Contents

Angular JS 1

1. Introduction to Angular JS .....
2. Data Binding .....
3. AngularJS Expressions .....
4. AngularJS Applications .....
5. AngularJS Number .....
6. AngularJS String .....
7. AngularJS Object .....
8. AngularJS Arrays .....
9. AngularJS Directives .....
10. Creatign Directive .....
11. AngularJS Modules .....
    - 11.1 Defining the Module in HTML .....
    - 11.2 Creating the Module object in JS Script .....
    - 11.3 Module Loading & Dependencies .....
    - 11.4 Configuration Blocks .....
    - 11.5 Dependencies .....
    - 11.6 Creation or Retrieval .....

Scanned by CamScanner

12. AngularJS Controllers .....

12.1 Defining the controller in HTML .....

12.2 Creating the controller object in JS Script .....

12.3 Setting up the initial state of a $scope object .....

12.4 Adding Behavior to a Scope Object .....

12.5 External JS Files .....

12.6 Scope Inheritance .....

12.7 Two Way Data Binding .....

13. AngularJS Filters .....

13.1 uppercase Filter .....

13.2 lowercase Filter .....

13.3 currency Filter .....

13.4 date Filter .....

13.5 orderBy Filter .....

13.6 filter Filter .....

13.7 Custom Filter .....

Scanned by CamScanner

1. Introduction to Angular JS

- It is Open source client side Javascript framework to build well architected dynamic web application.
- It is not a library of useful functions.
- It enlarges browser based application with MVC capability. It provides a platform to write client side application using Java Script using MVC.

Model:

Model is responsible for all behaviour and data of the application.
It is responsible for managing the data.
It includes all creation, retrieval, update and delete of data.

View:

View is responsible for the presentation of the data.
It is also responsible for the User intraction.

Controller:

Controller is responsible for controlling the request.
It handles the intraction between the view and Model.

- It enhances HTML by adding directives, custom tags, attributes, expressions and templates with HTML.
- AngularJS directives are HTML attributes with an ng prefix.
- When you want to make your page HTML valid then you can use data-ng-, instead of ng- .
- It is perfect for Single Page Applications (SPAs).
- Misko Hevery, a Google employee, started to work with AngularJS in 2009.
- AngularJS version 1.0 was released in 2012.
- Angular instructs the browser about new syntax through a construct called as directives.
  - Data binding, as in {{}}.
  - DOM control structures for repeating, showing and hiding DOM fragments.
  - Support for forms and form validation.
  - Attaching new behavior to DOM elements, such as DOM event handling.
  - Grouping of HTML into reusable components.
- It provides
  - Built-in multi-browser support (cross browser compliant).
  - Code reusability by creating reusable component.
  - Data binding capacity
- It is distributed as a JavaScript file, and can be added to a web page with a <script> tag:

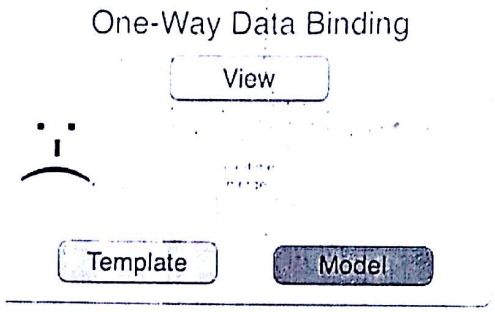
```
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.3/angular.min.js"></script>
<script src="script/angular.min.js"></script>
```

Scanned by CamScanner

2. Data Binding

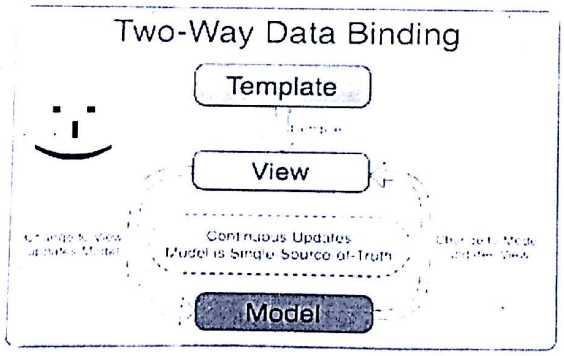
- Data-binding is the automatic synchronization of data between the model and view components.
- The view is a projection of the model at all times. When the model changes, the view reflects the change, and vice versa.

Data Binding in Classical Template Systems



- Once data is bound, changes to the model or related sections of the view are NOT automatically reflected in the view.
- Any changes that the user makes to the view are not reflected in the model.

Data Binding in Angular JS Template



- Angular templates work differently. First the template i.e. uncompiled HTML along with any additional markup or directives, is compiled on the browser.
- The compilation step produces a live view. Any changes to the view are immediately reflected in the model, and any changes in the model are propagated to the view. You can think of the view as simply an instant projection of your model.
- The controller is completely separated from the view and unaware of it. It is easy to test your controller in isolation without the view and the related DOM/browser dependency.

Scanned by CamScanner

3. AngularJS Expressions

- You can write the AngularJS expressions inside double braces.
- AngularJS will "display" data exactly where the expression is written.

  Syntax:

  {{ expression }}
- It bind AngularJS data to HTML in the same way as the ng-bind directive does.
- It can contain literals, operators, and variables.
- It does not support conditionals, loops, or exceptions.
- It support filters.
- The ng-init directive initialize AngularJS application variables.

  Ex:

  {{5 + 5}}
  {{ fname + " " + lname }}

index.html

```
<html>
<body>
<!-- <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.3/angular.min.js">
</script> -->
<script src="script/angular.min.js"></script>
<div ng-app=""> {{4+8}} </div>
{{4+8}}
</body>
</html>
```

index.html

```
<html>
<head> <script type="text/javascript" src="scripts/angular.min.js"></script> </head>
<body>
<div ng-app="">
   <input type="text" ng-model="myname" /> <br /><br />
   Your name is  <span ng-bind="myname" style="color:red;font-weight:bold;"></span>
   <br/> Your name is {{myname}}
</div>
<br /> Your name is {{myname}}
</body>
</html>
```

Scanned by CamScanner

- The ng-app directive defines an AngularJS application.
- The ng-model directive binds the value of HTML controls (input, select, textarea) to application data.
- The ng-bind directive binds application data to the HTML view.

Here

- AngularJS starts automatically when the web page has loaded.
- The ng-app directive instructs AngularJS that the element is the "owner" of an AngularJS application.
- The ng-model directive is binding the value of the input field to the application variable name.
- The ng-bind directive binds the innerHTML of the element to the application variable name.

