



AngularJS Tutorial

Hery Vandro

<https://www.facebook.com/vandro>

<https://www.mahirkoding.com/>

vandrohery99@gmail.com

DAFTAR ISI

Installation	3
Modules.....	4
Directive	4
Expressions.....	6
Model and Data Bindings.....	7
Controllers.....	7
Scope	7
Filters	9
Tables.....	11
AngularDOM.....	13
Angular Events.....	13
Forms and Validations.....	15
Routings & Views	18
Cookies.....	21
Value, Factory dan Services	22

AngularJS Tutorial

AngularJS adalah salah satu framework javascript buatan Google yang cukup terkenal yang banyak digunakan saat ini untuk menangani proses di bagian front end. AngularJS memungkinkan kita untuk membangun web apps yang interaktif dengan konsep arsitektur **MVC** (Model View Controller) dan **MVVM** (Model View – ViewModel).

Hingga saat tutorial ini ditulis, Angular telah keluar dalam 3 versi utama. Versi pertama dikenal dengan AngularJS. Sedangkan versi kedua lebih dikenal dengan sebutan Angular saja dan versi yang satu ini sudah tergolong jenis Typescript (<https://en.wikipedia.org/wiki/TypeScript>). Yang ketiga, baru baru ini google juga telah merilis Angular versi 4.0. Dalam pembahasan tutorial ini, saya akan menggunakan AngularJS (v1).

Dengan AngularJS, kita bisa membuat **SPA (Single Page Applications)** lebih efisien serta masih banyak kelebihan lain yang bisa dilakukan dengan AngularJS.

Tutorial singkat ini ditujukan bagi pemula yang tertarik untuk belajar AngularJS. Ebook ini dibagikan secara gratis dan bebas untuk dishare kembali bagi yang membutuhkan.

Installation

AngularJS homepage : <https://angularjs.org/>

Download AngularJS

Branch

1.6.x (latest)1.2.x (legacy)

Build

MinifiedUncompressedZip

CDN

<https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js>

Bower

bower install angular#1.6.4

npm

npm install angular@1.6.4

Extras

[Browse additional modules](#)

Previous Versions

Download

AngularJS dapat diinstall dengan cara mendownloadnya via **official web AngularJS** dan menguploadnya seperti biasa atau langsung menggunakan link **CDN**.

```
<!DOCTYPE html>
<html>
<head>
  <title>AngularJS - Tutorial</title>
  <script type="text/javascript" src="js/angular.min.js"></script>
</head>
```

```
<script type="text/javascript" src="https://ajax.googleapis.com/ajax/
libs/angularjs/1.6.4/angular.min.js"></script>
```

Modules

Modul dalam AngularJS digunakan untuk mendefinisikan scope tag tersebut menggunakan AngularJS. Oleh karena itu, pendefinisian module biasanya diletakkan di tag body sebagai root element dari sebuah html. Module juga berfungsi sebagai **container** bagi controller nantinya.

AngularJS modules dideklarasikan menggunakan directive **ng-app**. (akan dijelaskan di materi berikutnya).

Directive

Directive adalah sebuah **attribute** yang biasanya berawalan dengan prefix **ng-** dan berfungsi sebagai marker layaknya class dan id ataupun sebagai event driven. Bisa dikatakan juga bahwa directive adalah **tag-khususnya** si AngularJS. Ada beberapa macam directive yang akan kita gunakan selama pembahasan ini, untuk lebih lengkapnya, bisa dipelajari di dokumentasi AngularJS.

Berikut adalah contoh sederhana penggunaan directive :

```
<!DOCTYPE html>
<html>
<head>
  <title>AngularJS - Tutorial</title>
  <script type="text/javascript" src="js/angular.min.js"></script>
</head>
<body ng-app="">
  <input type="text" ng-model="name">
  <h1>Hello, {{ name }}</h1>
</body>
</html>
```

Directive **ng-app** digunakan untuk mendeklarasikan bahwa scope **body** menggunakan **AngularJS**. Tanpa deklarasi ng-app, maka AngularJS **tidak akan berjalan**.

directive **ng-model** digunakan untuk mendeklarasikan menghubungkan antara view dengan controller. Ng-model juga bisa digunakan sebagai perantara **two way binding**.

{{ name }} digunakan untuk mengeluarkan isi sebuah data, dalam kasus ini yang di output adalah data dari model name.

