

**AngularJS – MVC.Net**

---

**AngularJs – MVC.Net**

**Version No.: 1.0**

**Date of Release: 07/07/2014**

**AngularJS – MVC.Net**

---

***REVISION HISTORY***

---

Date	Version No.	Summary of Changes
07/07/2014	1.0	Initial Version

## TABLE OF CONTENTS

---

Summary of Changes .....	2
<b>1.0.....INTRODUCTION</b>	<b>4</b>
<b>2.0.....ANGULARJS - CONTROL .....</b>	<b>5</b>
<b>2.1.....EXAMPLE 1 - MODAL AND READ-ONLY DISPLAY WITH {{}} .....</b>	<b>5</b>
<b>2.2.....EXAMPLE 2 - MODAL AND READ-ONLY DISPLAY WITH BINDINGS.....</b>	<b>6</b>
<b>2.3.....EXAMPLE 3 - READ-ONLY DISPLAY AND FORMATTING .....</b>	<b>7</b>
<b>2.4.....EXAMPLE 4 – INPUT-TEXT AND NG-MODLE .....</b>	<b>8</b>
<b>2.5.....EXAMPLE 5 - INPUT-DATE AND NG-MODLE .....</b>	<b>10</b>
<b>2.6.....EXAMPLE 6 - INPUT-DATETIME AND NG-MODLE .....</b>	<b>12</b>
<b>2.7.....EXAMPLE 7 - INPUT-MONTH AND NG-MODLE .....</b>	<b>13</b>
<b>2.8.....EXAMPLE 8 - INPUT-TIME AND NG-MODLE .....</b>	<b>14</b>
<b>2.9.....EXAMPLE 9 - INPUT-EMAIL AND NG-MODLE.....</b>	<b>16</b>
<b>2.10 ..EXAMPLE 10 - INPUT-CHECKBOX AND NG-MODLE .....</b>	<b>17</b>
<b>2.11 ..EXAMPLE 11 - INPUT-RADIO AND NG-MODLE .....</b>	<b>18</b>
<b>2.12 ..EXAMPLE 12 - INPUT-URL AND NG-MODLE .....</b>	<b>20</b>
<b>3.0.....ANGULARJS - CONTROLLER .....</b>	<b>20</b>
<b>3.1.....EXAMPLE 13 - NG-CONTROLLER BASIC .....</b>	<b>21</b>
<b>3.2.....EXAMPLE 14 - NG-CONTROLLER AND LIST BASIC .....</b>	<b>22</b>
<b>3.3.....EXAMPLE 15 - NG-CONTROLLER AND LIST BASIC - ADD.....</b>	<b>23</b>
<b>3.4.....EXAMPLE 16 - NG-CONTROLLER AND LIST BASIC - REMOVE.....</b>	<b>25</b>
<b>3.5.....EXAMPLE 17 - NG-CONTROLLER AND OBJECT LIST BASIC – ADD/REMOVE</b>	<b>27</b>
<b>3.6.....EXAMPLE 18 - NG-CONTROLLER AND OBJECT LIST BASIC – FILTER .....</b>	<b>30</b>
<b>3.7.....EXAMPLE 19 - NG-CONTROLLER AND OBJECT LIST BASIC – ORDERBY... </b>	<b>33</b>
<b>3.8.....EXAMPLE 20 - NG-CONTROLLER AND OBJECT LIST BASIC – ORDERBY DYNAMIC</b>	<b>35</b>

## AngularJS – MVC.Net

---

### 1.0 Introduction

AngularJS is one of the most prominent technology for web applications. Most of the developers use SPA(Single page Application) development when they use AngularJS in their projects.

Before going ahead to discover this in detail, we will understand some basic points for this.

- AngularJS was originally developed in 2009 by Miško Hevery and Adam Abrons[10] at Brat Tech LLC.
- This is maintain by Google.
- As of July 6, 2015, release 1.4.2 (code name nebular-readjustment) is the current stable version.
- AngularJS supports following browsers.

Framework	AngularJS
Internet Explorer	8+ (9+)
Mozilla Firefox	4+
Safari	5+
Opera	11+
Chrome	30+

- AngularJS is open source and MIT\* license.
- AngularJS is develop on JavaScript language.
- AngularJS is a HTML enhanced for Web Apps.
- It is a framework for HTML/Web application.
- It includes web components and model driven view.
- It is a databinding framework.
- It is develop for Single Page Application.
- Model contents only data no behavior.
- Controller is set of functions which have behavior for Application.
- AngularJS provides 3 important things.
  - Routing – Updating page based on URL.
  - Templating –
  - Data binding – It maintain synchronization between Model and View.

\*The MIT License is a free software license originating at the Massachusetts Institute of Technology (MIT).[1] It is a permissive free software license, meaning that it permits reuse

## AngularJS – MVC.Net

within proprietary software provided all copies of the licensed software include a copy of the MIT License terms and the copyright notice

## 2.0 AngularJS - Control

To understand AngularJS Control, we will use example from basics to enhance.

### 2.1 Example 1 - Modal and Read-only display with {{}}

Following Example shows basic of AngularJS example. In this example, we have created basic page, which has one Textbox, and Key-in Values are getting render by AngularJS on page.

Followings needs to understand in this example.

1. ng-app :-> The ngApp directive designates the root element of the application and is typically placed near the root element of the page - e.g. on the <body> or <html> tags.
2. Script Src:-> We can add AngularJS library Via two methods.
  - a. CDN server
  - b. Local Server

In this both path shows about CDN server.

3. ng-model:-> The ng-model directive binds an input, select, textarea to a property.
  - a. It is a two-way binding.
4. Double Curly braces {{ }}:-> This helps to render/display values on HTML Page. It is a one-way binding.

#### Modal and display

```
<html ng-app>
  <head>
    <meta charset="utf-8">
    <title>Angular.js Example</title>
    <!--<script
src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.3.14/angular.min.js"></script-->
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
  </head>
  <body>
    Name:<input ng-model="name" type="text"/>
    Hello {{name}}
  </body>
</html>
```

## AngularJS – MVC.Net

Name:  Hello Abhay

## 2.2 Example 2 - Modal and Read-only display with Bindings

Following Example shows basic of AngularJS example and continuation of above example 1. In this example, we have created basic page, which has one Textbox, and Key-in Values are getting render by AngularJS on page.

Followings needs to understand in this example.

1. Ng-bind:-> The ng-bind attribute tells Angular to replace the text content of the specified HTML element with the value of a given expression.
  - a. It is preferable to use ng-bind instead of {{ expression }}.
  - b. It can use by span, Div, P etc tags.
  - c. Ng-bind is a one-way binding.

### Modal and Read-only display with Bindings

```
<html ng-app>
  <head>
    <meta charset="utf-8">
    <title>Angular.js Example</title>
    <!--<script
src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.3.14/angular.min.js"></script>-->
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>

  </head>
  <body>
    Name:<input ng-model="name" type="text"/>
    Hello <span ng-bind="name"></span>
  </body>
</html>
```

Name:  Hello Abhay

## AngularJS – MVC.Net

## 2.3 Example 3 - Read-only display and Formatting

Following Example shows basic of AngularJS example for formatting of data and. In this example, we have created basic page, which has one Textbox, and Key-in Values are getting render by AngularJS on page with formatting.

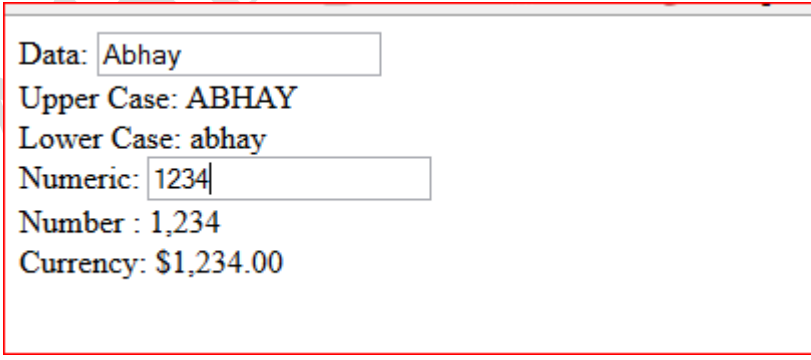
Followings needs to understand in this example.

1. There are multiple filters available with AngularJS, We have shown following of them in this example.
  - a. Uppercase:-> Convert text into upper text.
  - b. Lowercase:-> Convert text into lower text.
  - c. Number:-> Display Number into Numeric format.
  - d. Currency:-> Display Number in Currency format.
2. “|” operator is used to apply filter with ng-model.

### Read-only display and Formatting

```
<html ng-app>
<head>
  <meta charset="utf-8">
  <title>Angular.js Example</title>
  <!--<script
src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.3.14/angular.min.js"></script-->
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>

</head>
<body>
  Data: <input ng-model="data" type="text"/> <BR>
  Upper Case: <span ng-bind="data | uppercase"></span> <BR>
  Lower Case: <span ng-bind="data | lowercase"></span> <BR>
  Numeric: <input ng-model="numericData" type="text"/> <BR>
  Number : <span ng-bind="numericData | number "></span> <BR>
  Currency: <span ng-bind="numericData | currency"></span>
</body>
</html>
```



Data: Abhay  
Upper Case: ABHAY  
Lower Case: abhay  
Numeric: 1234  
Number : 1,234  
Currency: \$1,234.00

## AngularJS – MVC.Net

## 2.4 Example 4 – Input-Text and ng-modle

Following Example shows basic of AngularJS example for formatting of data and. In this example, we have created basic page, which has one Textbox, and Key-in Values are getting render by AngularJS on page with formatting.

Followings needs to understand in this example.

1. There are tags available for validation of input textbox..
  - a. ng-pattern:-> It sets pattern validation error key if the ng-model value does not match a RegExp found by evaluating the Angular expression given in the attribute value.
  - b. ng-show:-> The ngShow directive shows or hides the given HTML element based on the expression provided to the ngShow attribute.
  - c. \$error:-> Is an object hash, containing references to controls or forms with failing validators, where: keys are validation tokens (error names).
    - i. It is follow by “Name” attribute and refer validation added for attribute.
  - d. ng-required:-> It sets required validation error key if the value is not entered.
  - e. ng-minlength:-> It sets minlength validation error key if the value is shorter than minlength.
  - f. ngMaxlength:-> It sets maxlength validation error key if the value is longer than maxlength. Setting the attribute to a negative or non-numeric value, allows view values of any length.
  - g. ngTrim:-> If set to false Angular will not automatically trim the input..
    - i. The default value of this is true.
2. “||” operator is used to add “OR” operation between two Boolean values.
3. “&&” operator is used to add “AND” operation between two Boolean values.