4. AngularJS Applications

- The ng-app directive defines the application, the ng-controller directive defines the controller.
- AngularJS modules define AngularJS applications and AngularJS controllers control AngularJS applications.

index.html

```
<html> <head>
<script type="text/javascript" src="script/angular.min.js"></script>
</head> <body>
<div data-ng-app="studApp" data-ng-controller="StudentController">
        Enter Name <input type="text" data-ng-model="fname"><br>
        Enter Email <input type="text" data-ng-model="email"><br>
        <br> Full Name: {{fname + " , Email :" + email}}
        </div>
        <script>
        var app = angular.module('studApp', []);
        app.controller('StudentController', function($scope) {
                $scope.fname = "Manish";
                $scope.email = "manish@gmail.com";
        });
        </script>
</body> </html>
```

Scanned by CamScanner

5. AngularJS Number

index.html

```
<html> <head>
<script type="text/javascript" src="script/angular.min.js"></script>
</head>
<body>
        <div data-ng-app="" data-ng-init="ab=1;bc=5">
                <br /> Sum: {{ ab + bc }}
                <br /> Sum: <span data-ng-bind="ab + bc"></span>
        </div>
</body> </html>
```

6. AngularJS String

index.html

```
<html> <head>
<script type="text/javascript" src="script/angular.min.js"></script>
</head>
<body>
        <div data-ng-app="" data-ng-init="fname='Manish';lname='Ranjan'">
                <br /> Name: {{ fname +" "+ lname }}
                <br /> Name: <span data-ng-bind='fname+" " + lname'></span>
        </div>
</body> </html>
```

7. AngularJS Object

index.html

```
<html> <head>
<script type="text/javascript" src="script/angular.min.js"></script>
</head>
<body>
 <div data-ng-app="" data-ng-init="student={fname:'Manish',lname:'Ranjan'}">
        <br /> Name: {{ student.fname +" "+ student.lname }}
        <br /> Name: <span data-ng-bind='student.fname+" " + student.lname'></span>
</div>
</body>
</html>
```

Scanned by CamScanner

8. AngularJS Array

index.html

```
<html> <head>
<script type="text/javascript" src="script/angular.min.js"></script>
</head>
<body>
<div data-ng-app="" data-ng-init="values=[1,2,3,4]">
        <br /> Value: {{ values[0]}}
        <br /> Value: <span data-ng-bind='values[0]'></span>
</div>
</body> </html>
```

9. AngularJS Directives

- In Angular JS, You can extend HTML with new attributes called Directives.
- AngularJS directives are HTML attributes with an ng prefix.
- When you want to make your page HTML valid then you can use data-ng-, instead of ng- .
- The ng-app directive defines an AngularJS application.
- The ng-model directive binds the value of HTML controls (input, select, textarea) to application data.
- The ng-bind directive binds application data to the HTML view.
- The ng-init directive initialize AngularJS application variables.

Normalization

- AngularJS normalizes an element's tag and attribute name to determine which elements match which directives. We typically refer to directives by their case-sensitive camelCase normalized name (e.g. ngModel).
- The normalization process is as follows:
  - Remove x- and data- from the front of the element/attributes.
  - Convert the :, -, or _ delimited name to camelCase.
- For example, the following forms are all equivalent and match the ngBind directive:

```
<div data-ng-app="" >
        <input ng-model='name'/>
        <br /> {{ name}}
        <br /> <span ng-bind="name"></span>
        <br /> <span ng:bind="name"></span>
        <br /> <span ng_bind="name"></span>
        <br /> <span data-ng-bind="name"></span>
        <br /> <span x-ng-bind="name"></span>
</div>
```

Scanned by CamScanner

Directive types

- $compile can match directives based on element names (E), attributes (A), class names (C), and comments (M).
- The built-in the AngularJS directives show in their documentation page which type of matching they support.
- The following are the various ways a directive can be referenced from within a template.
    - <my-dir></my-dir>
    -
    -

Data Binding

- Data binding in AngularJS, synchronizes AngularJS expressions with AngularJS data.
- The {{ firstName }} expression is an AngularJS data binding expression and is synchronized with ng-model="firstName".

ng-app

- The ng-app directive defines the root element of an AngularJS application.
- The ng-app directive will automatically initialize the application when a web page is loaded.

ng-init

- The ng-init directive defines initial values for an AngularJS application datas.

ng-model

- The ng-model directive binds the value of HTML controls (input, select, textarea) to application data.
- The ng-model directive can also:
    - Provide type validation for application data (number, email, required).
    - Provide status for application data (invalid, dirty, touched, error).
    - Provide CSS classes for HTML elements.
    - Bind HTML elements to HTML forms.

ng-disabled

- This directive sets the disabled attribute on the element.

ng-click

- This directive allows you to specify custom behavior when an element is clicked.

ng-show

- This directive shows or hides the given HTML element based on the expression.

ng-hide

- This directive shows or hides the given HTML element based on the expression.

Scanned by CamScanner

index.html

```
<html> <head>
<script type="text/javascript" src="script/angular.min.js"></script>
</head>
<body>
        <div data-ng-app="myApp" data-ng-controller="MyController">
                <input type="button" value="SHOW " data-ng-click="show()" />
                <input type="button" value="HIDE " data-ng-click="hide()" />
                <input type="button" value="DISABLE" data-ng-click="disableButton()" />
                <input type="button" value="ENABLE" data-ng-click="enableButton()" />
                <input type="button" value="CLICK ME" data-ng-disabled="disableMe"
onclick="alert('You have clicked me');"/> <br />
                <p data-ng-show="showName">Manish K Ranjan</p>
                <p data-ng-hide="hideEmail">manish@gmail.com</p>
        </div>
        <script type="text/javascript">
                var app = angular.module("myApp", []);
                app.controller("MyController", function($scope) {
                        $scope.showName = true;
                        $scope.disableMe = true;
                        $scope.show = function() {
                                $scope.showName = true;
                                $scope.hideEmail = false;
                        }
                        $scope.hide = function() {
                                $scope.showName = false;
                                $scope.hideEmail = true;
                        }
                        $scope.disableButton = function() {
                                $scope.disableMe = true;
                        }
                        $scope.enableButton = function() {
                                $scope.disableMe = false;
                        }
                });
        </script>
</body> </html>
```

Scanned by CamScanner

ng-switch

- The ngSwitch directive is used to conditionally swap DOM structure on your template based on a scope expression.

index.html

```
<html> <head>
<script type="text/javascript" src="script/angular.min.js"></script>
</head>
<body>
        <div data-ng-app="myApp">
                <div data-ng-controller="myController">
                        <div data-ng-switch on="myData">
                                <div data-ng-switch-when="1">Value 1</div>
                                <div data-ng-switch-when="2">Value 2</div>
                                <div data-ng-switch-default>Default</div>
                        </div>
                </div>
        </div>
        <script>
                var app = angular.module("myApp", []);
                app.controller("myController", function($scope) {
                        $scope.myData = 1;
                });
        </script>
</body> </html>
```

ng-repeat

- The ng-repeat directive clones HTML elements once for each item in a collection (in an array).

Variable	Type	Description
$index	number	iterator offset of the repeated element (0..length-1).
$first	boolean	true if the repeated element is first in the iterator.
$middle	boolean	true if the repeated element is between the first and last in the iterator.
$last	boolean	true if the repeated element is last in the iterator.
$even	boolean	true if the iterator position $index is even
$odd	boolean	true if the iterator position $index is odd

Scanned by CamScanner

index.html

```
<html> <head>
<script type="text/javascript" src="script/angular.min.js"></script>
</head> <body>
        <div data-ng-app="myApp" data-ng-controller="MyController">
                <div data-ng-repeat="x in values">{{x}}</div>
        </div>
        <script type="text/javascript">
                var app = angular.module("myApp", []);
                app.controller("MyController", function($scope) {
                        $scope.values = [ 99, 33, 22, 11, 55, 66 ];
                });
        </script>
</body> </html>
```

- By default, ngRepeat does not allow duplicate items in arrays. Because when there are duplicates, it is not possible to maintain a one-to-one mapping between collection items and DOM elements.
- When you do need to repeat duplicate items then you can substitute the default tracking behavior with your own using the track by expression.