See result here :

https://bit.ly/hery_angular_1

```
<!DOCTYPE html>
<html>
<head>
  <title>AngularJS - Tutorial</title>
  <script type="text/javascript" src="js/angular.min.js"></script>
</head>
<body ng-app="">
  <input type="text" ng-model="name" ng-init="name='John Doe'">
  <h1 ng-bind="name"></h1>
</body>
</html>
```

Jika ingin menginisialisasi **nilai awal** model **name**, bisa menggunakan directive **ng-init** seperti contoh diatas. Inisialisasi data model tidak selamanya melalui directive **ng-init**, namun juga bisa lewat controller yang akan dijelaskan nanti.

Proses mengeluarkan isi dari model name bisa dilakukan dengan 2 cara, bisa dengan menggunakan scope `{{ }}` ataupun menggunakan directive ng-bind. **Hasilnya sama saja.**

Untuk penggunaan directive sendiri dapat dilakukan dengan **4 metode** :

- **Element Directive** - `<my-directive></my-directive>` - (E)
- **Attribute Directive** - `<div my-directive></div>` - (A)
- **Class Directive** - `<div class="my-directive"></div>` - (C)
- **Comment Directive** `<!--directive:my-directive-->` - (M)

Di AngularJS, kita juga bisa membuat custom directive sesuai keinginan kita. Misalnya :

```
<body ng-app="myApp" ng-controller="testCtrl">
  <my-directive></my-directive>
</body>
<script type="text/javascript">
  var app = angular.module("myApp", []);

  app.directive("myDirective", function(){
    return {
      restrict : 'EA',
      template : "<h1>{{ material }} Lesson!</h1>"
    }
  })

  app.controller("testCtrl", function($scope){
    $scope.material = "Javascript";
  })
</script>
```

Custom directive yang kita buat akan menampilkan “**Javascript Lesson!**” di bagian yang telah kita deklarasikan. Lalu, perlu diketahui bahwa **naming convention** di AngularJS menggunakan aturan **lowerCamelCase** sehingga **huruf pertama pada kata kedua** menggunakan huruf **kapital**.

Property **restrict** ‘**EA**’ berfungsi untuk mendeklarasikan bahwa directive **my-directive** hanya boleh digunakan sebagai **Element Directive** dan **Attribute Directive**. Jika ingin menggunakan **my-directive** sebagai **Comment Directive**, berarti kita tinggal menambahkan **simbol M**, sehingga menjadi : **EAM**.

Secara **default**, jika property **restrict** tidak dibuat maka custom directive kita hanya dapat digunakan pada mode **EA**.

Javascript Lesson!

Expressions

Expressions dalam AngularJS bertugas untuk mengeluarkan output di html, sama seperti **printf** ataupun **echo** di **php**. **Expressions** dalam AngularJS sangat mirip dengan **javascript**, kita bisa juga melakukan operasi **string**, **number**, **array**, **object**, dll.

```
<body ng-app="">
  <div style="background-color: {{ colors }}; width: 200px;">
    <input type="text" ng-model="colors">
    <br>
    <!-- Numbers -->
    <span>{{ 10 * 20 }}</span>
    <br>
    <!-- String -->
    <span>{{ "Umur Saya : " + 19 }}</span>
    <br>
    <!-- Arrays -->
    <span ng-init="majors=['Teknik Informatika', 'Sistem Informasi',
      'Hotel Management']">
      {{ "Major : " + majors[0] }}
    </span>
    <br>
    <!-- Objects -->
    <span ng-init="user={name:'John', age:19}">
      {{ "Name : " + user.name + ", Age : " + user.age }}
    </span>
  </div>
</body>
```

See result here :

https://bit.ly/hery_angular_2

Model and Data Bindings

Salah satu kemampuan lain AngularJS adalah memungkinkan adanya **two way data binding**. Disini, kamu bisa **mensinkronisasikan** data dari **view** dengan **controller** begitu pula **sebaliknya**. Untuk memungkinkan hal ini terjadi, maka kita membutuhkan directive **ng-model**. Setiap perubahan yang terjadi pada model, maka view juga dapat **langsung** terkena dampak perubahannya. Contoh untuk model dan data bindings sudah dijelaskan di materi sebelumnya.

