Angular JS Custom Directive for From to Date

(Reusable component for all the projects)

VERSION CONTROL

Prepared by:	Ashok Jeyachandran
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	Ashok Jeyachandran	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 FROM TO DATE

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.

- <projectName>FrmToDate is a custom directive which is used to get from and to date values.
- All the date functionalities are handled with this directive which in-turn uses bootstrap date-picker directive.

1.1 PURPOSE

- Using <projectName>FrmToDate directive user will be able to select the From and To dates using the date picker pop-up or by manually entering the dates in the required format.

1.2 SUPPORTING ELEMENTS BY ANGULAR JS TO CREATE DIRECTIVE FOR GRID

- **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:
- 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**

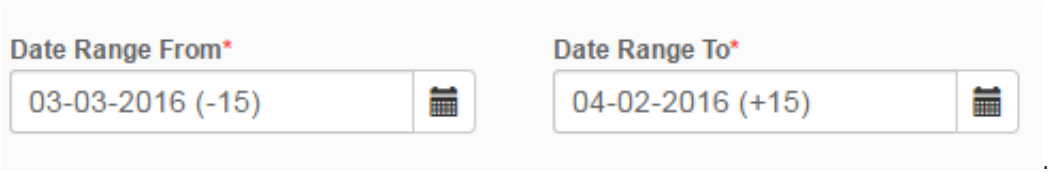
1.5 SAMPLE CODE

1.5.1 ANGULAR AND HTML DIRECTIVE CODE

```
(function () {
    angular.module('<projectName>UI.common.<projectName>Form').directive('<projectName>DynamicAttrFrmToDate', <projectName>DynamicAttrFr

    function <projectName>DynamicAttrFrmToDate($compile, $filter, dateFilter, multiFieldFactory, $parse, userSettings, $log, $http, $tim
        var template ='<div class="{{classVal}} ht60 dateLft parent-class">' +
            '<label>{{itemInfo1.label}}<span class="mandatory" ng-if="itemInfo1.<projectName>Val.required">*</span></label><div class="i
            '<input autocomplete="off" type="text" placeholder={{placeholder}} class="form-control first" ng-model="model1" uib-datepick
            '   ng-disabled="disabledfield" ng-keypress="fromKeyPress()" max-date="model2" ng-change="changeFunction()" ng-blur="addDays(
            '   datepicker-options="FromDateOptions" datepicker-append-to-body="true" ng-pattern="" ng-required="itemInfo1.<projectName>V
            '<span class="input-group-btn mlmin2"><button tabindex="-1" type="button" class="btn btn-default" ng-disabled="disabledfield'

            ' <div class="glyphicon glyphicon-calendar"></div></button></span></div>' +
                //[SOON: Work around IE shows Invalid date with valid integer]'<span class="error" ng-show="(form[itemInfo1.name].$dirty
            '<span class="error" ng-show="(form[itemInfo1.name].$dirty || submitted) && ((form[itemInfo1.name].$error.date && isOkay0))":
            '<span class="error" ng-show="(form[itemInfo1.name].$dirty || submitted) && form[itemInfo1.name].$invalid && !form[itemInfo1
            '<span class="error" ng-show="((form[itemInfo1.name].$dirty || submitted) && !form[itemInfo1.name].$error.date && form[itemI

            '<div class="{{classVal}} ht60 dateRgt parent-class">' +
            '<label>{{itemInfo2.label}}<span class="mandatory" ng-if="itemInfo2.<projectName>Val.required">*</span></label>' +
            '      <div class="input-group" id="toDate">  <input autocomplete="off" type="text" placeholder={{placeholder}} class="form-
            '         ng-disabled="disabledfield" ng-change="changeFunction()" ng-keypress="toKeyPress()" min-date="model1" ng-blur="add
            '           ng-required="itemInfo2.<projectName>Val.required" />  <span class="input-group-btn mlmin2">  ' +
            '   <button tabindex="-1" class="btn btn-default" type="button" ng-disabled="disabledfield" ng-click="openTo($event)">   ' +
            '      <i class="glyphicon glyphicon-calendar"></i></button>   </span>  </div>   '+
                //[SOON: Work around IE shows Invalid date with valid integer]'<span class="error" ng-show="(form[itemInfo2.name].$dirty
            '<span class="error" ng-show="(form[itemInfo2.name].$dirty || submitted) && ((form[itemInfo2.name].$error.date && isOkay1))":
            '<span class="error" ng-show="((form[itemInfo2.name].$dirty || submitted) && !form[itemInfo2.name].$error.date && form[itemI
        var directive = {
            restrict: 'EA',
            require:'^?form',
            scope:{
                configJSON: '=configjson',
                model1: '=ngModel1',
                model2: '=ngModel2',
                submitted: '=',
                disabledfield: '=?',
                changeFunction:'&change',
                focus :'='
            },
            link: function (scope, element, attrs,ctrl) {
```

2. FROM TO DATE DIRECTIVE INFORMATION USING ANGULAR JS

2.1 VIEW (SAMPLE)



2.2 GENERAL INFORMATION ABOUT THE DIRECTIVE

- **Directive Name:**
  <projectName>-dynamic-attr-frm-to-date

- **Directive Type:**
  Element Level Directive

- **Attributes:**
  1. ng-model1
     - Provides two way binding between directive and controller for From field.
  2. ng-model2
     - Provides two way binding between directive and controller for To field.
  3. Submitted
     - Boolean value to perform all the validations.
  4. configJSON
     - Contains all the layout details for all the fields in the current page.
  5. disabledfield
     - To determine whether the field should be disabled.

2.3 DB JSON OBJECT STRUCTURE

FROM:

```
"busprtnr_inactiveDateFrom": {
    "type": "date",
    "label": "Inactive Date From",
    "name": "inactiveDateFrm",
    "show": true
},
```

TO:

```
"busprtnr_inactiveDateTo": {
    "type": "date",
    "label": "Inactive Date To",
    "name": "inactiveDateTo",
    "show": true
}
```

2.4 JSON OBJECT VARIABLE DETAILS

Variable Name	Accepts type	Explanation
type	String	Directive Type "date"
name	String	name for the date control which will be used for validations.
label	String	Display label for dropdown.
show	Boolean	Display or hide directive

options	Object	Which will contain the datasource and the displayfields
dataSource	String	Determines the list that needs to be shown to user
value	String	Value of the dropdown

2.5 Current issues and Addresses

- All console.log ( ) usages have been removed from the directive.
- Unused scope variables have been removed.
- Parent dependency have been removed with the usage of isolate scope.

2.6 Pros and Cons

Pros:

- Easy way of creating 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.