### Input-Text and ng-modle

```
<html>
<head>
  <meta charset="utf-8">
  <title>Angular.js Example</title>
  <!--<script
src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.3.14/angular.min.js"></script>-->
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
</head>
<body>

  <form name="myform" ng-app>
    Number:
    <input type="Text"
      name="numberDataName"
      id="numberDataID"
      ng-model="numberData"
      require
      ng-pattern="/^\d{0,9}(\.\d{1,9})?$/">
    <span ng-show="myform.numberDataName.$error.pattern">Not valid
```



## AngularJS – MVC.Net

```

number!</span><br>
Number key-in is <span ng-bind="numberData"></span>.<br>
Invalid Number:
<input type="Text"
  name="numberIDataName"
  id="numberIDataID"
  ng-model="numberIData"
  require
  ng-pattern="/^\d{0,9}(\.\d{1,9})?$/"/>
  <span ng-show="myform.numberIDataName.$error.pattern">Not valid
number!</span><br>
Number key-in is <span ng-bind="numberIData"></span>.<br>
Min and Max Number:
<input type="Text"
  name="numbermDataName"
  id="numbermDataID"
  ng-model="numbermData"
  ng-required="true"
  ng-minlength="2"
  ng-maxlength="5">
  <span ng-show="myform.numbermDataName.$error.required">It is require.</span>
  <span ng-show="myform.numbermDataName.$error.minlength ||
myform.numbermDataName.$error.maxlength">Please add string with length of 2-
5!</span><br>
Min and Max key-in is <span ng-bind="numbermData"></span>.<br>
Trim String:
<input type="Text"
  name="stringDataName"
  id="stringDataID"
  ng-model="stringData"
  require
  ng-trim="true">
<br>
String key-in is <span ng-bind="stringData"></span>.<br>

Not Trim String:
<input type="Text"
  name="stringntDataName"
  id="stringntDataID"
  ng-model="stringntData"
  require
  ng-trim="false">
<br>
String key-in is <span ng-bind="stringntData"></span>.<br>

</form>

</body>
</html>

```

## AngularJS – MVC.Net

Number:   
 Number key-in is .  
 InValid Number:   
 Number key-in is .  
 Min and Max Number:  It is require.  
 Min and Max key-in is .  
 Trim String:   
 String key-in is .  
 Not Trim String:   
 String key-in is .

Number:   
 Number key-in is 1234.  
 InValid Number:  Not valid number!  
 Number key-in is .  
 Min and Max Number:  Please add string with length of 2-5!  
 Min and Max key-in is .  
 Trim String:   
 String key-in is Abhay.  
 Not Trim String:   
 String key-in is Abhay .

## 2.5 Example 5 - Input-Date and ng-modle

Following Example shows basic of AngularJS example for formatting of data and. In this example, we have created basic page, which has one Textbox for Date, and Key-in Values are getting render by AngularJS on page with formatting.

Followings needs to understand in this example.

1. Date is specific type of input and it supports following tag.
  - a. min:-> It sets the min validation error key if the value entered is less than min. This must be a valid ISO date string (yyyy-MM-dd).
  - b. Max:-> It sets the max validation error key if the value entered is greater than max. This must be a valid ISO date string (yyyy-MM-dd).

### Input-Date and ng-modle

```
<html>
```

## AngularJS – MVC.Net

```

<head>
  <meta charset="utf-8">
  <title>Angular.js Example</title>
  <!--<script
src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.3.14/angular.min.js"></script>-->
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>

</head>
<body>

  <form name="myform" ng-app>
    Date: <input type="date" ng-model="dateData" placeholder="yyyy-MM-dd" /> <br>
    Date Key-in is <span ng-bind="dateData"></span>.<br />

    InValid Date: <input type="date" name="dateRangeDataName" ng-
model="dateRangeData" placeholder="yyyy-MM-dd" min="2015-01-01" max="2015-12-31" />
<span ng-show="myform.dateRangeDataName.$error.min ||
myform.dateRangeDataName.$error.max">Please key-in date of year 2015!</span><br>
    Date Key-in is <span ng-bind="dateRangeData"></span>.<br />

    Require Date: <input type="date" name="dateRiqureDataName" ng-
model="dateRangeData" ng-required="true" placeholder="yyyy-MM-dd" min="2015-01-01"
max="2015-12-31" /> <span ng-show="myform.dateRiqureDataName.$error.required">Please
key-in Date!</span><br>
    Date Key-in is <span ng-bind="dateRangeData"></span>.<br />
  </form>
</body>
</html>

```

Date:   
 Date Key-in is .  
 InValid Date:   
 Date Key-in is .  
 Require Date:  Please key-in Date!  
 Date Key-in is .

Date:   
 Date Key-in is 12-31-2015.  
 InValid Date:  Please key-in date of year 2015!  
 Date Key-in is .  
 Require Date:   
 Date Key-in is Thu Dec 31 2015 00:00:00 GMT-0500 (Eastern Daylight Time).

## AngularJS – MVC.Net

## 2.6 Example 6 - Input-DateTime and ng-model

Following Example shows basic of AngularJS example for formatting of data and. In this example, we have created basic page, which has one Textbox for DateTime, and Key-in Values are getting render by AngularJS on page with formatting.

This is more or less same as Input-Date, followings needs to understand in this example.

1. DateTime is specific type of input and it supports following tag.
  - a. min:-> It sets the min validation error key if the value entered is less than min. This must be a valid ISO date string (yyyy-MM-dd).
  - b. Max:-> It sets the max validation error key if the value entered is greater than max. This must be a valid ISO date string (yyyy-MM-dd).

### Input-DateTime and ng-model

```
<html>
<head>
  <meta charset="utf-8">
  <title>Angular.js Example</title>
  <!--<script
src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.3.14/angular.min.js"></script>-->
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
</head>
<body>

  <form name="myform" ng-app>
    Date: <input type="datetime-local" ng-model="dateData" placeholder="yyyy-MM-dd" /> <br>
    Date Key-in is <span ng-bind="dateData | date:'MM-dd-yyyy'"></span>.<br />

    InValid Date: <input type="datetime-local" name="dateRangeDataName" ng-model="dateRangeData" placeholder="yyyy-MM-dd" min="2015-01-01T12:00" max="2015-12-31T12:00" /> <span ng-show="myform.dateRangeDataName.$error.min || myform.dateRangeDataName.$error.max">Please key-in date of year 2015 between(2015-01-01T12:00 and 2015-12-31T12:00)!</span><br>
    Date Key-in is <span ng-bind="dateRangeData"></span>.<br />

    Require Date: <input type="datetime-local" name="dateRiqureDataName" ng-model="dateRiqureData" ng-required="true" placeholder="yyyy-MM-dd" min="2015-01-01T12:00" max="2015-12-31T12:00" /> <span ng-show="myform.dateRiqureDataName.$error.required">Please key-in Date!</span><br>
    Date Key-in is <span ng-bind="dateRiqureData"></span>.<br />
  </form>
</body>
</html>
```

## AngularJS – MVC.Net

Date:   
 Date Key-in is .  
 InValid Date:   
 Date Key-in is .  
 Require Date:  Please key-in Date!  
 Date Key-in is .

Date:   
 Date Key-in is 12-31-2015.  
 InValid Date:  Please key-in date of year 2015 between(2015-01-01T12:00 and 2015-12-31T12:00)!  
 Date Key-in is .  
 Require Date:   
 Date Key-in is Thu Dec 31 2015 12:00:00 GMT-0500 (Eastern Daylight Time).

## 2.7 Example 7 - Input-Month and ng-modle

Following Example shows basic of AngularJS example for formatting of data and. In this example, we have created basic page, which has one Textbox for Month, and Key-in Values are getting render by AngularJS on page with formatting.

This is more or less same as Input-Date, followings needs to understand in this example.

2. month is specific type of input and it supports following tag.
  - a. min:-> It sets the min validation error key if the value entered is less than min. This must be a valid ISO month string (yyyy-MM).
  - b. Max:-> It sets the max validation error key if the value entered is greater than max. This must be a valid ISO month string (yyyy-MM).

### Input-month and ng-modle

```
<html>
<head>
  <meta charset="utf-8">
  <title>Angular.js Example</title>
  <!--<script
src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.3.14/angular.min.js"></script-->
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
</head>
<body>
```

## AngularJS – MVC.Net

```

<form name="myform" ng-app>
  month: <input type="month" ng-model="dateData" placeholder="yyyy-MM" /> <br>
  month Key-in is <span ng-bind="dateData | date:'MM-yyyy'"></span>.<br />

  InValid month: <input type="month" name="dateRangeDataName" ng-
model="dateRangeData" placeholder="yyyy-MM" min="2015-01" max="2015-12" /> <span ng-
show="myform.dateRangeDataName.$error.min ||
myform.dateRangeDataName.$error.max">Please key-in month of year 2015!</span><br>
  month Key-in is <span ng-bind="dateRangeData"></span>.<br />

  Require month: <input type="month" name="dateRiqureDataName" ng-
model="dateRiqureData" ng-required="true" placeholder="yyyy-MM" min="2015-01"
max="2015-12" /> <span ng-show="myform.dateRiqureDataName.$error.required">Please
key-in month!</span><br>
  month Key-in is <span ng-bind="dateRiqureData"></span>.<br />
</form>
</body>
</html>

```

month: yyyy-MM  
month Key-in is .  
InValid month: yyyy-MM  
month Key-in is .  
Require month: yyyy-MM Please key-in month!  
month Key-in is .

month: 2015-06  
month Key-in is 06-2015.  
InValid month: 2014-12 Please key-in month of year 2015!  
month Key-in is .  
Require month: 2015-07  
month Key-in is Wed Jul 01 2015 00:00:00 GMT-0400 (Eastern Standard Time).

## 2.8 Example 8 - Input-Time and ng-modle

Following Example shows basic of AngularJS example for formatting of data and. In this example, we have created basic page, which has one Textbox for time, and Key-in Values are getting render by AngularJS on page with formatting.

## AngularJS – MVC.Net

This is more or less same as Input-Date, followings needs to understand in this example.

1. time is specific type of input and it supports following tag.
  - a. min:-> It sets the min validation error key if the value entered is less than min. This must be a valid ISO time string (HH:mm:ss).
  - b. Max:-> It sets the max validation error key if the value entered is greater than max. This must be a valid ISO time string (HH:mm:ss).

### Input-time and ng-model

```
<html>
<head>
  <meta charset="utf-8">
  <title>Angular.js Example</title>
  <!--<script
src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.3.14/angular.min.js"></script>-->
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
</head>
<body>

  <form name="myform" ng-app>
    Time: <input type="time" ng-model="timeData" placeholder="HH:mm:ss" /> <br>
    Time Key-in is <span ng-bind="timeData | date:'HH:mm'"></span>.<br />

    InValid Time: <input type="time" name="timeRangeDataName" ng-
model="timeRangeData" placeholder="HH:mm:ss" min="09:00:00" max="17:30:00" /> <span
ng-show="myform.timeRangeDataName.$error.min ||
myform.timeRangeDataName.$error.max">Please key-in time between(09:00:00 and
17:30:00)!</span><br>
    Time Key-in is <span ng-bind="timeRangeData"></span>.<br />

    Require Time: <input type="time" name="timeRiqureDataName" ng-
model="timeRiqureData" ng-required="true" placeholder="HH:mm:ss" min="09:00:00"
max="17:30:00" /> <span ng-show="myform.timeRiqureDataName.$error.required">Please
key-in time!</span><br>
    Time Key-in is <span ng-bind="timeRiqureData"></span>.<br />
  </form>
</body>
</html>
```

Time:

Time Key-in is .

InValid Time:

Time Key-in is .

Require Time:  Please key-in time!

Time Key-in is .

## AngularJS – MVC.Net

Time:   
 Time Key-in is 09:00.  
 InValid Time:  Please key-in time between(09:00:00 and 17:30:00)!  
 Time Key-in is .  
 Require Time:   
 Time Key-in is Thu Jan 01 1970 09:00:00 GMT-0500 (Eastern Daylight Time).

## 2.9 Example 9 - Input-Email and ng-modle

Following Example shows basic of AngularJS example for formatting of data and. In this example, we have created basic page, which has one Textbox for Email, and Key-in Values are getting render by AngularJS on page with formatting.

This is more or less same as Input-Email, followings needs to understand in this example.