index.html

```
<html> <head>
<script type="text/javascript" src="script/angular.min.js"></script>
</head>
<body>
        <div data-ng-app="myApp" data-ng-controller="MyController">
                <div data-ng-repeat="x in values track by $index">{{x}}</div>
        </div>
        <script type="text/javascript">
                var app = angular.module("myApp", []);
                app.controller("MyController", function($scope) {
                        $scope.values = [ 99, 33, 22, 11, 55, 66, 99, 33, 11, 66, 99];
                });
        </script>
</body> </html>
```

Scanned by CamScanner

- The ng-repeat directive is perfect for displaying tables.

index.html

```
<html>
<head>
<script type="text/javascript" src="script/angular.min.js"></script>
</head> <body>
        <div data-ng-app="myApp" data-ng-controller="MyController">
                <table width='60%'>
                <tr> <td>ID</td> <td>NAME</td> <td>EMAIL</td> </tr>
                <tr data-ng-repeat="st in students">
                        <td>{{st.id}}</td> <td>{{st.name}}</td> <td>{{st.email}}</td>
                </tr>
                </table>
        </div>
        <script type="text/javascript">
                var app = angular.module("myApp", []);
                app.controller("MyController", function($scope) {
                $scope.students = [ { id : 44, name : 'Manish', email : 'manish@gmail.com'},
                                { id : 77, name : 'Sri', email : 'sri@gmail.com' },
                                { id : 66, name : 'Kumar', email : 'kumar@gmail.com' },
                                { id : 99, name : 'Nivas', email : 'nivas@gmail.com' }];
                });
        </script>
</body>
</html>
```

member

10. Creating Directives

- There are many build-in directives defined in Angular JS. When the build-in functions is not suitable for your requirement then you can define your own directives also.
- When you create custom directives then it must be directives are registered on modules.
- You can use the module.directive API to register custom directive on mocule.
- module.directive takes the normalized directive name followed by a factory function.
- The factory function is invoked only once when the compiler matches the directive for the first time.

Scanned by CamScanner

index.html

```
<html> <head>
<script type="text/javascript" src="script/angular.min.js"></script>
</head>
<body>
        <div data-ng-app="myApp">
                <div data-ng-controller="StudController">
                        <div data-my-student></div>
                </div>
        </div>
        <script>
        var app = angular.module('myApp', []);
        app.controller('StudController', function($scope) {
                $scope.student = {
                        name : 'Manish',
                        address : 'Bangalore, KA'
                };
        });
        app.directive('myStudent', function() {
        return {
                template : 'Name: {{student.name}} <br/> Address: {{student.address}}'
                };
        });
        </script>
</body> </html>
```

Directive name

modules

- When the template code if bigger then it is better to break the template apart into its own HTML file and load it with the templateUrl option.

student.html

```
<table>
        <tr>
                <td>Name</td>
                <td>Address</td>
        </tr>
        <tr data-ng-repeat="st in students">
                <td>{{st.name}}</td>
                <td>{{st.address}}</td>
        </tr>
</table>
```

Scanned by CamScanner

index.html

```
<html> <head>
<script type="text/javascript" src="script/angular.min.js"></script>
</head>
<body>
        <div data-ng-app="myApp">
                <div data-ng-controller="StudController">
                        <div data-my-student></div>
                </div>
        </div>
        <script>
                var app = angular.module('myApp', []);
                app.controller('StudController', function($scope) {
                        $scope.students = [ {
                                name : 'Manish',
                                address : 'Blore'
                        }, {
                                name : 'Kumar',
                                address : 'Hyd'
                        } ];
                });
                app.directive('myStudent', function() {
                        return {
                                templateUrl : 'student.html'
                        };
                });
        </script>
</body>
</html>
```

Scanned by CamScanner

11. AngularJS Modules

- It is a container for the different parts of app like controllers, services, filters, directives, etc.
- It defines an application.
- Defining the Module in HTML.

  <... ng-app="myApp">
- Creating the Module object in JS Script.

  var app = angular.module("myApp", []);
- The empty array in angular.module('myApp', []) is the list of modules myApp depends on.
- It is recommendable to break the application to multiple modules like this:
  - A module for each feature
  - A module for each reusable component (especially directives and filters)
  - And an application level module which depends on the above modules and contains any initialization code.
- The calls to angular.module can only be compiled after the library has been loaded. So before call to angular.module the library must be loaded.

Module Loading & Dependencies

- A module is a collection of configuration and run blocks which get applied to the application during the bootstrap process.
- The module consists of a collection of two kinds of blocks:
  - Configuration blocks
    - It is executed during the provider registrations and configuration phase.
    - Only providers and constants can be injected into configuration blocks.
  - Run blocks
    - It is executed after all of the services have been configured and the injector has been created and are used to kickstart the application.
    - Only instances and constants can be injected into run blocks.
    - Run blocks typically contain code which is hard to unit-test, and for this reason should be declared in isolated modules, so that they can be ignored in the unit-tests.

```
var app=angular.module('myApp', []);
app.config(function(injectables) {
        // Provider Injector
        You can only inject Providers (not instances) into config blocks.
}).
run(function(injectables) {
        // Instance-Injector
        // You can only inject instances (not Providers) into run blocks
});
```

Scanned by CamScanner

Configuration Blocks

- There are some convenience methods on the module which are equivalent to the config block.

```
var app=angular.module('myApp', []);
app.value('a', 123);
app.factory('a', function() { return 123; });
app.directive('directiveName', function (){});
app.filter('filterName', function (){});
```

IT IS SAME AS

```
var app =angular.module('myApp', []);
app.config(function($provide, $compileProvider, $filterProvider) {
  $provide.value('a', 123);
  $provide.factory('a', function() { return 123; });
  $compileProvider.directive('directiveName', function (){});
  $filterProvider.register('filterName', function (){});
});
```

Dependencies

- Modules can list other modules as their dependencies.
- Depending on a module implies that the required module needs to be loaded before the current module is loaded.
- The configuration blocks of the required modules execute before the configuration blocks of the current module.

Creation or Retrieval

- When you use angular.module('myApp', []) then it will create the module myApp and overwrite any existing module named myApp.
- You need to use angular.module('myApp') to retrieve an existing module.

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

  - Adding some directives and services

```
app.service('myService', function (){});
app.directive('myDirective', function (){});
```

  - Overwrites both myService and myDirective by creating a new module

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

  - throws an error because myOtherApp has yet to be defined

```
var app = angular.module('myOtherApp');
```

Scanned by CamScanner

12. AngularJS Controllers

- A controller is a JavaScript Object, created by a standard JavaScript object constructor.
- AngularJS applications are controlled by controllers.
- It control the data of AngularJS applications.
- The ng-controller directive defines the application controller.
- When a Controller is attached to the DOM via the ng-controller directive, Angular will instantiate a new Controller object, using the specified Controller's constructor function.
- A new child scope will be created and made available as an injectable parameter to the Controller's constructor function as $scope.
- If the controller has been attached using the controller as syntax then the controller instance will be assigned to a property on the new scope.
- Use controllers to:
  - Set up the initial state of the $scope object.
  - Add behavior to the $scope object.
- Do not use controllers to:
  - Manipulate DOM
    - Controllers should contain only business logic.
    - Putting any presentation logic into Controllers significantly affects its testability.
    - Angular has databinding for most cases and directives to encapsulate manual DOM manipulation.
  - Format input
    - You can use angular form controls.
  - Filter output
    - You can use angular filters.
  - Share code or state across controllers
    - You can use angular services.

Defining the controller in HTML

```
<... ng-app="myApp" ng-controller="StudentController">
```

Creating the controller object in JS Script.

```
app.controller('StudentController', function($scope) {
...
});
```

Setting up the initial state of a $scope object

- When you create an application you need to set up the initial state for the Angular $scope.
- You can set up the initial state of a scope by attaching properties to the $scope object.
- All the $scope properties will be available to the template at the point in the DOM where the Controller is registered.

