Angular JS Custom Directive for Number

(Reusable component for all the projects)

VERSION CONTROL

Prepared by:	Sindhu B S
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	Sindhu B S	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 NUMBER

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.

- projectNameNumber is a custom directive which allows users to enter only numbers into the field with 2 arrow buttons in the end for increasing and decreasing the value.
- The user can enter any number including negative numbers, floating- point numbers and exponential numbers.
- The values in the field will be controlled by the validations given to the particular field.

1.1 PURPOSE

- Restricting users to enter only numbers in a particular field.
- The reason behind using this directive and not a simple text field is that it does not need any explicit validations that would restrict alphabets or any other special characters.

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 Number 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 behind projectNameNumber very swiftly upon referring to 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

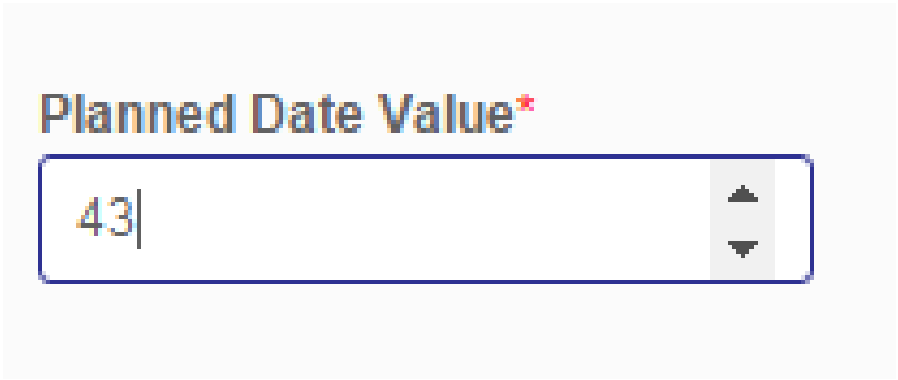
```
(function() {
    angular.module('projectNameUI.common.projectNameForm').directive('projectNameDynamicAttrNumber',projectNameDynamicAttrNu

  /* @ngInject */
    function projectNameDynamicAttrNumber($compile, isUndefinedOrNullValueService, SDFdefaultValueService){

        var directive = {
            restrict: 'E',
        templateUrl: 'app_modules/common/projectNameForm/directives/views/projectNameNumber.html',
            require: ['?^projectNameDynamicAttrTableUi' , '?^projectNameDynamicAttrGridUi'],
            scope: {
                model: '=ngModel',
                configJSON: '=configjson',
                field: '@field',
                disabledfield: '@disabledfield',
                changeFunction: '&ngChange',
                submitted: '=submitted',
                focus :'@'
            },
            link: function (scope, element, attrs, ctrl) {

            }
        };
        return directive;
    }
})();
```

1.5.2 HTML VIEW CODE

```
<div class="parent-class">
<input autocomplete="off" ng-show="itemInfo.show"
       type="number"
       ng-model="model"
       tooltip-enable="{{itemInfo.tooltip.showTooltip}}"
       uib-tooltip="{{displayTooltip}}" tooltip-trigger="mouseenter" tooltip-placement="{{itemInfo.tooltip.placement}}"
       name="{{name}}"
       ng-required="itemInfo.projectNameVal.required"
       min="{{itemInfo.projectNameVal.minValue}}"
       max="{{itemInfo.projectNameVal.maxValue}}"
       ng-minlength="itemInfo.projectNameVal.minLength"
       ng-maxlength="itemInfo.projectNameVal.maxLength"
       ng-pattern="{{itemInfo.projectNameVal.pattern}}"
       class="w100" ng-disabled="disabled"
       ng-class="{'submitted' : submitted}"
       ng-change="changeFunction()"
       ng-keypress="isNumberKey($event)"
       projectName-auto-focus="{{focusFirstInput}}" >
</div>
```

2. NUMBER DIRECTIVE INFORMATION USING ANGULAR JS

2.1 VIEW (SAMPLE)



2.2 GENERAL INFORMATION ABOUT THE DIRECTIVE

- **Directive Name:**
  projectName-dynamic-attr-number

- **Example:**
  <projectName-dynamic-attr-number field="Field Type" ng-model="dataJSON.value" ng-change="Change Function" configjson="configJSON" submitted = "submitted" ng-disabled = ”DisabledFlag”/>

NOTE: *All the values for the attributes are passed as string.*

- **Directive Type:**
  Element Level Directive

- **Attributes:**
  1. ng-model:
     - Provides two way binding between directive and controller.
  2. configJSON:
     - Contains all the layout details for all the fields in the current page.
  3. Field:
     - String value to get the layout out detail of that particular field.
  4. disabledfield
     - To determine whether the field should be disabled.
  5. submitted
     - Boolean value to perform all the validations.
  6. Change
     - Angular expression to be executed when input changes due to user interaction with the input element.

- **Location:**
  Directive:
  PROJECTNAME_CustApps_Web_UI_Client\app\app_modules\common\projectNameForm\directives\projectNameNumber.js
  View:
  PROJECTNAME_CustApps_Web_UI_Client\app\app_modules\common\projectNameForm\directives\views\projectNameNumber.html

2.3 DB JSON OBJECT STRUCTURE


```
" Number Field ": {
        "type": "Number",
        "label": "Number Field",
        "name": " Number Field ",
        "show": true,
        "showLabel": true,
        "projectNameVal": {
            "required": true,
            "maxLength": "3",
            "minLength" : "0",
            "maxValue": "5",
            "minValue": "-3"
        "errorMessage": {
            "required": "Required field",
            "maxLength": "Maximum 3 characters",
            "minLength": "Minimum 0 characters" ,
            "maxValue" : "Maximum value is 5" ,
             "minValue": "Minimum value is -3"
    }
  }
}
```

2.4 JSON OBJECT VARIABLE DETAILS

Variable Name	Accepts type	Explanation
type	String	Directive Type "Number"
name	String	name for the Number which will be used for validations.
label	String	Display label for Number.
show	Boolean	Display or hide directive
label	String	Label that should be shown to the user
showLabel	Boolean	Display or hide directive label
projectNameVal	Object	Contains required validation details
Required	Boolean	Determines whether the field is a mandatory field.
maxLength	String	Maximum length allowed for the field
minLength	String	Minimum length allowed for the field
maxValue	String	Maximum value allowed for the field

minValue	String	Minimum value allowed for the field
errorMessage	Object	Contains the errorMessages for the validations specified under projectNameVal

2.5 CURRENT ISSUES AND ADDRESSES

- Error Messages are passed to the directive from MongoDB so that they can be customized.
- All console.log ( ) usages have been removed from the directive.
- HTML view and the Directive have been separated.
- Unused scope variables have been removed.
- Parent dependency have been removed with the usage of isolate scope.
- Watch count have been reduced.

2.6 PROS AND CONS

Pros:

- Easy way of creating a number field with the required functionalities with minimal HTML code and minimal validations.
- 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.