2. Email is specific type of input and it supports tags as Textbox.

### Input-Email and ng-modle

```
<html>
<head>
  <meta charset="utf-8">
  <title>Angular.js Example</title>
  <!--<script
src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.3.14/angular.min.js"></script>-->
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
</head>
<body>

  <form name="myform" ng-app>
    Email: <input type="email" ng-model="emailData" /> <br>
    month Key-in is <span ng-bind="emailData"></span>.<br />

    InValid Email: <input type="email" name="emailDataName" ng-model="emailIData"
/> <span ng-show="myform.emailDataName.$error.email">Please key-in proper
email!</span><br>
    month Key-in is <span ng-bind="emailIData"></span>.<br />

    Require Email: <input type="email" name="emailRiqureDataName" ng-
model="emailRiqureData" ng-required="true" /> <span ng-
show="myform.emailRiqureDataName.$error.required">Please key-in email!</span><br>
    Email Key-in is <span ng-bind="emailRiqureData"></span>.<br />
  </form>
</body>
</html>
```



## AngularJS – MVC.Net

Email:

month Key-in is .

InValid Email:

month Key-in is .

Require Email:  Please key-in email!

Email Key-in is .

Email:

month Key-in is a@a.com.

InValid Email:  Please key-in proper email!

month Key-in is .

Require Email:  Please key-in email!

Email Key-in is .

## 2.10 Example 10 - Input-Checkbox and ng-modle

Following Example shows basic of AngularJS example checkbox values. In this example, we have created basic page, which has checkbox and values are getting render by AngularJS on page with formatting.

1. Checkbox is specific type of input and it supports following tags.
  - a. ng-false-value:-> The value to which the expression should be set when not selected.
  - b. ng-true-value:-> The value to which the expression should be set when selected.

### Input-Checkbox and ng-modle

```
<html>
<head>
  <meta charset="utf-8">
  <title>Angular.js Example</title>
  <!--<script
src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.3.14/angular.min.js"></script-->
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
```

## AngularJS – MVC.Net

```

</head>
<body>

  <form name="myform" ng-app>
    Checkbox: <input type="checkbox" ng-model="checkboxData" /> <br>
    Value is <span ng-bind="checkboxData"></span>.<br />

    Checkbox Checked Values: <input type="checkbox"
name="checkboxValueCheckedData" ng-true-value="'YES'" ng-false-value="'NO'" ng-
model="checkboxValueCheckedData" /> <br>
    Checkbox Checked Values is <span ng-
bind="checkboxValueCheckedData"></span>.<br />

    Checkbox Un-Checked Values: <input type="checkbox"
name="checkboxValueUnCheckedData" ng-true-value="'YES'" ng-false-value="'NO'" ng-
model="checkboxValueUnCheckedData" /> <br>
    Checkbox Un-Checked Values is <span ng-
bind="checkboxValueUnCheckedData"></span>.<br />

  </form>
</body>
</html>

```

Checkbox: ☒  
 Value is true.  
 Checkbox Checked Values: ☒  
 Checkbox Checked Values is YES.  
 Checkbox Un-Checked Values: ☐  
 Checkbox Un-Checked Values is NO.

## 2.11 Example 11 - Input-radio and ng-model

Following Example shows basic of AngularJS example for radio control. In this example, we have created basic page, which has radio and values are getting render by AngularJS on page with formatting.

1. Radio is specific type of input and it supports following tags.
  - a. Value:-> The value to which the ng-model expression should be set when selected. Note that value only supports string values.
  - b. ng-value:-> Angular expression to which ngModel will be set when the radio is selected. Should be used instead of the value attribute if you need a non-string ngModel (boolean, array, ...).

## AngularJS – MVC.Net

**Input-Radio and ng-model**

```
<html>
<head>
  <meta charset="utf-8">
  <title>Angular.js Example</title>
  <!--<script
src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.3.14/angular.min.js"></script>-->
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
</head>
<body>

  <form name="myform" ng-app>
    Please select color: <br>
    <input type="radio"    ng-model="colorData"    value="Blue"    /> Blue <br>
    <input type="radio"    ng-model="colorData"    value="Red"     /> Red <br>
    <input type="radio"    ng-model="colorData"    ng-value="otherColor" /> Other
  <br>
    <input type="TextBox"  ng-model="otherColor"   /> Other <br>
    color is <span ng-bind="colorData"></span>.<br />
  </form>
</body>
</html>
```

Please select color:

☒ Blue☐ Red☐ Other

Other

color is Blue.

Please select color:

☐ Blue☐ Red☒ Other

Other

color is Skyblue.

## AngularJS – MVC.Net

---

### 2.12 Example 12 - Input-URL and ng-modle

Following Example shows basic of AngularJS example for URL control. In this example, we have created basic page, which has URL and values are getting render by AngularJS on page with formatting.

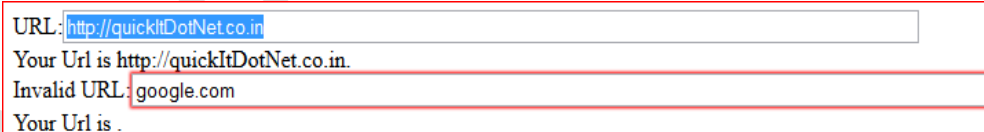
1. This is same as Textbox control , but it can contain only url's .
2. Text input with URL validation. Sets the url validation error key if the content is not a valid URL.

#### Input-URL and ng-modle

```
<html ng-app>
<head>
  <meta charset="utf-8">
  <title>Angular.js Example</title>
  <!--<script
src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.3.14/angular.min.js"></script-->
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>

</head>
<body>
  URL:<input ng-model="urlData" type="URL" size="100"/><BR />
  Your Url is {{urlData}}.<BR />

  Invalid URL:<input ng-model="urlIData" type="URL" size="100"/><BR />
  Your Url is {{urlIData}}.
</body>
</html>
```



The screenshot shows the rendered HTML output of the AngularJS application. It displays two input fields for URLs. The first field is labeled 'URL:' and contains the text 'http://quickitDotNet.co.in'. Below it, the text 'Your Url is http://quickitDotNet.co.in.' is shown. The second field is labeled 'Invalid URL:' and contains the text 'google.com'. Below it, the text 'Your Url is .' is shown, indicating that the input is not a valid URL.

### 3.0 AngularJS - Controller

In AngularJS, a Controller is a JavaScript constructor function that is use to control AngularJS applications.

## AngularJS – MVC.Net

The ng-controller directive defines the application controller. AngularJS use controllers to set up the initial state of the \$scope object and add behavior to the \$scope object. In addition, there are multiple objects associated with controller, which we will explore one by one.

### 3.1 Example 13 - ng-controller Basic

Following Example shows basic of AngularJS controller. As this is JavaScript function, we have to work with this in script block.

Followings needs to understand in this example.

1. angular.module:-> This is the key of AngularJS and It contains different part of app like controllers, services, filters, directives, etc.  
This block get hold of application and responsible for block, which is assign to it.

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

2. Controller:-> Once, we get reference of module of Application. We can add one or more controller into it. Each controller has its own scope.
3. Scope:-> Scope is an object that refers to the application model. It is an execution context for expressions. Scopes are arranged in hierarchical structure, which mimic the DOM structure of the application. Scopes can watch expressions and propagate events.
  - a. Property: Scope could be followed by property, which can be bind to control as one-way or Two-way bindings.
  - b. Function:-> Scope could be followed by method, which can be bind to control as one-way.
  - c. Events:-> Scope could be followed by Event( method only), which can be mapped to control as event.
4. Ng-click:-> This is same as HTML click event followed by AngularJS Event.

In following Example , FirstName and lastName are properties. FullName is method and ng-click is "alertName"

#### ng-controller Basic

```
<html>
  <head>
    <meta charset="utf-8">
    <title>Angular.js Example</title>
    <!--<script
src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.3.14/angular.min.js"></script>-->
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
    <script>
      var app = angular.module('myApp', []);
      app.controller('EmployeeCtrl', function ($scope) {
        $scope.firstName = "";
        $scope.lastName = "";
        $scope.fullName = function () {
```

## AngularJS – MVC.Net

```

        return $scope.firstName + " " + $scope.lastName;
    }
    $scope.alertName = function () {
        alert($scope.fullName());
    }
    });
</script>
</head>
<body >
    <form name="myform" ng-app="myApp" >
        <div ng-controller="EmployeeCtrl">
            First Name: <input type="Text" name="EmployeeFirstName"
ID="EmployeeFirstID" ng-required="true" ng-model="firstName"> <span ng-
show="myform.EmployeeFirstName.$error.required">It is require.</span> <br> <br>
            Last Name: <input type="Text" name="EmployeeLastName" ID="EmployeeLastID"
ng-required="true" ng-model="lastName"> <span ng-
show="myform.EmployeeLastName.$error.required">It is require.</span> <br> <br>
            <br>
            Full Name: {{fullName()}} <br>

            <button ng-click="alertName()" value="Click" >Click</button>
        </div>
    </form>
</body>
</html>

```

First Name:  It is require.

Last Name:  It is require.

Full Name:

### 3.2 Example 14 - ng-controller and List Basic

Following Example shows basic of List in AngularJS. List is the JavaScript array which could be used for multiple purpose.

Followings needs to understand in this example.

1. ng-repeat:-> This is more or less same as C# for each. AngularJS repeat value for given Array. .

```
ng-repeat="name in names"
```

2. Array:-> To use ng-repeat , AngularJS requires Array to be used with-in scope.

## AngularJS – MVC.Net

```
$scope.names = ['Abhay', 'Amruta', 'Ajay', 'Alok'];
```

**ng-controller and List**

```
<html>
  <head>
    <meta charset="utf-8">
    <title>Angular.js Example</title>
    <!--<script
src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.3.14/angular.min.js"></script-->
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
    <script>
      var app = angular.module('myApp', []);
      app.controller('EmployeeCtrl', function ($scope) {
        $scope.names = ['Abhay', 'Amruta', 'Ajay', 'Alok'];
      });
    </script>
  </head>
  <body >
    <form name="myform" ng-app="myApp" >
      <div ng-controller="EmployeeCtrl">

        List of Employee:
        <ul>
          <li ng-repeat="name in names">{{name}}</li>
        </ul>

      </div>
    </form>
  </body>
</html>
```

List of Employee:

- Abhay
- Amruta
- Ajay
- Alok

### 3.3 Example 15 - ng-controller and List Basic - Add

Following Example shows basic of List in AngularJS. To add any value in array AngularJS use push method, which is describe in this example.

Followings needs to understand in this example.

1. Array.push:-> Push method helps to add value in array. As AngularJS handle binding its own, we don't need to do any extra work to refresh the list.

## AngularJS – MVC.Net

## ng-controller and List Basic - Add

```
<html>
  <head>
    <meta charset="utf-8">
    <title>Angular.js Example</title>
    <!--<script
src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.3.14/angular.min.js"></script>-->
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
    <script>
      var app = angular.module('myApp', []);
      app.controller('EmployeeCtrl', function ($scope) {

        $scope.firstName = '';

        $scope.names = ['Abhay', 'Amruta', 'Ajay', 'Alok'];

        $scope.addName = function () {
          $scope.names.push($scope.firstName);
          $scope.firstName = '';
        };

      });
    </script>
  </head>
  <body ng-app="myApp" ng-controller="EmployeeCtrl">
    <form name="myform" ng-submit="addName()" >
      <div >
        Employee: <input type="text" ng-model="firstName" />
        <input type="submit" value="Add" > <br />
        List of Employee:
        <ul>
          <li ng-repeat="name in names">{{name}}</li>
        </ul>
      </div>
    </form>
  </body>
</html>
```

Employee:

List of Employee:

- Abhay
- Amruta
- Ajay
- Alok



## AngularJS – MVC.Net

Employee:

List of Employee:

- Abhay
- Amruta
- Ajay
- Alok

Employee:

List of Employee:

- Abhay
- Amruta
- Ajay
- Alok
- Abhishek

### 3.4 Example 16 - ng-controller and List Basic - Remove

Following Example shows basic of List in AngularJS. To remove any value in array AngularJS use push method, which is describe in this example.