Controllers

Sesuai namanya, controller bertugas untuk mengontrol data dan alur kerja dari web yang kita buat. Konsep **MVC** menekankan kita untuk memisah semua alur proses program ke dalam **controller-controller** yang berbeda.

```
<body ng-app="myApps">
  <div ng-controller="userController">
    <input type="text" ng-model="name">
    <input type="submit" ng-click="showResult()">
  </div>
</body>
<script type="text/javascript" >
  var app = angular.module("myApps", []);
  app.controller("userController", function($scope){
    $scope.name = "Dummy Name";
    $scope.showResult = function(){
      alert($scope.name);
    }
  });
</script>
```

Contoh diatas, kita akan membuat **userController** untuk menginisialisasi **model name** dan membuat function **showResult** untuk melakukan **alert** isi model name.

Pertama-tama, kita harus membuat variable angular module untuk "**myApps**". **Scope []** dalam inisialisasi angular module digunakan untuk menampung external dependencies jika diperlukan nantinya.

Lalu, apa itu **variable scope**?

Scope

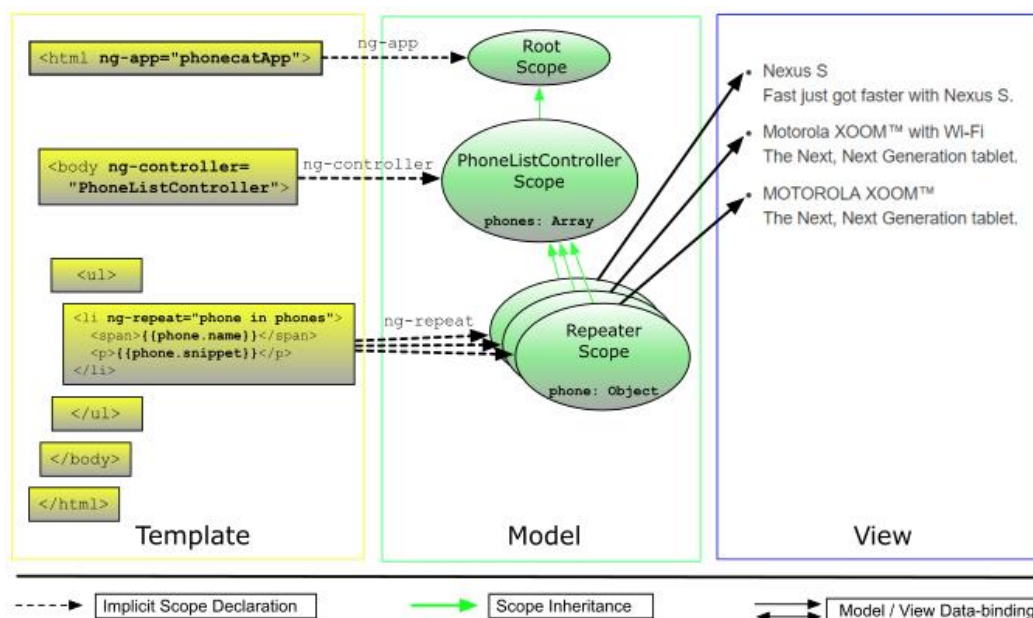
Scope adalah sebuah object yang menampung model, sebagai perantara bagi controller dan view. **Object** tersebut dapat berisi method dan variabels. Coba perhatikan kembali snippet code diatas, kita bisa mengakses seluruh **model** dan **function** dalam **userController** lewat variable scope.

Perlu diketahui, **\$scope** tidak hanya digunakan untuk mengambil/menampung variable model yang sudah ada. Namun kita juga bisa mendeklarasikannya sendiri **tanpa harus** menambahkan directive **ng-model** yang baru.

```
<body ng-app="myApps">
  <div ng-controller="userController">
    <input type="text" ng-model="name">
    <input type="submit" ng-click="showResult()">
  </div>
</body>
<script type="text/javascript">
  var app = angular.module("myApps", []);
  app.controller("userController", function($scope){
    $scope.degree = "S. Kom";
    $scope.name = "Dummy Name";
    $scope.showResult = function(){
      alert($scope.name+", "+$scope.degree);
    }
  });
</script>
```

Cara menggunakan **\$scope** cukup dengan **mempassing** **\$scope** di dalam function controller.