Scanned by CamScanner

Adding Behavior to a Scope Object

- We add behavior to the scope by attaching methods to the $scope object.
- These methods are then available to be called from the template/view.

index.html

```
<html>
<head>
<script type="text/javascript" src="script/angular.min.js"></script>
</head>
<body>
        <div data-ng-app="myApp" data-ng-controller="StudentController">
          <br />Title <select data-ng-model="title">
                <option value="Mr" >Mr</option>
                <option value="Ms">Ms</option>
          </select> <br />
                <br />F Name <input type="text" data-ng-model="fname" /> <br />
                <br />M Name <input type="text" data-ng-model="mname" /> <br />
                <br />L Name <input type="text" data-ng-model="lname" /> <br />
                <br /> Full Name {{fullName(title)}}
        </div>
        <script>
                var app = angular.module('myApp', []);
                app.controller('StudentController', function($scope) {
                        $scope.fname = "Manish";
                        $scope.mname = "K";
                        $scope.lname = "Ranjan";
                        $scope.fullName = function(title) {
                                if(!title) {
                                        title = "Mr";
                                }
                  return title +" " +$scope.fname + " " + $scope.mname + " "+ $scope.lname;
                        }
                });
        </script>
</body>
</html>
```

Scanned by CamScanner

External JS Files

- We can definc all the script related code like module, controllers in external Java script file also.

index.html

```
<html> <head>
<script type="text/javascript" src="script/angular.min.js"></script>
<script type="text/javascript" src="script/myStudentController.js"></script>
</head>
<body>
<div data-ng-app="myApp" data-ng-controller="StudentController">
        <br />F Name <input type="text" data-ng-model="fname" />
        <br /> <br />M Name <input type="text" data-ng-model="mname" /> <br />
        <br />L Name <input type="text" data-ng-model="lname" /> <br />
        <br /> Full Name {{fullName()}}
        <hr />
        <div data-ng-repeat="st in students">
                {{st.id+" "+st.name +" " +st.email}}
        </div>
</div> </body> </html>
```

myStudentController.js

```
var app = angular.module('myApp', []);
app.controller('StudentController', function($scope) {
        $scope.fname = "";
        $scope.mname = "";
        $scope.lname = "";
        $scope.students = [ {
                id : 99,
                name : 'Manish',
                email : 'manish@gmail.com'
        }, {
                id : 88,
                name : 'Ranjan',
                email : 'ranjan@gmail.com'
        } ];
        $scope.fullName = function() {
                return $scope.fname + " " + $scope.mname + " " + $scope.lname;
        }
});
```

Scanned by CamScanner

Scope Inheritance

- Each Angular application has exactly one root scope, but may have several child scopes.
- Since the ng-controller directive creates a new child scope, we get a hierarchy of scopes that inherit from each other.
- When new scopes are created, they are added as children of their parent scope.
- The $scope that each Controller receives will have access to properties and methods defined by Controllers higher up the hierarchy.

index.html

```
<html>
<head>
<script type="text/javascript" src="script/angular.min.js"></script>
</head> <body>
        <div data-ng-app="myApp">
                <h1>{{name}}</h1>
                <div data-ng-controller="FirstController">
                        <h1>{{name+"\t"+email+"\t"+phone}}</h1>
                        <div data-ng-controller="SecondController">
                                <h1>{{name+"\t"+email+"\t"+phone}}</h1>
                                <div data-ng-controller="ThirdController">
                                        <h1>{{name+"\t"+email+"\t"+phone}}</h1>
                                </div>
                        </div>
                </div>
        </div>
        <script type="text/javascript">
                var app = angular.module("myApp", []);
                app.controller("FirstController", function($rootScope, $scope) {
                        $rootScope.name = "Srinivas";
                        $scope.name = "Manish";
                        $scope.email = "man@gmail.com";
                        $scope.phone = "11111";
                });
                app.controller("SecondController", function($scope) {
                        $scope.email = "rahul@gmail.com";
                        $scope.phone = "22222";
                });
                app.controller("ThirdController", function($scope) {
                        $scope.phone = "33333";
                });
```

Scanned by CamScanner

```
        </script>
</body>
</html>
```

- When Angular evaluates {{ <name> }}, first it looks at the scope associated with the given element for the <name> property.
- When no such property is found, it searches the parent scope and so on until the root scope is reached.

Two Way Data Binding

index.html

```
<html>
<head> <script type="text/javascript" src="script/angular.min.js"></script> </head>
<body>
<div data-ng-app="myApp" data-ng-controller='myController'>
        <br/>Name <input type ='text' data-ng-model='student.name'/>
        <br/>Name <input type ='text' data-ng-model='student.email'/>
        <br/><br/>
        <input  type='submit' data-ng-click='showData()' value='Click'/>
</div>

<script type="text/javascript">
var app = angular.module("myApp",[]);
app.controller('myController',function($scope){
        $scope.student = {};
        $scope.showData  = function(){
                console.log($scope.student);
        }
});
</script>
</body>
</html>
```

Scanned by CamScanner

13. AngularJS Filters

- Filters format the value of an expression for displaying to the user.
- They can be used in view templates, controllers or services.
- AngularJS comes with a collection of built-in filters.
- You can also define define your custom filter.
- AngularJS filters also can be used to transform data.
- Filters can be applied to expressions in view templates.
- You can add the filters to expressions and directives using a pipe character (|).

  {{ expression | filter }}
- Filters can be applied to the result of another filter. This is called "chaining".

  {{ expression | filter1 | filter2 | ... }}
- Filters may have arguments.

  {{ expression | filter:argument1:argument2:... }}

Filter	Description
uppercase	Format a string to upper case.
lowercase	Format a string to lower case.
currency	Format a number to a currency format.
orderBy	Orders an array by an expression.
filter	Select a subset of items from an array.
date	Formats date to a string based on the requested format

uppercase & lowercase

index.html

```
<html> <head>
<script type="text/javascript" src="script/angular.min.js"></script>
</head> <body>
        <div data-ng-app="">
                <br />F Name <input type="text" data-ng-model="fname" />
                <br />M Name <input type="text" data-ng-model="mname" />
                <br />L Name <input type="text" data-ng-model="lname" />
                <hr />
                Full Name: {{(fname+" "+mname+" "+lname) | uppercase }}
                <hr/>
                Full Name: {{(fname+" "+mname+" "+lname) | lowercase }}
        </div>
</body> </html>
```

Scanned by CamScanner

currency

{{ currency_expression | currency : symbol : fractionSize}}

- When no currency symbol is provided, default symbol for current locale is used.

index.html

```
<html>
<head>
<script type="text/javascript" src="script/angular.min.js"></script>
</head>
<body>
<div data-ng-app="">
                <br />Quantity <input type="text" data-ng-model="qty" /> <br />
                <br />Price <input type="text" data-ng-model="price" /> <br />
                <br /> Total Amount: {{ (qty * price) | currency}}
                <br /> Total Amount: {{ (qty * price) | currency:'Rs '}}
                <br /> Total Amount: {{ (qty * price) | currency:'Rs ':0}}
                <br /> Total Amount: {{ (qty * price) | currency:'Rs ':1}}
        </div>
</body>
</html>
```

date:

- Formats date to a string based on the requested format

  {{ date_expression | date : format : timezone}}
- Format string can be composed of the following elements:

Filter	Description
yyyy	4 digit year (0001, 2017)
yy	2 digit year (00-99)
MMMM	Full Month Name (January)
MMM	Three Character Month Name (Jan)
MM	Month in Digit (01)
M	Month in Digit (1)
dd	Day of Month (01-31)
d	Day of Month (1-31)
EEEE	Day in Week (Sunday)
EEE	Day in Week (Sun)

Scanned by CamScanner

HH	Hour in day (00-23)
hh	Hour in AM/PM (01-12)
mm	Minute in Hour (00-59)
ss	Second in Minute (00-59)
sss	Millisecond in Secons(000-999)
Z	4 Digit (+sign) of timezone offset (-1200 +1200)
medium	'MMM d, y h:mm:ss a' for en_US locale (Sep 3, 2010 12:05:08 PM)
short	'M/d/yy h:mm a' for en_US locale (9/3/10 12:05 PM)
fullDate	'EEEE, MMMM d, y' for en_US locale (Friday, September 3, 2010)
longDate	'MMMM d, y' for en_US locale (September 3, 2010)
mediumDate	'MMM d, y' for en_US locale (Sep 3, 2010)
shortDate	'M/d/yy' for en_US locale (9/3/10)
mediumTime	'h:mm:ss a' for en_US locale (12:05:08 PM)
'shortTime'	'h:mm a' for en_US locale (12:05 PM)

index.html

```
<html>
<head>
<script type="text/javascript" src="script/angular.min.js"></script>
</head>
<body>
        <div x-ng-app="myApp" data-ng-controller="DateController">

                <br />{{dvalue}}
                <br />{{dvalue | date}}
                <br />{{dvalue | date :'dd/MMM/yyyy hh:mm:ss a'}}
                <br />{{dvalue | date :'medium'}}
                <br />{{dvalue | date :'mediumDate'}}
                <br />{{dvalue | date :'mediumTime'}}
        </div>
        <script type="text/javascript">
        var app = angular.module("myApp",[]);
        app.controller("DateController",function($scope){
                $scope.dvalue=new Date();
        });
        </script>
</body>
</html>
```

Scanned by CamScanner

orderBy

- Orders a specified array by the expression predicate.
- It is ordered alphabetically for strings and numerically for numbers.

{{ orderBy_expression | orderBy : expression : reverse}}

index.html

```
<html>
<head> <script type="text/javascript" src="script/angular.min.js"></script> </head>
<body>
<div data-ng-app="" data-ng-init="values = [12,32,45,34,21,56,43]">
        <div data-ng-repeat="data in values | orderBy">
                {{data}}
        </div>
        <hr/>
        <div data-ng-repeat="data in values | orderBy : $index : 'reverse'">
                {{data}}
        </div>
</div>
</body>
</html>
```

index.html

```
<html>
<head> <script type="text/javascript" src="script/angular.min.js"></script> </head>
<body>
<div data-ng-app="myApp" data-ng-controller='myController'>
        <div data-ng-repeat="st in students | orderBy:'id' ">
                {{st.id+" "+st.name +" " +st.email}}
        </div>
<hr/>
        <div data-ng-repeat="st in students | orderBy:'-id' ">
                {{st.id+" "+st.name +" " +st.email}}
        </div>
<hr/>
        <div data-ng-repeat="st in students | orderBy:'id' : 'reverse' ">
                {{st.id+" "+st.name +" " +st.email}}
        </div>
</div>

<script type="text/javascript">
var app = angular.module("myApp",[]);
app.controller('myController',function($scope){
        $scope.students=[
                {id:44,name:'Manish',email:'manish@gmail.com'},
                {id:77,name:'Anand',email:'anand@gmail.com'},
```

Scanned by CamScanner

```
                {id:66,name:'Kumar',email:'kumar@gmail.com'},
                {id:99,name:'Sonu',email:'sonu@gmail.com'},
                {id:88,name:'Ranjan',email:'ranjan@gmail.com'}
                ];

});
</script>
</body>
</html>
```

filter

- The filter filter can only be used on arrays, and it returns an array containing only the matching items.

index.html

```
<html>
<head>
<script type="text/javascript" src="script/angular.min.js"></script>
</head>
<body data-ng-app='myApp'>
        <div data-ng-controller="MyController">
                <div>Names :
                        <span data-ng-repeat="name in names"><br/>{{name}}</span>
                </div>
                <div>
                        Filtered Name <span data-ng-repeat="name in names | filter : 'n'
"><br/> {{name}} </span>
                </div>
        </div>

        <script type="text/javascript">
                var app=angular.module('myApp', [])
                app.controller('MyController',function ($scope) {
                        $scope.names = [ 'Manish','Sri' , 'Ranjan' , 'Rahul' , 'Nivas', 'Dande'];
                });
        </script>
</body>
</html>
```

Scanned by CamScanner

Custom Filter

- You can add the custom filter with application object.

```
var app = angular.module("myApp", []);
app.filter('myFilter', function() {
        return function(<params for custom filter>) {
        // Your code Goes Here
}
});
```

- In view template:

```
{{<value> | myFilter }}
```

index.html

```
<html>
<head>
<script type="text/javascript" src="script/angular.min.js"></script>
</head>
<body>
        <div data-ng-app="myApp">
                <br /> Enter Id <input type="text" data-ng-model="id" /> <br />
                <br/>Your Id is : {{ id}}
                <br/>Your Filtered Id is : {{ id | myid }}
                <br/>Your Filtered Id is : {{ id | uppercase | myid }}
        </div>
        <script type="text/javascript">
                var app = angular.module("myApp", []);
                app.filter('myid', function() {
                        return function(id) {
                                if (id == undefined || id.length == 0)
                                        return "";
                                else
                                        return 'CC-' + id;
                        }
                });
        </script>
</body>
</html>
```

Scanned by CamScanner

index.html

```
<html>
<head>
<script type="text/javascript" src="script/angular.min.js"></script>
</head>
<body>
        <div data-ng-app="myApp" data-ng-controller='myController'>
                <table width='60%'>
                        <tr>
                                <td>Sr. No.</td>
                                <td>ID</td>
                                <td>NAME</td>
                                <td>EMAIL</td>
                        </tr>
    <tr data-ng-repeat="st in students" style="background-color: {{ $index | colorFilter}}">
                                <td>{{$index+1}}</td>
                                <td>{{st.id}}</td>
                                <td>{{st.name}}</td>
                                <td>{{st.email}}</td>
                        </tr>
                </table>
        </div>
        <script type="text/javascript">
                var app = angular.module("myApp", []);
                app.controller('myController', function($scope) {
                        $scope.students = [ {
                                id : 44,
                                name : 'Manish',
                                email : 'manish@gmail.com'
                        }, {
                                id : 88,
                                name : 'Ranjan',
                                email : 'ranjan@gmail.com'
                        } ];
                });

                app.filter("colorFilter", function() {
                        return function(index) {
                                if (index % 2 == 0)
                                        return "lightblue";
                                else
                                        return "lightyellow";
                        }
                });
        </script>
</body>
</html>
```

Scanned by CamScanner

Java Learning Center

Angular JS

Put Into Knowledge & Practice

Doc Version - 1.0

Part 2

Author
Srinivas Dande

Scanned by CamScanner

14. Angular JS Services

- AngularJS services are objects that are wired together using dependency injection (DI).
- You can use services to organize and share code across your app.
- Services are:
  - Lazily instantiated – AngularJS only instantiates a service when an application component depends on it.
  - Singletons – Each component dependent on a service gets a reference to the single instance generated by the service factory.
- AngularJS offers several useful services (like $http).
- When required, you can create your own services also.