Followings needs to understand in this example.

1. Array.splice:-> Splice method helps to remove value from array. As AngularJS handle binding its own, we don't need to do any extra work to refresh the list.

#### ng-controller and List Basic - Remove

```
<html>
<head>
  <meta charset="utf-8">
  <title>Angular.js Example</title>
  <!--<script
src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.3.14/angular.min.js"></script-->
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
  <script>
    var app = angular.module('myApp', []);
    app.controller('EmployeeCtrl', function ($scope) {

      $scope.firstName = '';

      $scope.names = ['Abhay', 'Amruta', 'Ajay', 'Alok'];

      $scope.addName = function () {
```

## AngularJS – MVC.Net

```

$scope.names.push($scope.firstName);
$scope.firstName = '';
});

$scope.removeName = function (name) {
    var i = $scope.names.indexOf(name);
    $scope.names.splice(i, 1);
};

});
</script>
</head>
<body ng-app="myApp" ng-controller="EmployeeCtrl">
    <form name="myform" ng-submit="addName()" >
        <div >
            Employee: <input type="text" ng-model="firstName" />
            <input type="submit" value="Add" /> <br />
            List of Employee:
            <ul>
                <li ng-repeat="name in names">{{name}} <a href="" ng-
click="removeName(name)">remove</a></li>
            </ul>
        </div>
    </form>
</body>
</html>

```

Employee:

List of Employee:

- Abhay [remove](#)
- Amruta [remove](#)
- Ajay [remove](#)
- Alok [remove](#)

Employee:

List of Employee:

- Abhay [remove](#)
- Amruta [remove](#)
- Ajay [remove](#)
- Alok [remove](#)
- Abhishek [remove](#)

## AngularJS – MVC.Net

Employee:

Add

List of Employee:

- Abhay [remove](#)
- Amruta [remove](#)
- Ajay [remove](#)
- Alok [remove](#)

### 3.5 Example 17 - ng-controller and Object List Basic – Add/Remove

Following Example shows basic of Object List in AngularJS. We are following same example as above and show object list example. In Javascript, Easy way to create Object list is JSON data which is used in this example.

This example is follow with above example. In this example, only JSON Data is used to describe about Object List with Add/Remove functionalities.

#### ng-controller and Object List Basic – Add/Remove

```
<html>
  <head>
    <meta charset="utf-8">
    <title>Angular.js Example</title>
    <!--<script
src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.3.14/angular.min.js"></script-->
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
    <script>
      var app = angular.module('myApp', []);
      app.controller('EmployeeCtrl', function ($scope) {

        $scope.firstName = '';
        $scope.salary = 0.0;

        var Empllyee = function () {
          name = '';
          Salary = 0;
        }

        $scope.emplyees = [
          { "name": "Abhay", "Salary": 10000 },
          { "name": "Amruta", "Salary": 10000 },
          { "name": "Ajay", "Salary": 10000 },
          { "name": "Alok", "Salary": 10000 },
        ];
      });
    </script>
  </head>
  <body>
    <div>
      <div>Employee: <input type="text"></div>
      <div>Add</div>
    </div>
    <div>List of Employee:</div>
    <ul>
      <li>• Abhay <a href="#">remove</a></li>
      <li>• Amruta <a href="#">remove</a></li>
      <li>• Ajay <a href="#">remove</a></li>
      <li>• Alok <a href="#">remove</a></li>
    </ul>
  </body>
</html>
```

[illegible]

Employee:

Salary:

List of Employee:

Name	Salary	
Abhay	\$10,000.00	<a href="#">remove</a>
Amruta	\$10,000.00	<a href="#">remove</a>
Ajay	\$10,000.00	<a href="#">remove</a>
Alok	\$10,000.00	<a href="#">remove</a>

AngularJS – MVC.Net

---

Employee: Salary: 

List of Employee:

Name	Salary	
Abhay	\$10,000.00	<a href="#">remove</a>
Amruta	\$10,000.00	<a href="#">remove</a>
Ajay	\$10,000.00	<a href="#">remove</a>
Alok	\$10,000.00	<a href="#">remove</a>

Employee: Salary: 

List of Employee:

Name	Salary	
Abhay	\$10,000.00	<a href="#">remove</a>
Amruta	\$10,000.00	<a href="#">remove</a>
Ajay	\$10,000.00	<a href="#">remove</a>
Alok	\$10,000.00	<a href="#">remove</a>
Abhishek	\$5,000.00	<a href="#">remove</a>

## AngularJS – MVC.Net

Employee:   
Salary:

List of Employee:

Name	Salary	
Amruta	\$10,000.00	<a href="#">remove</a>
Ajay	\$10,000.00	<a href="#">remove</a>
Alok	\$10,000.00	<a href="#">remove</a>
Abhishek	\$5,000.00	<a href="#">remove</a>

### 3.6 Example 18 - ng-controller and Object List Basic – filter

Following Example shows filter on Object List in AngularJS. We are following same example as above and show object list example.

This example is follow with above example. In this example, Only filter get applied this is client side filter via AngularJS.

Filter is applicable for all column available in object List as shown in example. We have applied filter on name and salary.

#### ng-controller and Object List Basic – filter

```

<html>
  <head>
    <meta charset="utf-8">
    <title>Angular.js Example</title>
    <!--<script
src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.3.14/angular.min.js"></script-->
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
    <script>
      var app = angular.module('myApp', []);
      app.controller('EmployeeCtrl', function ($scope) {

        $scope.firstName = '';
        $scope.salary = 0.0;

        var Employee = function () {
          name = '';
          Salary = 0;
        }
      }
    </script>
  </head>
  <body>
    <div>
      Employee: <input type="text">
      Salary: <input type="text" value="0">
      <input type="button" value="Add"/>
    </div>
    <div>
      List of Employee:
      <table>
        <thead>
          <tr>
            <th>Name</th>
            <th>Salary</th>
            <th></th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td>Amruta</td>
            <td>$10,000.00</td>
            <td><a href="#">remove</a></td>
          </tr>
          <tr>
            <td>Ajay</td>
            <td>$10,000.00</td>
            <td><a href="#">remove</a></td>
          </tr>
          <tr>
            <td>Alok</td>
            <td>$10,000.00</td>
            <td><a href="#">remove</a></td>
          </tr>
          <tr>
            <td>Abhishek</td>
            <td>$5,000.00</td>
            <td><a href="#">remove</a></td>
          </tr>
        </tbody>
      </table>
    </div>
  </body>
</html>

```

## AngularJS – MVC.Net

[illegible]

## AngularJS – MVC.Net

Employee: Salary: Search: 

List of Employee:

Name	Salary	
Abhay	\$10,000.00	<a href="#">remove</a>
Amruta	\$15,000.00	<a href="#">remove</a>
Ajay	\$30,000.00	<a href="#">remove</a>
Alok	\$20,000.00	<a href="#">remove</a>

Employee: Salary: Search: 

List of Employee:

Name	Salary	
Amruta	\$15,000.00	<a href="#">remove</a>



## AngularJS – MVC.Net

Employee:   
Salary:   
  
Search:   
  
List of Employee:  

Name	Salary	
Alok	\$20,000.00	<a href="#">remove</a>

### 3.7 Example 19 - ng-controller and Object List Basic – Orderby

Following Example shows orderBy on Object List in AngularJS. We are following same example as above and show object list example.

This example is follow with above example. In this example, Only orderby get applied this is client side orderBy via AngularJS.

OrderBy is applicable for column which is applied in object List as shown in example.

We have applied orderBy on salary. We can “-“ to do descending or default is ascending. In this example, we have used descending order salary.

#### ng-controller and Object List Basic – Orderby

```

<html>
  <head>
    <meta charset="utf-8">
    <title>Angular.js Example</title>
    <!--<script
src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.3.14/angular.min.js"></script-->
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
    <script>
      var app = angular.module('myApp', []);
      app.controller('EmployeeCtrl', function ($scope) {

        $scope.firstName = '';
        $scope.salary = 0.0;

        var Employee = function () {
          name = 'AA';

```

## AngularJS – MVC.Net

```

Salary = 0;

}
$scope.employees = [
{ "name": "Abhay", "Salary": 10000 },
{ "name": "Amruta", "Salary": 15000 },
{ "name": "Ajay", "Salary": 30000 },
{ "name": "Alok", "Salary": 20000 },
];

$scope.addName = function () {
    var e = new Employee();
    e.name = $scope.firstName;
    e.Salary = $scope.salary;
    $scope.employees.push(e);
    $scope.firstName = '';
    $scope.salary = 0.0;
};

$scope.removeName = function (emp) {
    var i = $scope.employees.indexOf(emp);
    $scope.employees.splice(i, 1);
};

});
</script>
</head>
<body ng-app="myApp" ng-controller="EmployeeCtrl">
    <form name="myform" ng-submit="addName()" >
        <div >
            Employee: <input type="text" ng-model="firstName" /> <br /> <br />
            Salary:&nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; <input type="text" ng-
model="salary" /> <br /> <br />
            <input type="submit" value="Add" /> <br />
            Search: <input type="text" ng-model="query" /> <br /> <br />
            List of Employee:

            <Table>
                <tr><th>Name</th><th>Salary</th><th></th></tr>
                <tr ng-repeat="emp in employees | filter:query | orderBy:'-
Salary'"><td>{{emp.name}}</td><td>{{emp.Salary | currency}}</td> <td> <a href="" ng-
click="removeName(emp)">remove</a></td> </tr>
            </Table>
        </div>
    </form>
</body>
</html>

```

## AngularJS – MVC.Net

Employee:   
Salary:   
  
Search:   
  
List of Employee:  

Name	Salary	
Ajay	\$30,000.00	<a href="#">remove</a>
Alok	\$20,000.00	<a href="#">remove</a>
Amruta	\$15,000.00	<a href="#">remove</a>
Abhay	\$10,000.00	<a href="#">remove</a>

### 3.8 Example 20 - ng-controller and Object List Basic – Orderby Dynamic

Following Example shows dynamic order by on Object List in AngularJS. We are following same example as above and show object list example.

This example is follow with above example. In this example, we are doing order by dynamically. this is client side orderBy via AngularJS.

OrderBy is applicable for column which is applied in object List as shown in example.

We have applied default order by on name column.

We have created sortField property which will contain the name of field which require to get sorted. In this example “name” is default-sorted field.

We have created reverse property/flag, which will contain the order by “asending-false” / “descending – true” value, which require to be sort. In this example “asending” is default-sorted sequence.

#### ng-controller and Object List Basic – Orderby Dnynamic

```

<html>
  <head>
    <meta charset="utf-8">
    <title>Angular.js Example</title>
    <!--<script
src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.3.14/angular.min.js"></script>-->
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
    <script>

```

## AngularJS – MVC.Net

[illegible]

## AngularJS – MVC.Net

```
</html>
```

Employee: Salary: Search: 

List of Employee:

<u>Name</u>	<u>Salary</u>	
Abhay	\$10,000.00	<a href="#">remove</a>
Ajay	\$30,000.00	<a href="#">remove</a>
Alok	\$20,000.00	<a href="#">remove</a>
Amruta	\$15,000.00	<a href="#">remove</a>

Employee: Salary: Search: 

List of Employee:

<u>Name</u>	<u>Salary</u>	
Amruta	\$15,000.00	<a href="#">remove</a>
Alok	\$20,000.00	<a href="#">remove</a>
Ajay	\$30,000.00	<a href="#">remove</a>
Abhay	\$10,000.00	<a href="#">remove</a>

### 3.9 Example 21 - ng-controller and Object List– Dropdown

Following Example shows how to use dropdown via Object list.

This example is follow with above example. In this example, we are adding one dropdown to select/filter employee.

