variable Name	Accepts type	Explanation
type	btn_dropdown	Directive Type "projectNamebutton"
label	String	
name	String	Used in view page
Class	String	Primary or secondary
show	Boolean	Display or hide
projectNameVal	String	Option available in button dropdown

2.5 CURRENT ISSUES AND ADDRESSES

- All Console.log( ) usages have been removed from the directive.
- HTML view and the Directive have been separated.
- Unused scope variables have been removed.

2.6 PROS AND CONS

Pros:

- Easy way of creating button dropdown with the required functionalities with minimal HTML code.
- Re-usable component which will be used across the application.

Cons:

- Need to have some understanding in the directive code to use it to good effect.