Angular JS Custom Directive for File

(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 FILE 4

1 Introduction 4
1.1 Purpose 4
1.2 Supporting elements by angular js to create directive for Grid 4
1.3 Intended Audience 5
1.4 Defining a directive 5
1.5 SAMPLE CODE 6
1.5.1 ANGULAR DIRECTIVE CODE 6
1.5.2 HTML VIEW CODE 6

2. FILE DIRECTIVE INFORMATION USING ANGULAR JS 7

2.1 view (sample) 7
2.2 general information about the directive 7
2.3 DB JSON object structure 8
2.4 JSON object variable details 9
2.5 Current issues and Addresses 9

1. INTRODUCTION TO ANGULAR JS CUSTOM DIRECTIVE FOR FILE

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>file is a custom directive used for uploading files.

1.1 PURPOSE

- The <projectName>button is used to upload input documents or files.

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:
  - ✓ General users will use the <projectName>Button 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('<projectName>UI.common.<projectName>Form').directive('<projectName>DynamicAttrFile', <projectName>DynamicAttrFile);
    <projectName>DynamicAttrFile.$inject = ['$compile', 'multiFieldFactory', '$templateCache'];
    function <projectName>DynamicAttrFile($compile, multiFieldFactory, $templateCache) {
        // Changing required error message: To keep consistent required error message.
        var directive = {
            restrict: 'E',
            templateUrl:'<projectName>File.html',
            scope:{
                configJSON:'=configjson',
                change:'&',
                submitted:'='
            },
            link:function(scope,element,attrs,ctrl){

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

1.5.2 HTML VIEW CODE

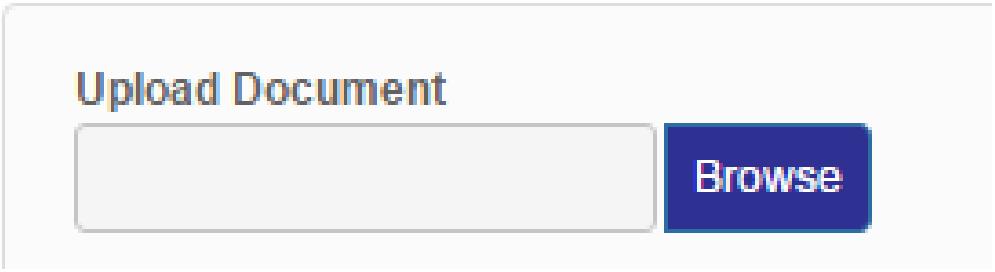
```
<div class="parent-class"><label>{{itemInfo.label}}<span class="mandatory"
                                                ng-if="itemInfo.<projectName>Val.required">*</span></label>

    <input class="fileUploadText" name="attrName" ng-required="itemInfo.<projectName>Val.required" tabindex="-1"
           ng-pattern="{{itemInfo.<projectName>Val.pattern}}" ng-disabled=true ng-model="itemInfo.fileText"
           type="text" ng-maxlength="{{itemInfo.<projectName>Val.maxLength}}"/>


    <div tabindex="0" class="fileUpload btn btn-primary" tooltip-enable="{{itemInfo.tooltip.showTooltip}}"
         uib-tooltip="{{displayTooltip}}" tooltip-trigger="mouseenter"
         tooltip-placement="{{itemInfo.tooltip.placement}}"><span>Browse</span>
        <input class="upload" ng-if="attrShow" type="file" tabindex="-1" ng-model="dataJSON[itemInfo.name]"
               ng-file-select="change()($files,fileFormat)" class="w100"></div>
 <span class="help-block" ng-if="itemInfo.desc">{{itemInfo.desc}}</span>
<span class="error err-required" ng-if="itemInfo.<projectName>Val.required">Required field</span>
<span class="error err-maxlength" ng-if="itemInfo.<projectName>Val.maxLength">{{itemInfo.<projectName>Val.errorMessage.maxLength}}</span
>
<span class="error err-pattern" ng-if="itemInfo.<projectName>Val.pattern">{{itemInfo.<projectName>Val.errorMessage.pattern}}</span>
<span class="error err-filetype" ng-if="fileType"> {{itemInfo.<projectName>Val.errorMessage.fileType}}</span>
<span class="error err-filesize" ng-if="fileSize && !fileType" >{{itemInfo.<projectName>Val.errorMessage.fileSize}}</span>
<span class="error err-customerror" ng-if="customError && !fileSize && !fileType" >{{itemInfo.<projectName>Val.errorMessage.custom}}</
span></div>
```

2. FILE DIRECTIVE INFORMATION USING ANGULAR JS

2.1 VIEW (SAMPLE)



2.2 GENERAL INFORMATION ABOUT THE DIRECTIVE

- **Directive Name:**
  <projectName>-dynamic-attr-file

- **Directive Type:**
  File Level Directive

- **Attributes:**

  There are 3 parameters needs to add to implement in <projectName>File

  a) **configjson**="configJSON"

  b) **Click :-**It is used to tell the directive what is the function name is used in controller .(Do not add curly braces in function name)

  c) **submitted**="submitted"

```
<<projectName>-dynamic-attr-file  field="Packing_List_File"
filename={{uploadfilename}} file-type={{typeError}} form="formName" custom-
error="{{custom}}"
        configjson="configJSON"  click="functionName" submitted="submitted" />
```

2.3 DB JSON OBJECT STRUCTURE


```
"Packing_List_File" : {
        "type" : "file",
        "label" : "Packing List File",
        "name" : "Packing_List_File",
        "show" : true,
        "fileText" : "packing",
        "link" : "onFileSelect($files,'application/x-zip-compressed')",
        "<projectName>Val" : {
          "required" : true,
          "pattern" : "/^[a-zA-Z0-9/.]*$/",
          "maxLength" : "30",
          "errorMessage" : {
            "required":"Required Field",
            "maxLength" : "Maximum 30 characters",
            "pattern" : "Alphanumeric  characters only",
            "fileType" : "Archive File Format (.zip) only",
            "fileSize" : ""
          }
        }
      }
```

2.4 JSON OBJECT VARIABLE DETAILS

Variable Name	Accepts type	Explanation
type	file	Directive Type "<projectName>button"
label	String	
name	String	Used in view page
show	Boolean	Display or hide
fileText	String	Default Text
Link	String	Function name
required	Boolean	
pattern	File pattern	
maxlength	Max length Accepted	

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.