## AngularJS – MVC.Net

In this example, we have shown that how to assign value also for dropdown. To explore this in detail, We have added EmpID column in JSON Data.

Followings needs to understand in this example.

1. ng-options:-> The ngOptions attribute can be used to dynamically generate a list of <option> elements for the <select> element using the array or object obtained by evaluating the ngOptions comprehension expression.
2. Track by:-> This is used to assign values for options.

```
ng-options="emp.name for emp in employees track by emp.empId"
```

### ng-controller and Object List Basic – Dropdown

```
<html>
<head>
  <meta charset="utf-8">
  <title>Angular.js Example</title>
  <!--<script
src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.3.14/angular.min.js"></script>-->
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
  <script>
    var app = angular.module('myApp', []);
    app.controller('EmployeeCtrl', function ($scope) {

      $scope.firstName = '';
      $scope.salary = 0.0;
      $scope.sortField = 'name';
      $scope.reverse = false;

      var Employee = function () {
        name = 'AA';
        Salary = 0;
        empId = -1;
      }

      $scope.employees = [
        { "empId":101, "name": "Abhay", "Salary": 10000 },
        { "empId": 102, "name": "Amruta", "Salary": 15000 },
        { "empId": 103, "name": "Ajay", "Salary": 30000 },
        { "empId": 104, "name": "Alok", "Salary": 20000 },
      ];

      $scope.addName = function () {
        var e = new Employee();
        e.name = $scope.firstName;
        e.Salary = $scope.salary;
        $scope.employees.push(e);
        $scope.firstName = '';
        $scope.salary = 0.0;
      };

      $scope.removeName = function (emp) {
        var i = $scope.employees.indexOf(emp);
        $scope.employees.splice(i, 1);
      };
    });
  </script>
</head>
<body>
  <div>
    <div>
      <input type="text" value="" />
      <input type="text" value="" />
      <input type="button" value="Add" />
    </div>
    <div>
      <table>
        <tr>
          <th>EmpID</th>
          <th>Name</th>
          <th>Salary</th>
        </tr>
        <tr>
          <td>101</td>
          <td>Abhay</td>
          <td>10000</td>
        </tr>
        <tr>
          <td>102</td>
          <td>Amruta</td>
          <td>15000</td>
        </tr>
        <tr>
          <td>103</td>
          <td>Ajay</td>
          <td>30000</td>
        </tr>
        <tr>
          <td>104</td>
          <td>Alok</td>
          <td>20000</td>
        </tr>
      </table>
    </div>
  </div>
</body>
</html>
```

## AngularJS – MVC.Net

[illegible]

Employee:

Salary:

Search:

List of Employee:

<u>Name</u>	<u>Salary</u>	
Abhay	\$10,000.00	<a href="#">remove</a>
Ajay	\$30,000.00	<a href="#">remove</a>
Alok	\$20,000.00	<a href="#">remove</a>
Amruta	\$15,000.00	<a href="#">remove</a>

## AngularJS – MVC.Net

Employee:   
Salary:   
Add  
Search:  ▼  
List of Employee:  

Name	Salary	
Abhay	\$10,000.00	<a href="#">remove</a>

```

<br>
"
Search: "
▼<select id="ddEmployee" ng-model="query" ng-options="emp.name for emp in employees track by emp.empId" class="ng-valid ng-dirty ng-valid-parse ng-touched">
<option value="101" selected="selected" label="Abhay">Abhay</option>
<option value="102" label="Amruta">Amruta</option>
<option value="103" label="Ajay">Ajay</option>
<option value="104" label="Alok">Alok</option>
</select>

```

### 3.10 Example 22 - ng-controller and Object List– Dropdown Option

Following Example shows how to use dropdown via Object list.

This example is follow with above example. In this example, we are adding one dropdown to select/filter employee.

In this example, we have shown that how to assign value also for dropdown by using ng-repeat.

#### ng-controller and Object List Basic – Dropdown Option

```

<html>
  <head>
    <meta charset="utf-8">
    <title>Angular.js Example</title>
    <!--<script
src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.3.14/angular.min.js"></script>-->
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
    <script>
      var app = angular.module('myApp', []);
      app.controller('EmployeeCtrl', function ($scope) {

        $scope.firstName = '';
        $scope.salary = 0.0;

```



## AngularJS – MVC.Net

```

$scope.sortField = 'name';
$scope.reverse = false;

var Employee = function () {
    name = '';
    Salary = 0;
    empId = -1;
}

$scope.employees = [
{ "empId": 101, "name": "Abhay", "Salary": 10000 },
{ "empId": 102, "name": "Amruta", "Salary": 15000 },
{ "empId": 103, "name": "Ajay", "Salary": 30000 },
{ "empId": 104, "name": "Alok", "Salary": 20000 },
];

$scope.addName = function () {
    var e = new Employee();
    e.name = $scope.firstName;
    e.Salary = $scope.salary;
    $scope.employees.push(e);
    $scope.firstName = '';
    $scope.salary = 0.0;
};

$scope.removeName = function (emp) {
    var i = $scope.employees.indexOf(emp);
    $scope.employees.splice(i, 1);
};

});
</script>
</head>
<body ng-app="myApp" ng-controller="EmployeeCtrl">
    <form name="myform" ng-submit="addName()" >
        <div >
            Employee: <input type="text" ng-model="firstName" /> <br /> <br />
            Salary:&nbsp; &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<input type="text" ng-
model="salary" /> <br /> <br />
            <input type="submit" value="Add" > <br />
            Search: <select id="ddEmployee" ng-model="query" >
                <option ng-repeat="emp in employees"
value="{{emp.empId}}">{{emp.name}} </option>
            </select> <br /> <br />
            List of Employee:
            <Table>
                <tr><th><a href="" ng-click="sortField = 'name'; reverse =
!reverse">Name</a></th><th><a href="" ng-click="sortField = 'Salary'; reverse =
!reverse">Salary</a></th><th></th></tr>
                <tr ng-repeat="emp in employees | filter:query |
orderBy:sortField:reverse"><td>{{emp.name}}</td><td>{{emp.Salary | currency}}</td>
<td> <a href="" ng-click="removeName(emp)">remove</a></td> </tr>
            </Table>
        </div>
    </form>
</body>
</html>

```

## AngularJS – MVC.Net

Employee: Salary: Search: 

List of Employee:

<u>Name</u>	<u>Salary</u>	
Abhay	\$10,000.00	<a href="#">remove</a>
Ajay	\$30,000.00	<a href="#">remove</a>
Alok	\$20,000.00	<a href="#">remove</a>
Amruta	\$15,000.00	<a href="#">remove</a>

Employee: Salary: Search: 

List of Employee:

<u>Name</u>	<u>Salary</u>	
Abhay	\$10,000.00	<a href="#">remove</a>

Search: "

```
<select id="ddEmployee" ng-model="query" class="ng-valid ng-dirty ng-valid-parse ng-touched">
  <!-- ngRepeat: emp in employees -->
  <option ng-repeat="emp in employees" value="101" class="ng-binding ng-scope">Abhay </option>
  <!-- end ngRepeat: emp in employees -->
  <option ng-repeat="emp in employees" value="102" class="ng-binding ng-scope">Amruta </option>
  <!-- end ngRepeat: emp in employees -->
  <option ng-repeat="emp in employees" value="103" class="ng-binding ng-scope">Ajay </option>
  <!-- end ngRepeat: emp in employees -->
  <option ng-repeat="emp in employees" value="104" class="ng-binding ng-scope">Alok </option>
  <!-- end ngRepeat: emp in employees -->
</select>
```

## AngularJS – MVC.Net

### 4.0 AngularJS – Routing and Template

Routing is used for deep-linking URLs to controllers and views (HTML partials). It watches `$location.url()` and tries to map the path to an existing route definition.

Routing is taken care by AngularJS routing engine. To use this we have to include “angular-route.min.js” in file.

Once JS file get included, we have to configure route by “\$routeProvider” into AngularJS Application module, which we have already discussed before.

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

Templates are written with HTML that contains Angular-specific elements and attributes. Angular combines the template with information from the model and controller to render the dynamic view that a user sees in the browser.

#### 4.1 Example 23 – Routing and Template

Following Example shows how to use Routing and templates in AngularJS.

To understand this, We have done some changes in previous example.

1. Make data available to multiple control we have created data as Javascript object and then assign to controller.
2. In This example template is created in Javascript Itself as Text.
3. We have pass one parameter call “[‘ngRoute’]” into AngularJS application module.
4. We have used ng-View directive to define area of rendering Data.
5. ngView is a directive that complements the \$route service by including the rendered template of the current route into the main layout (index.html) file. Every time the current route changes, the included view changes with it according to the configuration of the \$route service.
6. AngularJS require configuring routes into AngularJS application module.
7. AngularJS does this by “.config” method.
8. \$routeProvider used for configuring routes.
9. When(path, route):-> When method adds a new route definition to the \$route service.
10. otherwise(params) :->Otherwise method Sets route definition that will be used on route change when no other route definition is matched.

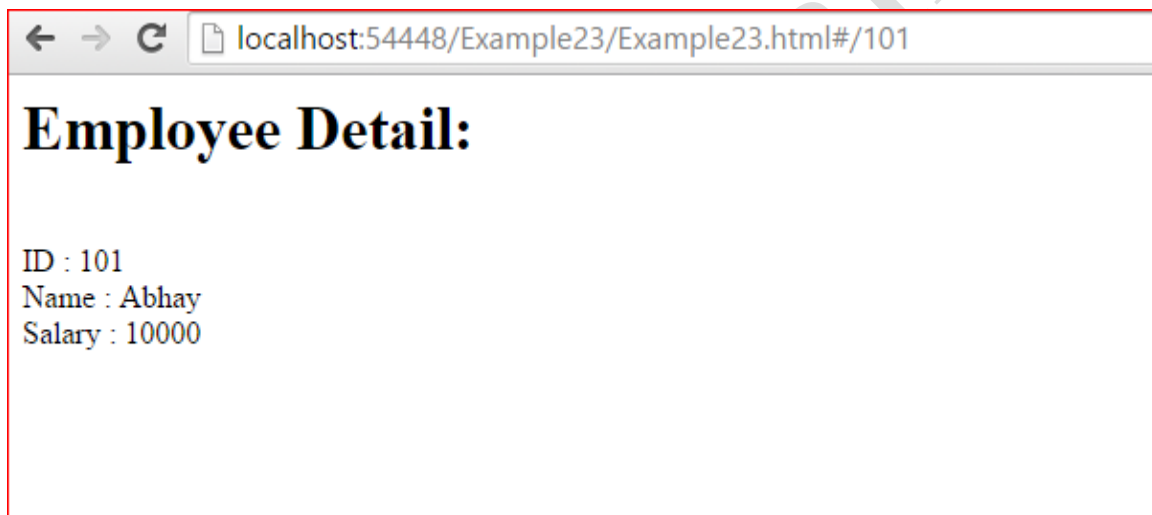
#### Routing and Template

```
<html ng-app="myApp" >
  <head>
    <meta charset="utf-8">
    <title>Angular.js Example</title>
    <!--<script
src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.3.14/angular.min.js"></script-->
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
```

## AngularJS – MVC.Net

```
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular-  
route.min.js"></script>  
<script>  
  var app = angular.module('myApp', ['ngRoute']);  
  
  var employeesData = [  
    { "empId": 101, "name": "Abhay", "Salary": 10000 },  
    { "empId": 102, "name": "Amruta", "Salary": 15000 },  
    { "empId": 103, "name": "Ajay", "Salary": 30000 },  
    { "empId": 104, "name": "Alok", "Salary": 20000 },  
  ];  
  
  app.config(function ($routeProvider) {  
    $routeProvider.  
      when('/', {  
        template: '<h1>Employee List:</h1> <BR><ul><li ng-repeat="emp in  
employees"> <a ng-href="#/{emp.empId}">{{emp.name}}</a> </li><ul>',  
        controller: 'EmployeeListCtrl'  
      }).  
      when('/:empId', {  
        template: '<h1>Employee Detail:</h1> <BR>ID :  
{{employee.empId}}<BR>Name : {{employee.name}}<BR>Salary : {{employee.Salary}}<BR>',  
        controller: 'EmployeeDetailCtrl'  
      }).  
      otherwise({  
        redirectTo: '/'  
      });  
  });  
  
  app.controller('EmployeeDetailCtrl', function ($scope, $routeParams) {  
    $scope.empId = $routeParams.empId;  
  
    $scope.employees = employeesData;  
    $scope.employee = $scope.employees.filter(function (entry) {  
      return entry.empId == $scope.empId;  
    })[0];  
  
  });  
  
  app.controller('EmployeeListCtrl', function ($scope) {  
    $scope.employees = employeesData;  
  
  });  
</script>  
</head>  
<body >  
  <div ng-view>  
  
  </div>  
  
</body>  
</html>
```

## AngularJS – MVC.Net



## 4.2 Example 24 – Routing with Different Html Page

Following Example shows how to use Routing and templates with different html page in AngularJS.