Using a Service

- You need to add the services as a dependency for the component (controller, service, filter or directive) to use an AngularJS service.

$location

- Location service has methods which return information about the location of the current web page.

index.html

```
<html>
<head>
<script type="text/javascript" src="script/angular.min.js"></script>
</head>
<body>
        <div data-ng-app="myApp" data-ng-controller="myCtrl">
<br/> <input type="button" value="SHOW INFO" data-ng-click="showUrlInfo()"/>
<br/> <input type="button" value="CHANGE" data-ng-click="changeUrlInfo()"/>
<br/>
                <div>Page URL is {{myUrl}}</div>
                <div>Page Host is {{myHost}}</div>
                <div>Page Port is {{myPort}}</div>
                <div>Page Path is {{myPath}}</div>
                <div>Page Search is {{mySearch}}</div>
                <div>Page Hash is {{myHash}}</div>
        </div>
        <script>
                var app = angular.module('myApp', []);
                app.controller('myCtrl', function($scope, $location) {
                        $scope.showUrlInfo = function() {
                                $scope.myUrl = $location.absUrl();
                                $scope.myHost = $location.host();
```

Scanned by CamScanner

```
                $scope.myPort = $location.port();
                $scope.myPath = $location.path();
                $scope.mySearch = $location.search();
                $scope.myHash = $location.hash();
            };
            $scope.changeUrlInfo = function() {
                $location.url('/hello?ab=123#middle');
                $location.path('/hello').search({ab: 123}).hash('middle');
            }
        });
    </script>
</body>
</html>
```

$interval

- $interval service of AngularJS' represents the functionality of window.setInterval.

```
<html> <head>
<script type="text/javascript" src="script/angular.min.js"></script>
</head>
<body>
    <div data-ng-app="myApp" data-ng-controller="myCtrl">
        <div>
            <h3>{{theTime}}</h3>
        </div>
    </div>
    <script>
        var app = angular.module('myApp', []);
        app.controller('myCtrl', function($scope, $interval) {
            $scope.theTime = new Date().toLocaleTimeString();
            $interval(function() {
                $scope.theTime = new Date().toLocaleTimeString();
            }, 1000);
        });
    </script>
</body> </html>
```

Scanned by CamScanner

$timeout

- $timeout service of AngularJS' represents the functionality of window.setTimeout.

index.html

```
<html>
<head>
<script type="text/javascript" src="script/angular.min.js"></script>
</head>
<body>
	<div data-ng-app="myApp" data-ng-controller="myCtrl">
		<div>
			<h3>{{message}}</h3>
		</div>
	</div>
	<script>
		var app = angular.module('myApp', []);
		app.controller('myCtrl', function($scope, $timeout) {
			$scope.message = "Hello World!";
			$timeout(function() {
				$scope.message = "How are you today?";
			}, 2000);
		});
	</script>
</body>
</html>
```

$window

- $window provides reference to the browser's window object.

index.html

```
<html> <head>
<script type="text/javascript" src="script/angular.min.js"></script>
</head>
<body>
	<div data-ng-app="myApp" data-ng-controller="myCtrl">
		<input type="text" data-ng-model="name"/>
		<button data-ng-click="showMessage()">ALERT</button>
	</div>
	<script>
		var app = angular.module('myApp', []);
```

Scanned by CamScanner

```
                app.controller('myCtrl', [ '$scope', '$window',
                        function($scope, $window) {
                                $scope.name = 'Sri';
                                $scope.showMessage = function() {
                                        $window.alert('Welcome '+$scope.name);
                                };
                        } ]);
        </script>
</body>
</html>
```

index.html

```
<html> <head>
<script type="text/javascript" src="script/angular.min.js"></script>
</head>
<body>
        <div data-ng-app="myApp" data-ng-controller="myCtrl">
                <br />
                <input type="button" value="CHANGE" data-ng-click="changeUrlInfo()" />
        </div>
        <script>
                var app = angular.module('myApp', []);
                app.controller('myCtrl', function($scope, $location,$window) {
                        $scope.changeUrlInfo = function() {
                                $window.location.href= 'http://google.com'
                        }
                });
        </script>
</body>
</html>
```

index.html

```
<html> <head>
<script type="text/javascript" src="script/angular.min.js"></script>
</head> <body>
        <div data-ng-app="myApp" data-ng-controller="myCtrl">
        <button data-ng-click="showMessage()">Confirm</button>
        </div>
        <script>
                var app = angular.module('myApp', []);
                app.controller('myCtrl', [ '$scope', '$window', function($scope, $window) {
```

Scanned by CamScanner

```
$scope.showMessage = function() {
        var res= $window.confirm("Are you sure to proceed");
        if(res) {
                $window.alert('You have selected to proceed');
        } else{
                $window.alert('You have selected not to proceed');
        }
};
} ]);
</script>
</body> </html>
```

Creating Services

- An AngularJS service can be created or registered or created in following different ways,
  - Using the service() method
  - Using the factory() method
  - Using the provider() method
- You can define your own services by registering the service's name and service factory function, with an AngularJS module.
- The service factory function generates the single object or function that represents the service to the rest of the application.
- The object or function returned by the service is injected into any component (controller, service, filter or directive) that specifies a dependency on the service.

Using the service() method

```
var app = angular.module('myApp', []);
app.service('calculator', function() {
  this.calculate = function (a, b) {
    return a * b;
  }
});
```

Using the factory () method

```
var app = angular.module('myApp', []);
app.factory('calculator', function() {
        var factory = {};
        factory.calculate = function(a, b) {
                return a * b
        }
        return factory;
});
```

Scanned by CamScanner

Using the provider () method

```
var app = angular.module('myApp', []);
app.config(function($provide) {
        $provide.factory('calculator', function() {
                var factory = {};
                factory.calculate = function(a, b) {
                        return a * b
                }
                return factory;
        });
});
```

index.html

```
<html>
<head>
<script type="text/javascript" src="script/angular.min.js"></script>
</head>
<body>
        <div data-ng-app="myApp" data-ng-controller="myCtrl">
                <input type="text" data-ng-model='a' /> <input type="text"
                        data-ng-model='b' /> <input type="button" value='RESULT'
                        data-ng-click='showResult()' /> <br /> {{result}}
        </div>
        <script>
                var app = angular.module('myApp', []);

                app.service('calculator', function() {
                        this.calculate = function(a, b) {
                                return a * b;
                        }
                });
                app.controller('myCtrl', function($scope, calculator) {
                        $scope.showResult = function() {
                                $scope.result = calculator.calculate($scope.a, $scope.b);
                        };
                });
        </script>
</body>
</html>
```

Scanned by CamScanner

index.html

```
<html> <head>
<script type="text/javascript" src="script/angular.min.js"></script>
</head>
<body>
	<div data-ng-app="myApp" data-ng-controller="myCtrl">
		<input type="text" data-ng-model='a' /> <input type="text"
			data-ng-model='b' /> <input type="button" value='RESULT'
			data-ng-click='showResult()' /> <br /> {{result}}
	</div>
	<script>
		var app = angular.module('myApp', []);
		app.factory('calculator', function() {
			var factory = {};
			factory.calculate = function(a, b) {
				return a * b
			}
			return factory;
		});
		app.controller('myCtrl', function($scope, calculator) {
			$scope.showResult = function() {
				$scope.result = calculator.calculate($scope.a, $scope.b);
			};
		});
	</script>
</body> </html>
```

index.html

```
<html> <head>
<script type="text/javascript" src="script/angular.min.js"></script>
</head>
<body>
	<div data-ng-app="myApp" data-ng-controller="myCtrl">
		<input type="text" data-ng-model='a' /> <input type="text"
			data-ng-model='b' /> <input type="button" value='RESULT'
			data-ng-click='showResult()' /> <br /> {{result}}
	</div>
	<script>
		var app = angular.module('myApp', []);
```

