Table of Contents

[Overview 2](#_Toc434837349)

[Prerequisites 2](#_Toc434837350)

[Features 2](#_Toc434837351)

[Getting Started 3](#_Toc434837352)

[Quick Start 3](#_Toc434837353)

[Wiring It Up 3](#_Toc434837354)

[Usage 3](#_Toc434837355)

[Sample app 6](#_Toc434837356)

[License 10](#_Toc434837357)

**Overview**

The library is built to ease one of the most trivial tasks involved in web development i.e. displaying form validation messages in the **AngularJS realm**. It is inspired by a number of form validations techniques available on the web and heavily borrows from **ng-messages** a library to facilitate the display of validations messages on the user screen.

**Prerequisites**

1. AngularJS 1.4.x and higher.
2. Angular- messages library 1.5.x +
3. A web server

**Features**

1. Works with any angularJS application above 1.4.x.
2. Easy to configure, requires only around 5 attributes to get started.
3. Works with multiple forms on a single page.
4. Provides option to display all the messages at the top of the page in Summary view or display each message beside the respective controls.
5. Provides focus back to control when clicked on error messages in summary view.
6. Provides option to control the visibility of messages as a whole from within controllers.
7. Provides real time feedback based on user input.

## Getting Started

### Quick Start

Include the **FormValidation.js** and the **angular-messages.js** file in the web application in the following manner

<head>

<script type="text/javascript" src="~/Scripts/app/FormValidation.min.js"></script>

<script type="text/javascript" src="~/Scripts/angular-messages.min.js"></script>

</head>

### Wiring It Up

1. Include the required library.
2. Ensure that you inject **‘FormValidations’** into your app by adding it to the dependency list.

angular.module("myApp", ['FormValidations']);

1. To set up a few initial values in the **app.js** section the following syntax is to be used.

angular.module("myApp", ['FormValidations'])

.config(['FormValidationsConfig', function (FormValidationsConfig) {

FormValidationsConfig.basePath = 'ValidationMsg';

FormValidationsConfig.displayOption = 'Item';

}])

The explanation for **basePath** and **displayOption** is written in the following section

## Usage

The directive is to be used on **FORM** tag in the following manner

<form name="formDemo"

role="form"

validate-form

resource-path="Demo"

form-visibility-map="formMap"

display-option="Item">

validate-form

This directive is to be placed on the **form tag**. Without this directive none of the features of the FormValidations library would work.

resource-path

This attribute serves as an input parameter to the validate-form directive, it is the path of the .json file where validation messages are stored as key value pairs.

Eg. <input type="text" class="form-control" name="Name" ng-model="Demo.Name" placeholder="Enter Name" ng-required="true">

To display validation messages for this control, the json file must have a key values pair in the following manner ie “<controlName>\_<validationName> “: “Message”, where <controlName> is the name property on the control and the< validationName> is the one that gets registered with the $validators on the control.

NOTE: The keys are case sensitive and should be written ONLY in the following format

{

"Name\_required": "Please provide a Name"

}

If the base path is not set in the **.config** section on the app then the full path needs to be set here. The .json extension can be ignored, though its presence won`t change anything.

resource-path="/ValidationMsg/Demo" – Take special notice of the leading **“/”** if the full path is provided.

form-visibility-map

This is an **optional attribute**, it accepts an array on the $scope object in the following format.

scope.formMap = [{ "name": "formDemo", "visible": true }];

where “name” is the name of the form. It can be useful in the following

1. When the visibility of all messages associated with the form needs to be controlled from within the controller.
2. In case of multiple forms, the “visble” property can decide the messages associated with which forms needs to display and which ones need to be hidden.

Eg. scope.formMap = [{ "name": "formDemo", "visible": true },

{ "name": "formDemo1", "visible": true },

{ "name": "formDemo2", "visible": false }];

Apart from declaring the visibility map up front, the visibility can also be changed on user or system events. e.g.

scope.ButtonClicked = function () {

scope.formMap[2].visible = true;

};

If the 2 scenarios mentioned are not required, then this **attribute can be left out** from the form tag.

display-option

This is also an **optional attribute**, where the valid values are “Summary” and “Item”. This helps in deciding whether all the messages are to be shown at the top of the form or each message to be shown beside or below the respective controls.

This option can also be set in the .config section, if not set at both the places it would default to “Summary” view and the option in the form tag will override what is provided in the config section.

1. Summary: When using this option the following directive needs to be placed at the desired section based on the design. This directive can also exist on some other view (In case of partial views), but the only mandatory point would be that the partial view loads in the current **route context. When you click on the error message the focus will be on the control.**

<render-validation-messages></render-validation-messages>

Or

<div render-validation-messages></div>

1. Item: When using this option the following directive needs to be placed within the form element at the desired section based on the form design. The validation-for directive accepts a parameter which is the name of the control whose validation messages are to be displayed in the particular section.