To explain this, We are using same example as above but just keeping template into different file.

### Routing and Template - Example24.html

```
<html ng-app="myApp" >
  <head>
    <meta charset="utf-8">
    <title>Angular.js Example</title>
    <!--<script
src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.3.14/angular.min.js"></script-->
```

## AngularJS – MVC.Net

```

<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular-
route.min.js"></script>
  <script>
    var app = angular.module('myApp', ['ngRoute']);

    var employeesData = [
      { "empId": 101, "name": "Abhay", "Salary": 10000 },
      { "empId": 102, "name": "Amruta", "Salary": 15000 },
      { "empId": 103, "name": "Ajay", "Salary": 30000 },
      { "empId": 104, "name": "Alok", "Salary": 20000 },
    ];

    app.config(function ($routeProvider) {
      $routeProvider.
        when('/', {
          template: '<h1>Employee List:</h1> <BR><ul><li ng-repeat="emp in
employees"> <a ng-href="#/{emp.empId}">{{emp.name}}</a> </li><ul>',
          controller: 'EmployeeListCtrl'
        }).
        when('/:empId', {
          template: '<h1>Employee Detail:</h1> <BR>ID :
{{employee.empId}}<BR>Name : {{employee.name}}<BR>Salary : {{employee.Salary}}<BR>',
          controller: 'EmployeeDetailCtrl'
        }).
        otherwise({
          redirectTo: '/'
        });
    });

    app.controller('EmployeeDetailCtrl', function ($scope, $routeParams) {
      $scope.empId = $routeParams.empId;

      $scope.employees = employeesData;
      $scope.employee = $scope.employees.filter(function (entry) {
        return entry.empId == $scope.empId;
      })[0];

    });

    app.controller('EmployeeListCtrl', function ($scope) {
      $scope.employees = employeesData;
    });
  </script>
</head>
<body >
  <div ng-view>

  </div>
</body>
</html>

```

## Routing and Template - Example24\_EList.html

## AngularJS – MVC.Net

```
<h1>Employee List:</h1>
<br>
<ul>
  <li ng-repeat="emp in employees">
    <a ng-href="#/{{emp.empId}}">{{emp.name}}</a>
  </li>
</ul>
```

## Routing and Template - Example24\_EDetail.html

```
<h1>Employee Detail:</h1>
<br>
ID : {{employee.empId}}<br>
Name : {{employee.name}}<br>
Salary : {{employee.Salary}}<br>
```



## AngularJS – MVC.Net



## 5.0 AngularJS – Bootstrap

AngularJS is JavaScript framework to support html rendering and Databinding. However, to do UI more user friendly we require CSS. Bootstrap is a popular style sheet, which we can use to make our page visible and colorful.

### 5.1 Example 25 – Bootstrap

In this example, We have done just little extension of above example and apply style sheet for same.

We have add reference of Bootstrap CDN link for CSS and JS.

#### BootStrap - Example25.html

```
<html ng-app="myApp" >
  <head>
    <meta charset="utf-8">
    <title>Angular.js Example</title>

    <!-- Latest compiled and minified CSS -->
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-
datepicker/1.4.0/css/bootstrap-datepicker.min.css">
    <!-- Optional theme -->
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap-theme.min.css">

    <!-- Latest compiled and minified JavaScript -->
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-
datepicker/1.4.0/js/bootstrap-datepicker.min.js"></script>
```



## AngularJS – MVC.Net

```
<!--<script
src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.3.14/angular.min.js"></script>-->
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular-
route.min.js"></script>

<script>
    var app = angular.module('myApp', ['ngRoute']);

    var employeesData = [
        { "empId": 101, "name": "Abhay", "Salary": 10000, "DOJ": "2014-07-07" },
        { "empId": 102, "name": "Amruta", "Salary": 15000, "DOJ": "2014-07-07" },
        { "empId": 103, "name": "Ajay", "Salary": 30000, "DOJ": "2014-07-07" },
        { "empId": 104, "name": "Alok", "Salary": 20000, "DOJ": "2014-07-07" },
    ];

    app.config(function ($routeProvider) {
        $routeProvider.
            when('/', {
                templateUrl: 'Example25_EList.html',
                controller: 'EmployeeListCtrl'
            }).
            when('/:empId', {
                templateUrl: 'Example25_EDetail.html',
                controller: 'EmployeeDetailCtrl'
            }).
            when('/Edit/:empId', {
                templateUrl: 'Example25_EEdit.html',
                controller: 'EmployeeDetailCtrl'
            }).
            otherwise({
                redirectTo: '/'
            });
    });

    app.controller('EmployeeDetailCtrl', function ($scope, $routeParams) {
        $scope.empId = $routeParams.empId;

        $scope.employees = employeesData;
        $scope.employee = $scope.employees.filter(function (entry) {
            return entry.empId == $scope.empId;
        })[0];
    });

    app.controller('EmployeeListCtrl', function ($scope) {
        $scope.employees = employeesData;
    });
</script>
</head>
<body >
```

## AngularJS – MVC.Net

```

    <div class="container">
      <div ng-view>

        </div>
      </div>
    </body>
  </html>

```

## Routing and Template - Example24\_EList.html

```

<table class="table table-striped">
  <thead>
    <tr>
      <th>Employee List</th>
    </tr>
    <tr>
      <th>Emp ID </th>
      <th>Name</th>
      <th>Salary</th>
      <th>Edit</th>
    </tr>
  </thead>
  <tbody>
    <tr ng-repeat="emp in employees">
      <td><a ng-href="#/{{emp.empId}}"> {{emp.empId}}</a></td>
      <td> {{emp.name}}</td>
      <td> {{emp.Salary | currency}}</td>
      <td>
        <a ng-href="#/Edit/{{emp.empId}}">
          <button class="btn">
            <span class="glyphicon glyphicon-pencil"></span> Edit
          </button>
        </a>
      </td>
    </tr>
  </tbody>
</table>

```

## Routing and Template - Example24\_EDetail.html

```

<h1>Employee Detail:</h1>

<form name="frmEmployee" class="form-horizontal">

  <div class="form-group">
    <label class="col-sm-2 control-label">ID:</label>
    <div class="col-sm-10">
      <input type="text" ng-model="employee.empId" ng-disabled="!edit"
placeholder="Name">
    </div>
  </div>

  <div class="form-group">
    <label class="col-sm-2 control-label">Employee Name:</label>
    <div class="col-sm-10">
      <input type="text" ng-model="employee.name" ng-disabled="!edit"

```

## AngularJS – MVC.Net

```

placeholder="Name">
    </div>
</div>
<div class="form-group">
    <label class="col-sm-2 control-label">Salary:</label>
    <div class="col-sm-10">
        <input type="text" name="empSalary" ng-model="employee.Salary" ng-
disabled="!edit" placeholder="Salary" ng-pattern="/^\d{0,9}(\.\d{1,9})?$/">
        <span ng-show="frmEmployee.empSalary.$error.pattern">Not valid
Salary!</span><br>
    </div>
</div>
<div class="form-group">
    <label class="col-sm-2 control-label">DOJ:</label>
    <div class="col-sm-10">
        <input type="date" name="employeeDOJ" ID="employeeDOJID" ng-
disabled="!edit" ng-model="employee.DOJ" min="2010-01-01" value="{{ employee.DOJ |
date: 'yyyy-MM-dd' }}" />

    </div>
</div>
</form>

<a ng-href="#/">Back</a>

```

## Routing and Template - Example24\_EEdit.html

```

<h1>Employee Edit</h1>

<form name="frmEmployee" class="form-horizontal">

    <div class="form-group">
        <label class="col-sm-2 control-label">ID:</label>
        <div class="col-sm-10">
            <input type="text" ng-model="employee.empId" ng-disabled="!edit"
placeholder="Name">
        </div>
    </div>

    <div class="form-group">
        <label class="col-sm-2 control-label">Employee Name:</label>
        <div class="col-sm-10">
            <input type="text" ng-model="employee.name" ng-disabled="edit"
placeholder="Name">
        </div>
    </div>

    <div class="form-group">
        <label class="col-sm-2 control-label">Salary:</label>
        <div class="col-sm-10">
            <input type="text" name="empSalary" ng-model="employee.Salary" ng-
disabled="edit" placeholder="Salary" ng-pattern="/^\d{0,9}(\.\d{1,9})?$/">

```

## AngularJS – MVC.Net

```

        <span ng-show="frmEmployee.empSalary.$error.pattern">Not valid
Salary!</span><br>
    </div>
</div>
<div class="form-group">
    <label class="col-sm-2 control-label">DOJ:</label>
    <div class="col-sm-10">
        <input type="date" name="employeeDOJ" ID="employeeDOJID" ng-
model="employee.DOJ" min="2010-01-01" value="{{ employee.DOJ | date: 'yyyy-MM-dd' }}"
/>

    </div>
</div>
</form>
<a ng-href="#/">Back</a>

```

Employee List			
Emp ID	Name	Salary	Edit
101	Abhay	\$10,000.00	 Edit
102	Amruta	\$15,000.00	 Edit
103	Ajay	\$30,000.00	 Edit
104	Alok	\$20,000.00	 Edit

## Employee Edit

ID:

101

Employee Name:

Abhay

Salary:

10000

DOJ:

07/07/2014

[Back](#)

## AngularJS – MVC.Net

## Employee Edit

ID: Employee Name: Salary: DOJ: [Back](#)

### Employee List

Emp ID	Name	Salary	Edit
101	Abhay	\$20,000.00	<a href="#">Edit</a>
102	Amruta	\$15,000.00	<a href="#">Edit</a>
103	Ajay	\$30,000.00	<a href="#">Edit</a>
104	Alok	\$20,000.00	<a href="#">Edit</a>

## Employee Detail:

ID: Employee Name: Salary: DOJ: [Back](#)

## AngularJS – MVC.Net

---

### 6.0 AngularJS – IIS hosting

AngularJS is JavaScript framework it can host on IIS, Apache, node.js and other available hosting environment.

As we would like to extend our examples for MVC, we are using IIS.

#### 6.1 IIS Hosting

In this example, we have done just little extension of above example.

Now, this is time to host these examples on any server.

We are going to use host these example on IIS server as we are going to add MVC application on it.

To add any application on IIS, we have to follow some steps, which are following one.

1. Consolidate files:-> Lets keep all examples folder in one folder.
2. Open IIS: If IIS is install on your m/c., you can just type “inetmgr” on run window.
3. Site Name: Add site name as require.
4. Application pool: select application pool.
5. Physical Path: Add physical files, which has all example files.
6. Port: If you want, you can change port.