Scanned by CamScanner

```
app.config([ '$provide', function($provide) {
        $provide.factory('calculator', function() {
                var factory = {};
                factory.calculate = function(a, b) {
                        return a * b
                }
                return factory;
        });
} ]);
app.controller('myCtrl', function($scope, calculator) {
        $scope.showResult = function() {
                $scope.result = calculator.calculate($scope.a, $scope.b);
        };
});
</script>
</body>
</html>
```

The difference between services and factories

- .service() is a method on module that takes a name and a function that defines the service.
- We can inject and use the particular service in other components, like controllers, directives and filters.

```
var app = angular.module('myApp', []);
app.service('calculator', function() {
  this.calculate = function (a, b) {
    return a * b;
  }
});
```

- .factory() is a method on module and it also takes a name and a function, that defines the factory.
- We can inject and use the particular this thing exactly the same way as service.

```
var app = angular.module('myApp', []);
app.factory('calculator', function() {
        var factory = {};
        factory.calculate3 = function(a, b) {
                return a * b
        },
```

Scanned by CamScanner

```
        factory.calculate2 = function(a, b) {
                return a + b
        }
        return factory;
});
```

OR

```
app.factory('calculator', function() {
        return {
                calculate3: function(a, b) {
                        return a + b
                },
                calculate2: function(a, b) {
                        return a + b
                }
        }
});
```

- Instead of working with this in the factory, we're returning an object literal.
- A service is a constructor function whereas a factory is not.
- In the case of service, Object.create() will be called with the service constructor function, when it gets instantiated.
- A factory function is really just a function that gets called, which is why we have to return an object explicitly.

Module Dependencies

index.html

```
<html>
<head>
<script type="text/javascript" src="script/angular.min.js"></script>
</head>
<body>
        <div data-ng-app="myApp" data-ng-controller="myCtrl">
                <input type="text" data-ng-model='a' />
                <input type="button" value='RESULT' data-ng-click='showResult()' />
                <br /> {{result}}
        </div>
```

Scanned by CamScanner

```
<script>
    var calcApp = angular.module('CalculatorModule', []);
    calcApp.service('calculator', function() {
        this.square = function(a) {
            return a * a
        };
    });

    var app = angular.module('myApp', [ 'CalculatorModule' ]);
    app.controller('myCtrl', function($scope, calculator) {
        $scope.showResult = function() {
            $scope.result = calculator.square(parseInt($scope.a));
        };
    });
</script>
</body> </html>
```

$http

- The $http service is a core AngularJS service that is used to communicate with the remote HTTP servers via the browser's XMLHttpRequest object or via JSONP.
- The $http service is a function
  - Which takes a single argument **a configuration object** that is used to generate an HTTP request
  - Which returns a promise.

```
...
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-webmvc</artifactId>
<version>4.1.7.RELEASE</version>
</dependency>

<dependency>
<groupId>com.fasterxml.jackson.core</groupId>
<artifactId>jackson-databind</artifactId>
<version>2.4.6</version>
</dependency>
...
```

Scanned by CamScanner

AppConfig.java

```
package com.jlcindia.anguarjs.config;

import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;

@Configuration
@EnableWebMvc
@ComponentScan(basePackages = "com.jlcindia.anguarjs")
public class AppConfig {

}
```

AppInitializer.java

```
package com.jlcindia.anguarjs.config;

import
org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;

public class AppInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {

	@Override
	protected Class<?>[] getRootConfigClasses() {
		return new Class[] { AppConfig.class };
	}

	@Override
	protected Class<?>[] getServletConfigClasses() {
		return new Class[] { AppConfig.class };
	}

	@Override
	protected String[] getServletMappings() {
		return new String[] { "/services/*" };
	}

}
```

Scanned by CamScanner

Student.java

```
package com.jlcindia.anguarjs.config.domain;

public class Student {
        private int sid;
        private String name;
        private String email;

        public Student() {
        }
        public Student(int sid, String name, String email) {
                this.sid = sid;
                this.name = name;
                this.email = email;
        }
// Setters and Getters
}
```

User.java

```
package com.jlcindia.anguarjs.config.domain;

public class User {
        private int userId;
        private String name;
        private String email;
// Setters and Getters
}
```

StudentService.java

```
package com.jlcindia.anguarjs.service;

import java.util.ArrayList;
import java.util.List;
import org.springframework.stereotype.Service;
import com.jlcindia.anguarjs.config.domain.Student;
@Service
public class StudentService {
        private static List<Student> students = new ArrayList<>();
        static {
                students.add(new Student(99,"Sri","sri@gmail.com"));
```

Scanned by CamScanner

```
        students.add(new Student(88,"Manish","manish@gmail.com"));
        students.add(new Student(77,"Rahul","rahul@gmail.com"));
        students.add(new Student(44,"Kumar","kumar@gmail.com"));
        students.add(new Student(66,"Dande","dande@gmail.com"));
        students.add(new Student(55,"Nivas","nivas@gmail.com"));
        students.add(new Student(101,"Ranjan","ranjan@gmail.com"));
    }
    public Student getStudent(int sid) {
        try{
        return students.stream().filter(st -> st.getSid() == sid).findFirst().get()
        }catch(Exception e){
            return null;
        }
    }
    public boolean add(Student st) {
        return students.add(st);
    }
    public boolean remove(int sid) {
        return students.removeIf(st -> st.getSid() == sid);
    }
    public List<Student> getAll() {
        return students;
    }

}
```

HelloController.java

```
package com.jlcindia.anguarjs.controller;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;
import com.jlcindia.anguarjs.config.domain.User;

@RestController
@RequestMapping("/hello")
public class HelloController {
```

Scanned by CamScanner

```
@RequestMapping(value="/message",produces = { "text/plain" })
public String getMessage() {
System.out.println("HelloController -> getMessage");
return "Welcome to Angular $http Service world";
}

@RequestMapping(value="/message/{name}",produces = { "text/plain"})
public String getMessage(@PathVariable String name) {
return "Hi " + name + ", Welcome to Angular $http Service world";
}

@RequestMapping(value = "/user", consumes = { "application/json" }, produces = {
"application/json" }, method = RequestMethod.POST)
public User getMessage(@RequestBody User user) {
System.out.println(user.getUserId() +", "+ user.getName()+", "+user.getEmail());
user.setUserId(99);
user.setName("Srinivas");
user.setEmail("srinivas@jlc.com");
return user;
}
}
```

StudentController.java

```
package com.jlcindia.anguarjs.controller;

import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;
import com.jlcindia.anguarjs.config.domain.Student;
import com.jlcindia.anguarjs.config.domain.User;
import com.jlcindia.anguarjs.service.StudentService;

@RestController
@RequestMapping("/student")
public class StudentController {
```