<div class="input-group">

<input type="text"

class="form-control"

name="Name"

ng-model="Demo.Name"

placeholder="Enter Name"

ng-required="true"

ng-maxlength="10">

**<div validation-for="Name"></div>**

</div>

## Sample app

The sample for each component described above is compiled into a sample. The example is just for demo purposes and does not necessarily follow the best practices.

Declaring Angular module and controller

<script type="text/javascript">

angular.module("demo", ['FormValidations'])

.config(['FormValidationsConfig', function (FormValidationsConfig) {

debugger;

FormValidationsConfig.basePath = 'ValidationMsg';

FormValidationsConfig.displayOption = 'Summary';

}]).controller('DemoCtrl', ['$scope', function (scope) {

scope.Demo = { "Name": "", "Email": "", "Comments": "" };

scope.formMap = [{ "name": "formDemo", "visible": true }];

scope.ButtonClicked = function () {

alert("clicked");

};

}]);

</script>

The HTML markup

<!DOCTYPE html>

<html ng-app="demo">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Form Validation Demo</title>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.7/angular.js"></script>

<script type="text/javascript" src="~/Scripts/app/FormValidation.min.js"></script>

</head>

<body ng-controller="DemoCtrl">

<div class="container">

<div class="page-header">

<h1>Registration form <small>A form validation demo with basic fields</small></h1>

</div>

<!-- Registration form - START -->

<div class="container">

<div style="color:red;" render-validation-messages></div>

<div class="row">

<form name="formDemo" role="form"

validate-form

novalidate

resource-path="Demo"

form-visibility-map="formMap"

display-option="Summary">

<div class="col-lg-6">

<div class="form-group">

<label for="Name">Enter Name</label>

<div class="input-group">

<input type="text"

class="form-control"

name="Name"

ng-model="Demo.Name"

placeholder="Enter Name"

ng-required="true"

ng-maxlength="10">

<div style="color:red;" validation-for="Name"></div>

</div>

</div>

<div class="form-group">

<label for="Email">Enter Email</label>

<div class="input-group">

<input type="email"

class="form-control"

name="Email"

ng-model="Demo.Email"

placeholder="Enter Email"

ng-required="true">

<div style="color:red;" validation-for="Email"></div>

</div>

</div>

<div class="form-group">

<label for="Comments">Comments</label>

<div class="input-group">

<textarea name="Comments"

ng-model="Demo.Comments"

class="form-control"

rows="5"

ng-maxlength="30">

</textarea>

<div style="color:red;" validation-for="Comments"></div>

</div>

</div>

<button type="submit" ng-click=ButtonClicked() class="btn btn-info pull-left">Submit</button>

</div>

</form>

</div>

</div>

<!-- Registration form - END -->

</div>

</body>

</html>

Json File

{

"Name\_required": "Please provide a Name",

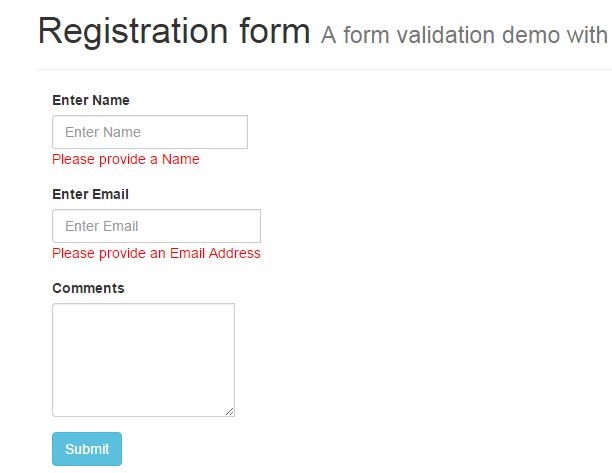
"Name\_maxlength": "Name cannot exceed more than 10 characters",

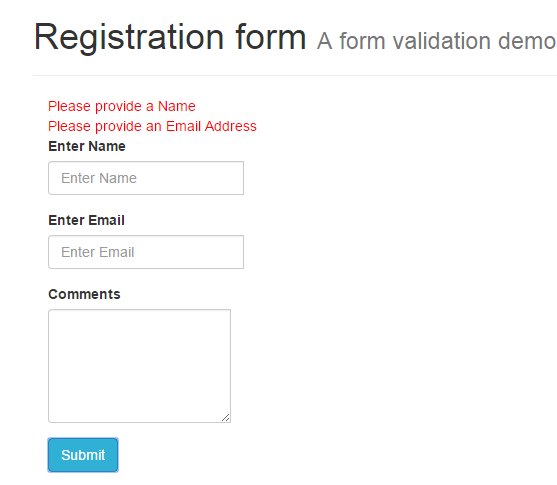
"Comments\_maxlegth": "Comments cannot exceed more than 30 characters",

"Email\_required": "Please provide an Email Address",

"Email\_email" : "Please provide a valid Email Address"

}





## License

The MIT License (MIT)

Copyright (c) 2015 Sumanth Sampath

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.