## AngularJS – MVC.Net

**Add Website**

Site name:

Application pool:

Content Directory

Physical path:

Pass-through authentication

Binding

Type:  IP address:  Port:

Host name:

Example: www.contoso.com or marketing.contoso.com

☒ Start Website immediately

## 7.0 AngularJS – First Application

In this section we are going to develop one of our first AngularJS Applications. To develop any application we should consider following things which are used in this section.

1. Login Functionality
2. Page Title
3. Grid/List Functionality
4. Grid/List-Search Functionality
5. Grid/List-Sorting Functionality

## AngularJS – MVC.Net

6. Grid/List-Paging Functionality
7. Add Functionality
8. Edit Functionality
9. Detail Functionality
10. Remove Functionality
11. Resource File
12. Log-out Functionality

This example has all functionalities but remember still we have not talk about server and all functionalities are maintained in client side.

### 7.1 Login Functionality

Login is critical functionality for all Application, when we work with AngularJS, we should maintain two level login server and client both side. As the beauty of AngularJS is to work on client side this functionality become critical for us.

There are multiple ways to achieve this functionality like cookies and local storage. In this example, we have used local storage although both one of them can be used here.

Also, like every request on server side require to check Authentication/Authorization management, this need to check in JavaScript also. In AngularJS, we can do this “resolve” functionality in routing.

In this example, we have done the same and created method “checkLogin”. In this, we have checked user has already logged-in or not. If user has already login into system it will automatically redirect list Page. This functionality we have achieved by “goToDashBoard” Method.

Routing is managed by “Example26\_config.JS” files.

#### Login Add Local Storage

```
$scope.LoginSubmit = function () {  
    var loginUser = $scope.loginData.filter(function (entry) {  
        return entry.UserID == $scope.UserID && entry.Password == $scope.Password;  
    })[0];  
  
    if (!loginUser) {  
        $scope.showMessage = true;  
    }  
    else {  
        if (loginUser.UserID == $scope.UserID && loginUser.Password ==  
$scope.Password) {  
            srtLogin(loginUser.UserID);  
            window.location = window.location + 'List';  
        }  
    }  
};
```



## AngularJS – MVC.Net

```
function srtLogin(val) {
    if (typeof (Storage) !== "undefined") {
        localStorage.setItem("UserID", val);
    }
}

function goToDashBoard() {
    if (typeof (Storage) !== "undefined") {
        var valUser = localStorage.getItem("UserID");

        if (valUser !== null) {
            if (valUser !== "-1") {
                window.location = "/Example26/Example26.html#/List";
            }
        }
    }
}
```

## 7.2 Page Title

Page Title is also one of critical functionality, which should implement for all pages.

This can achieve by routing configuration as it has feature to add title into it.

Once title has been added into routing configuration, it can implement into page as ng-binding.

Title
<pre>when('/', {     title: 'Login',     templateUrl: 'Example26_Elogin.html?v=1.0',     controller: 'EmployeeLoginCtrl',     resolve: { load: function () { goToDashBoard(); } } })</pre>
<pre>&lt;html ng-app="myApp" &gt;   &lt;head&gt;     &lt;meta charset="utf-8"&gt;     &lt;title ng-bind="title"&gt;&lt;/title&gt;</pre>

## 7.3 Grid/List Functionality

Any project Grid/List functionality is very common and required. To implement this in AngularJS we require only Data in JSON format and use “ng-repeat” tag with template.

We have implemented Grid/list in HTML Tabular format template and JSON data.

We need to set routing configuration for List page and provide controller name as well as template or Template URL. In this example, we have used Template URL.

## AngularJS – MVC.Net

As we have already discussed that check login functionality needs to implement for all controller we are implementing this via “checkLogin()” method.

We have created “EmployeeListCtrl” controller, which will have responsibility to get data from either server or from JS data and pass to View.

We have created “” template page for Grid/List, which use for display grid/list and data/model get from controller bind to it.

### Grid/List

```
var employeesData = [
  { "empId": 101, "name": "Abhay", "Salary": 10000, "DOJ": "2014-07-07", "gender": "Male", "Role": "Architect" },
  { "empId": 102, "name": "Amruta", "Salary": 15000, "DOJ": "2014-07-07", "gender": "Female", "Role": "Test Lead" },
  { "empId": 103, "name": "Ajay", "Salary": 30000, "DOJ": "2014-07-07", "gender": "Male", "Role": "Manager" },
  { "empId": 104, "name": "Alok", "Salary": 20000, "DOJ": "2014-07-07", "gender": "Male", "Role": "Developer" },
  { "empId": 105, "name": "Abhishek", "Salary": 20000, "DOJ": "2014-07-07", "gender": "Male", "Role": "Developer" },
  { "empId": 106, "name": "AmitB", "Salary": 20000, "DOJ": "2014-07-07", "gender": "Male", "Role": "QA" },
  { "empId": 107, "name": "Dnyanesh", "Salary": 20000, "DOJ": "2014-07-07", "gender": "Male", "Role": "Developer" },
  { "empId": 108, "name": "Kiran", "Salary": 20000, "DOJ": "2014-07-07", "gender": "Male", "Role": "Developer" },
  { "empId": 109, "name": "Dharmin", "Salary": 20000, "DOJ": "2014-07-07", "gender": "Male", "Role": "Developer" },
  { "empId": 110, "name": "Tushar", "Salary": 20000, "DOJ": "2014-07-07", "gender": "Male", "Role": "Developer" },
];

when('/List', {
  title: 'Home',
  templateUrl: 'Example26_EList.html?v=1.0',
  controller: 'EmployeeListCtrl',
  resolve: { init: function () { checkLogin();} }
})

function checkLogin() {
  if (typeof (Storage) !== "undefined") {
    var valUser = localStorage.getItem("UserID");

    if (valUser !== null) {
      if (valUser == "-1") {
        window.location = "/Example26/Example26.html";
      }
    }
    else {
      window.location = "/Example26/Example26.html";
    }
  }
}
```

## AngularJS – MVC.Net

```

}
app.controller('EmployeeListCtrl', function ($scope) {

    $scope.employees = employeesData;

    $scope.sort = function (keyname) {
        $scope.sortKey = keyname; //set the sortKey to the param passed
        $scope.reverse = !$scope.reverse; //if true make it false and vice versa
    }

    $scope.removeName = function (emp) {
        var i = $scope.employees.indexOf(emp);
        $scope.employees.splice(i, 1);
    };

});

<form class="form-inline">
    <div class="form-group">
        <label >Search</label>
        <input type="text" ng-model="search"
class="form-control" placeholder="Search">
    </div>

<table class="table table-striped">
    <thead>
        <tr>
            <th colspan="3">Employee List
            <!--<input type="text" ng-model="search.name" placeholder="Search Name" />-->
        </th>
            <th>
                <a ng-href="#/Add">
                    <button class="btn">
                        <span class="glyphicon glyphicon-pencil"></span> Add
                    </button>
                </a>
            </th>
        </tr>
        <tr style="cursor:pointer">
            <th ng-click="sort('empId')">
                <span class="glyphicon sort-icon" ng-show="sortKey=='empId'" ng-
class="{ 'glyphicon-chevron-up':reverse, 'glyphicon-chevron-down':!reverse}"> </span>
Emp ID
            </th>
            <th ng-click="sort('name')">
                <span class="glyphicon sort-icon" ng-show="sortKey=='name'" ng-
class="{ 'glyphicon-chevron-up':reverse, 'glyphicon-chevron-down':!reverse}"> </span>
Name</th>
            <th ng-click="sort('Salary')">
                <span class="glyphicon sort-icon" ng-show="sortKey=='Salary'" ng-
class="{ 'glyphicon-chevron-up':reverse, 'glyphicon-chevron-down':!reverse}"> </span>
Salary</th>
            <th>Edit</th>
            <th></th>
        </tr>
    </thead>

```

## AngularJS – MVC.Net

```

</thead>
<tbody>
  <tr dir-paginate="emp in
employees|orderBy:sortKey:reverse|filter:search|itemsPerPage:5" >
    <td><a ng-href="#/{{emp.empId}}"> {{emp.empId}}</a></td>
    <td> {{emp.name}}</td>
    <td> {{emp.Salary | currency}}</td>
    <td>
      <a ng-href="#/Edit/{{emp.empId}}">
        <button class="btn">
          <span class="glyphicon glyphicon-pencil"></span> Edit
        </button>
      </a>
    </td>
    <td>
      <a href="" ng-click="removeName(emp)">
        remove
      </a>
    </td>
  </tr>
</tbody>

</table>
<dir-pagination-controls
max-size="5"
direction-links="true"
boundary-links="true" >
</dir-pagination-controls>

</form>

```

## 7.4 Grid/List Search Functionality

We have achieved this functionality by using AngularJS feature filter option.

AngularJS apply this filter for all columns, which available in JSON Data even it is bind/display to Grid/List or not.

In addition, we require one input box, which will help user to require search criteria.

### Grid/List Search Functionality

```

<tbody>
  <tr dir-paginate="emp in
employees|orderBy:sortKey:reverse|filter:search|itemsPerPage:5" >
    <div class="form-group">
      <label>Search</label>
      <input type="text" ng-model="search"
class="form-control" placeholder="Search">
    </div>

```

Note: We have used “dirPagination” library to implement paging/sorting.

## AngularJS – MVC.Net

### 7.5 Grid/List Sorting Functionality

We have achieved this functionality by using AngularJS feature orderBy option.

AngularJS apply this filter for requested columns it could be on ascending and descending.

In addition, we require one input box, which will help user to require search criteria.

#### Grid/List Search Functionality

```
<tbody>
  <tr dir-paginate="emp in
employees|orderBy:sortKey:reverse|filter:search|itemsPerPage:5" >
    <div class="form-group">
      <label >Search</label>
      <input type="text" ng-model="search"
class="form-control" placeholder="Search">
    </div>
```

Note: We have used “dirPagination” library to implement paging/sorting.

### 7.6 Grid/List Paging Functionality

We have achieved this functionality by using AngularJS extended library “dirPagination”. This library has almost all feature which we require to do in any grid. But we have to understand that this library work well when all data is available on client side only.

This library has following features:

1. Paging
2. Sorting
3. Filter
4. Order By

You can download this library from GitHub. This library is open source and we can use this in any application.

<https://github.com/angular-ui/bootstrap/blob/master/src/pagination/pagination.js>

### 7.7 Add Functionality

To Implement Add functionality, we just need to add values into JSON Data/model. Once new data get added into JSON Data it automatically reflect to all other functionality like Grid.

## AngularJS – MVC.Net

To achieve this we have used “EmployeeDetailCtrl” which has “addemployee” method.

We have created “Example26\_EAdd.html” template, which has require validation and HTML, controls where user can provide require information.