Scanned by CamScanner

```
@Autowired
private StudentService service;

@RequestMapping(value = "/{sid}", produces = { "application/json" }, method =
RequestMethod.GET)
public Student getStudent(@PathVariable int sid) {
return service.getStudent(sid);
}

@RequestMapping(consumes = { "application/json" }, produces = { "application/json" },
method = RequestMethod.POST)
public boolean add(@RequestBody Student st) {
return service.add(st);
}

@RequestMapping(value = "/{sid}",produces = { "application/json" }, method =
RequestMethod.DELETE)
public boolean remove(@PathVariable int sid) {
return service.remove(sid);
}

@RequestMapping(produces = { "application/json" }, method = RequestMethod.GET)
public List<Student> getAll() {
return service.getAll();
}
}
```

hello.js

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

app.controller('helloController', function($scope, $http,$location) {
        $scope.message = "";
        $scope.messageWithName = "";
        $scope.getMessageFromServer = function() {
                var config = {
                        method : "GET",
                        url : $location.absUrl() + '/services/hello/message',
                        responseType:'text'
                };
                var httpRequest = $http(config);
                httpRequest.then(
                                function successCallback(response) {
                console.log('helloController - showMessage - success '+ response.data);
                                        $scope.message = response.data;
```

Scanned by CamScanner

```
                    }, function errorCallback(response) {
            console.log('helloController - showMessage - failed '+ response);
                        $scope.message = response;
                    });
    };
    $scope.getMessageFromServerWithName = function(name) {
        var config = {
            method : "GET",
            url : $location.absUrl() + '/services/hello/message/'+name,
            responseType:'text'
        };
        var httpRequest = $http(config);
        httpRequest.then(
                    function successCallback(response) {
console.log('helloController - showMessageWithName - success '+ response.data);
                        $scope.messageWithName = response.data;
                    }, function errorCallback(response) {
console.log('helloController - showMessageWithName - failed '+ response);
                        $scope.messageWithName = response;
                    });
    };
    $scope.getUser = function(id,sname,semail) {
        var config = {
            method : "POST",
            url : $location.absUrl() + '/services/hello/user',
            data: { userId: id, name : sname, email : semail }
        };
        var httpRequest = $http(config);
        httpRequest.then(
                    function successCallback(response) {
            console.log('helloController - getUser - success '+ response.data);
                        $scope.user = response.data;
                    }, function errorCallback(response) {
                    console.log('helloController - getUser - failed '+ response);
                        $scope.user = response;
                    });
    };
});
```

Scanned by CamScanner

index.html

```
<html>
<head>
<script type="text/javascript" src="script/angular.min.js"></script>
<script type="text/javascript" src="script/hello.js"></script>
</head>
<body>

        <div data-ng-app="helloApp" data-ng-controller='helloController'>
                <input type="button" data-ng-click='getMessageFromServer()' value='Show
Message' />
                <br/>Message from Server {{message}}
                <hr />
                <br/>
                <input type="text" data-ng-model='name' />
                <input type="button" data-ng-
click='getMessageFromServerWithName(name)' value='Show Message' />
                <br/>Message from Server with name {{messageWithName}}

                <hr />
                <br/>
                <input type="text" data-ng-model='sid' />
                <input type="text" data-ng-model='sname' />
                <input type="text" data-ng-model='semail' />

                <input type="button" data-ng-click='getUser(sid,sname,semail)'
value='Show User' />
                <br/>User Id {{user.userId}}
                <br/>Name {{user.name}}
                <br/>Email {{user.email}}
        </div>

</body>
</html>
```

Scanned by CamScanner

Student.js

```
var sApp = angular.module('studentApp', []);

sApp.controller('studentController', function($scope, $http) {
        $scope.getStudent = function(sid) {
                var config = {
                        method : "GET",
                        url : 'http://localhost:8080/angular-js-ws/services/student/'+sid,
                };
                var httpRequest = $http(config);
                httpRequest.then(
                                function successCallback(response) {
                console.log('studentController - getStudent - success '+ response.data);
                                        $scope.resultData = response.data;
                                }, function errorCallback(response) {
                        console.log('studentController - getStudent - failed '+ response);
                                        $scope.resultData = response;
                                });
        };

        $scope.getAll = function() {
                var config = {
                        method : "GET",
                        url : 'http://localhost:8080/angular-js-ws/services/student/',
                };
                var httpRequest = $http(config);
                httpRequest.then(
                                function successCallback(response) {
                        console.log('studentController - getAll - success '+ response.data);
                                        $scope.getAllResult = response.data;
                                }, function errorCallback(response) {
                        console.log('studentController - getAll - failed '+ response);
                                        $scope.getAllResult = response;
                                });
        };

        $scope.deleteStudent = function(sid) {
                var config = {
                        method : "DELETE",
                        url : 'http://localhost:8080/angular-js-ws/services/student/'+sid,
                };
```

Scanned by CamScanner

```
            var httpRequest = $http(config);
            httpRequest.then(
                        function successCallback(response) {
            console.log('studentController - deleteStudent - success '+ response.data);
                                $scope.deleteResult = response.data;
                        }, function errorCallback(response) {
                console.log('studentController - deleteStudent - failed '+ response);
                                $scope.deleteResult = response;
                        });
        };

        $scope.addStudent = function(id,sname,semail) {
            var config = {
                method : "POST",
                url : 'http://localhost:8080/angular-js-ws/services/student/',
                data: { sid: id, name : sname, email : semail }
            };
            var httpRequest = $http(config);
            httpRequest.then(
                        function successCallback(response) {
            console.log('studentController - addStudent - success '+ response.data);
                                $scope.addResult = response.data;
                        }, function errorCallback(response) {
                console.log('studentController - addStudent - failed '+ response);
                                $scope.addResult = response;
                        });
        };
});
```

student.html

```
<html> <head>
<script type="text/javascript" src="script/angular.min.js"></script>
<script type="text/javascript" src="script/student.js"></script>
</head>
<body>
        <div data-ng-app="studentApp" data-ng-controller='studentController'>
          Sid <input type="text" data-ng-model='sid' />
          <input type="button" data-ng-click='getStudent(sid)' value='Search Student' />
            <div data-ng-if='!resultData'>No Record Found</div>
            <div data-ng-if='resultData'>
```

Scanned by CamScanner

```
                <br />Sid <span data-ng-bind='resultData.sid'></span>
                <br />Name <span data-ng-bind='resultData.name'></span>
                <br />Email <span data-ng-bind='resultData.email'></span>
            </div>
            <hr />
            <input type="button" data-ng-click='getAll()' value='Show All' />
            <div data-ng-if='getAllResult'>
                <table>
                    <tr>
                        <td>Sid</td>
                        <td>Name</td>
                        <td>Email</td>
                    </tr>
                    <tr data-ng-repeat='st in getAllResult'>
                        <td>{{st.sid}}</td>
                        <td>{{st.name}}</td>
                        <td>{{st.email}}</td>
                    </tr>
                </table>
            </div>
            <hr />
            Sid <input type="text" data-ng-model='sid1' />
            <input type="button" data-ng-click='deleteStudent(sid1)' value='Remove' />
            <div data-ng-switch on='deleteResult'>
                <div data-ng-switch-when='true'>Student Deleted</div>
                <div data-ng-switch-when='false'>Student Not Deleted</div>
            </div>
            <hr />
            <input type="text" data-ng-model='sid2' />
            <input type="text" data-ng-model='sname' />
            <input type="text" data-ng-model='semail' />
        <input type="button" data-ng-click='addStudent(sid2,sname,semail)' value='Add' />
            <div data-ng-switch on='addResult'>
                <div data-ng-switch-when='true'>Student Added</div>
                <div data-ng-switch-when='false'>Student Not Added</div>

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

Scanned by CamScanner