**Add Functionality**

```
when('/Add', {
  title: 'Home',
  templateUrl: 'Example26_EAdd.html?v=1.0',
  controller: 'EmployeeDetailCtrl',
  resolve: { load: function () { checkLogin(); } }
})

app.controller('EmployeeDetailCtrl', function ($scope, $routeParams) {
  $scope.empId = $routeParams.empId;
  $scope.name = '';
  $scope.Salary = 0.0;
  $scope.DOJ;
  $scope.gender = '';
  $scope.Role = '';

  $scope.employees = employeesData;
  $scope.RoleList = RoleData;

  $scope.employee = $scope.employees.filter(function (entry) {
    return entry.empId == $scope.empId;
  })[0];

  $scope.addemployee = function () {
    var e = new employee();
    var v = getMax($scope.employees, 'empId')

    e.empId = v.empId + 1;
    e.name = $scope.name;
    e.Salary = $scope.Salary;
    e.DOJ = $scope.DOJ;
    e.gender = $scope.gender;
    e.Role = $scope.Role;
    $scope.employees.push(e);

    backButton();
  };
});

<h1>Employee Add</h1>

<form name="frmEmployee" ng-submit="addemployee()" class="form-horizontal">

  <div class="form-group">
    <label class="col-sm-2 control-label">ID:</label>
    <div class="col-sm-10">
      <input type="text" ng-model="empId" ng-disabled="!edit" placeholder="ID">
    </div>
  </div>
</div>
```

## AngularJS – MVC.Net

```

<div class="form-group">
  <label class="col-sm-2 control-label">Employee Name:</label>
  <div class="col-sm-10">
    <input type="text" name="empName" ng-model="name" ng-required="true" ng-
disabled="edit" placeholder="Name">
    <span ng-show="frmEmployee.empName.$error.required">Name is
required!</span><br>
  </div>
</div>
<div class="form-group">
  <label class="col-sm-2 control-label">Salary:</label>
  <div class="col-sm-10">
    <input type="text" name="empSalary" ng-model="Salary" ng-disabled="edit"
placeholder="Salary" ng-pattern="/^\d{0,9}(\.\d{1,9})?$/">
    <span ng-show="frmEmployee.empSalary.$error.pattern">Not valid
Salary!</span><br>
  </div>
</div>
<div class="form-group">
  <label class="col-sm-2 control-label">DOJ:</label>
  <div class="col-sm-10">
    <input type="date" name="employeeDOJ" ID="employeeDOJID" ng-model="DOJ"
min="2010-01-01" placeholder="yyyy-MM-dd" value="{{ employee.DOJ | date: 'yyyy-MM-dd'
}}"/>
  </div>
</div>
<div class="form-group">
  <label class="col-sm-2 control-label">Gender:</label>
  <div class="col-sm-10">
    <input type="radio" ng-model="gender" value="Male" /> Male
    <input type="radio" ng-model="gender" value="Female" /> Female

    <span ng-bind="employee.gender"></span>
  </div>
</div>
<div class="form-group">
  <label class="col-sm-2 control-label">Role:</label>
  <div class="col-sm-10">
    <select id="ddEmployee" ng-model="Role" >
      <option value="">Select Role</option>
      <option ng-repeat="role in RoleList" value="{{role.Role}}" ng-
selected="role.Role == employee.Role">{{role.Role}}
    </option>
    </select>
    <span ng-bind="Role"></span>
  </div>
</div>
<div class="form-group">
  <div class="col-sm-10">
    <input type="submit" value="Add">
  </div>
</div>
</form>

<a style="cursor:pointer" onclick="backButton()">Back</a>

```

## AngularJS – MVC.Net

---

### 7.8 Edit Functionality

To Implement Edit functionality, we just need get reference of exact data object from JSON Data array and assign pass to view.

To achieve this we have used “EmployeeDetailCtrl” which has “filter” method to retrieve reference of require object as we can see in above.

We have created “Example26\_EEdit.html” template, which has require validation and HTML, controls where user can provide require information.

Here “empId” is a parameter, which can retrieve, by controller and implement require logic.

#### Edit Functionality

```
when('/Edit/:empId', {  
  title: 'Employee Edit',  
  templateUrl: 'Example26_EEdit.html?v=1.0',  
  controller: 'EmployeeDetailCtrl',  
  resolve: { load: function () { checkLogin(); } }  
})
```

### 7.9 Detail Functionality

To Implement Detail functionality, we just need get reference of exact data object from JSON Data array and assign pass to view.

To achieve this we have used “EmployeeDetailCtrl” which has “filter” method to retrieve reference of require object as we can see in above.

We have created “Example26\_EDetail.html” template, which has require validation and HTML, controls where user can provide require information.

#### Detail Functionality

```
when('/:empId', {  
  title: 'Employee Detail',  
  templateUrl: 'Example26_EDetail.html?v=1.0',  
  controller: 'EmployeeDetailCtrl',  
  resolve: { load: function () { checkLogin(); } }  
})
```

### 7.10 Remove Functionality

To Implement remove functionality, we just need get reference of exact data object from JSON Data array and need to remove that from JSON Array.



## AngularJS – MVC.Net

It will automatically reflect the Grid.

### Remove Functionality

```
$scope.removeName = function (emp) {
    var i = $scope.employees.indexOf(emp);
    $scope.employees.splice(i, 1);
};
```

## 7.11 Resource Functionality

Resource management is also one of the critical functionality for any project. As end user can ask any time to change messages, label on page, this should be configurable for all applications and should not require compiling code also if one place we have changed, it should reflect to all pages.

## 7.12 Log-out Functionality

Log-out is critical functionality for all Application, which has login functionality, when we work with AngularJS, we should maintain two level log-out feature, server and client both side. As the beauty of AngularJS is to work on client side this functionality become critical for us.

To implement Log-out on client side, I have assigned one garbage value to it , you can implement any logic here.

### Logout Functionality

```
function srtLogout(val) {
    if (typeof (Storage) !== "undefined") {
        localStorage.setItem("UserID", val);
        window.location = "/Example26/Example26.html";
    }
}
```

## 7.13 Example 26 – First Application

Example 26 has implemented all feature as explained in above section to implement

## 8.0 AngularJS – First Application

## AngularJS – MVC.Net

---

1. AngularJS is a HTML enhanced for Web Apps.
2. To use Cache in AngularJS we have to use factory and dependency injection.
- 3.
4. It is a framework for HTML/web application .
5. It includes web components and model driven view.
6. It is developed by Misko Hungry.
7. It is a databinding framework.
8. Basically it is developed for Single Page Application.
9. This is maintained by Google.
10. It has initial release in 2009.
11. Model contains only data no behaviour.
12. Controller is set of functions which
13. Code mirror is library which does code highlight and
14. AngularJS provides 3 important things.
  - Routing – Updating page based on URL.
  - Templating –
  - Data binding – It maintains synchronization between Model and View.
15. Other Libraries:
  - Underscore
  - Backbone – 1<sup>st</sup> MVC in Javascript
  - JQuery
  - Q
  - AMD
  - RequiredJS
  - Promises
  - Handlebars – it is templating library.
  - Mustache- it is first templating library.
16. AngularJS has JQLite which is equivalent of JQuery but not JQuery.
17. AngularJS has convention that Controller should end with “Ctrl”.
18. We can create our own directive also please check online.
19. AngularJS provides module as “\$scope” for example:

```
function NameCtrl($scope){
    $scope.firstName = 'John';
    $scope.lastName = 'Smith';
}
<body ng-controller="NameCtrl">
```
20. AngularJS can have multiple ng-App in one page.

```
<html ng-app="nameApp">
var nameApp = angular.module('nameApp', []);
nameApp.controller('NameCtrl', function ($scope){
    $scope.firstName = 'John';
    $scope.lastName = 'Smith';
});
```

## AngularJS – MVC.Net

21. You can add “\$Scope.\$watch” function , when any property get change.

22. AngularJS has “\$Scope.\$apply()” method to apply changes on model.

23. AngularJS has “push” operation to add data in array.

```
$scope.names = ['abhay', 'smita', 'amruta'];
$scope.addName = function() {
    $scope.names.push($scope.enteredName);
};
});
```

```
<input type="text" ng-model="enteredName">
```

24. AngularJS has “splice” operation to remove data in array.

```
$scope.removeName = function(name) {
    var i = $scope.names.indexOf(name);
    $scope.names.splice(i, 1);
};
});
```

25. AngularJs has “ng-submit” attribute, if you add this in form it will call function assign to it when you click on submit button.

```
<html ng-app="nameApp">
<head>
    <meta charset="utf-8">
    <title>Angular.js Example</title>
    <script
src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.2.1/angular.min.js"></script>
    <script>
        var nameApp = angular.module('nameApp', []);
        nameApp.controller('NameCtrl', function ($scope){
            $scope.names = ['Larry', 'Curly', 'Moe'];

            $scope.addName = function() {
                $scope.names.push($scope.enteredName);
            };
        });
    </script>
</head>
<body ng-controller="NameCtrl">
    <ul>
        <li ng-repeat="name in names">{{ name }}</li>
    </ul>
    <form ng-submit="addName()">
        <input type="text" ng-model="enteredName">
        <input type="submit" value="add">
    </form>
</body>
```

## AngularJS – MVC.Net

---

</html>

26. Angular has “filter” operation to include filter with ng-repeate.

Search:<input ng-model="query" type="text"/>

<table>

<tr>

<th>Country</th>

<th>Population</th>

</tr>

<tr ng-repeat="country in countries | filter:query">

<td>{{ country.name }}</td>

<td>{{ country.population }}</td>

</tr>

</table>

27. Angular has “Orderby” operation to include order by operation with ng-repeate.

28.

29. To use Https secure url , need to use following code.

```
var forceSSL = function () {
```

```
  if ($location.protocol() !== 'https') {
```

```
    $window.location.href = $location.absUrl().replace('http', 'https');
```

```
  }
```

```
};
```

```
forceSSL();
```

30. AngularJs always refresh full view whenever it has any changes in viewmodel.

31. Maintain status of page on back and forward button of browser is difficult in SPA applications.

32. Always add querystring with current version , in all pages , even css and js files.

33. AngularJs has \$scope to define scope. Help to expose public properties.

34. AngularJs has \$http to use Ajax Dom operations.

35. AngularJs has \$location helps to do navigation from one URL to another URL.

36. These all are possible in one framework by using Angularjs, as compare Knockout we have to use multiple frameworks.

37. AngularJs does not use observable pattern which is used by Knockout.

38. AngularJs used ViewModle concepet, In this if viewmodle change view autometicly get change.

Developer does not deal with View in AngularJS. It always refresh full view.

39. There is no depency on Microsoft, It has developed by google.

40. SPA is a concept but for Application we implement as hybrid SPA and Page View.

41. It should be mini multiple SPA to create Applications.

42. It has routing engine by using file “Angular-route.min.js”

43. When we use MVC and AngularJs then there are two routing engine used one is Asp.net mvc and second is AjgularJs routing engine. To handle both engine simlteniously and work properly, we should use following lines into MVC RouteConfig. Here “/{\*catchally}” ignore appending URL which is handle by Angular and “Customer” route managed by Asp.net MVC.

## AngularJS – MVC.Net

---

```
routes.MapRoute(1
    name: "customer",
    url: "customer/{*catchall}",
    defaults: new { controller = "Home", action = "Customer" });
```

44. It has routing engine by using file “Angular-route.min.js”

Js Files:

- 45. Angular.min.js
- 46. Angular-route.min.js
- 47.

KeyWords	Use	Eaxmple	Comments
ng-app	The root element of an AngularJS Or AngularJs Directive	<	It could be on DIV or Body.
ng-bind	It sets the text of components	<span ng-bind="name"></span>	Work with Span.
Ng-model	It is two way binding between the view and scope.	<input type="text" ng-model="orderService.orderId" />	Works with textbox, checkbox , radio, dropdown.
ng-model-options			
ng-class			
ng-controller	Specifies a JavaScript controller class that evaluates HTML expressions.		
ng-show & ng-hide			
ng-switch			
ng-view			
ng-if			
ng-aria			
ng-animate			
{{ variableName }}	To display value	{{ HeadingCaption }}	
Data-ng-controller	To make scope of controller	Data-ng-controller="HomepageViewModel"	
ng-repeat	It is same as for each in C#.	ng-repeat="p in People"	

**AngularJS – MVC.Net**

KeyWords	Use	Eaxmple	Comments
ng-click	To assign click function	ng-click="showPerson(p)"	
ng-submit		<form ng-submit="addName()">	
ng-keydown			
Ng-src			