Jika ingin mengambil **seluruh object dari luar** controller yang berbeda, bisa gunakan **\$rootScope**. Scope yang ada dalam object **\$rootScope** mencakup keseluruhan dari isi app. (start from ng-app section)



Filters

Filters di AngularJS digunakan untuk **memformat data** ke output yang kita inginkan. Beberapa filters yang sering digunakan di AngularJS :

- **lowercase** – membuat string menjadi huruf kecil
- **uppercase** – membuat string menjadi huruf capital
- **orderBy** – mengurutkan data (kasih tanda – jika ingin mengurutkan descending)
- **filter** – menfilter data yang tampil
- **limitTo** – limit array/string yang tampil berdasarkan jumlah karakter/elemennya.
- **currency** – mengganti format angka menjadi mata uang
- **json** – memformat output dalam bentuk json
- **date** – memformat output dalam bentuk tanggal
- **dan lainnya** – <https://docs.angularjs.org/api/ng/filter>

```
<body ng-app="myApps">
  <div ng-controller="digimonController">
    <input type="text" ng-model="keyword">
    <ul>
      <li ng-repeat="digi in digimons | filter:keyword |
        orderBy:'-attack'">
        {{ digi.attack }} | {{ digi.name | uppercase |
        limitTo:5 }}
      </li>
    </ul>
  </div>
</body>
<script type="text/javascript" >
  var app = angular.module("myApps", []);
  app.controller("digimonController", function($scope){
    $scope.digimons = [
      {name:"Agumon", attack:200},
      {name:"Dukemon", attack:250},
      {name:"Leomon", attack:110},
      {name:"Calmaron", attack:90},
      {name:"Sakuyamon", attack:50},
      {name:"Salmon", attack:120},
    ];
  });
</script>
```

Directive **ng-repeat** digunakan untuk membuat **perulangan** di AngularJS. Aturan penggunaannya mirip seperti **foreach** di beberapa Bahasa pemrograman lainnya.

Filter "**filter**" akan membuat data yang ditampilkan **hanyalah** data yang memenuhi kriteria yang telah kita ketikkan di dalam inputan **keyword**.

Sedangkan filter "**orderBy**" akan membuat data yang tampil berurutan. Secara default, data akan diurutkan **ascending**. Jika ingin mengurutkan **descending**, maka tinggal menambahkan **tanda –** di depan argument.

See result here :

https://bit.ly/hery_angular_3

Untuk penggunaan **ng-repeat**, kita juga akan mengenal beberapa variable, diantaranya :

- **\$index** - Untuk mengetahui index element yang aktif
- **\$first** - Sebuah variabel boolean yang menandakan index pertama atau bukan.
- **\$middle** - Sebuah variabel boolean yang menandakan bukan index pertama dan bukan index terakhir.
- **\$last** - Sebuah variabel boolean yang menandakan index terakhir atau bukan.
- **\$odd** - Sebuah variabel boolean yang menandakan index ganjil
- **\$even** - Sebuah boolean yang menandakan index genap

```
<body ng-app="myApp" ng-controller="foodController">
  <div ng-repeat="food in foods">
    <p ng-show="$odd" style="background-color: yellow">
      <span>{{ food.foodName }}</span>
      <span>{{ food.price | currency }}</span>
      <span>{{ food.expiredDate | date }}</span>
    </p>
    <p ng-show="$even" style="background-color: green">
      <span>{{ food.foodName }}</span>
      <span>{{ food.price | currency }}</span>
      <span>{{ food.expiredDate | date }}</span>
    </p>
  </div>
</body>
<script type="text/javascript">
  var app = angular.module("myApp", []);
  app.controller("foodController", function($scope){
    $scope.foods = [
      {foodName : "Oreo Blue", price : 20, expiredDate : new Date(2018,5,12)},
      {foodName : "Oishi Pillows", price : 30, expiredDate : new Date(2017,10,11)},
      {foodName : "Biskuat", price : 15, expiredDate : new Date(2019,2,22)},
      {foodName : "Marine", price : 11, expiredDate : new Date(2020,4,19)},
      {foodName : "Tango", price : 12, expiredDate : new Date(2019,8,2)}
    ]
  })
</script>
```

Paragraph **kuning** akan muncul kalau data yang muncul adalah data dengan index **ganjil**. Sedangkan yang **hijau** adalah data dengan index **genap**.

NB : Index data dimulai dari 0.

Oreo Blue \$20.00 Jun 12, 2018

Oishi Pillows \$30.00 Nov 11, 2017

Biskuat \$15.00 Mar 22, 2019

Marine \$11.00 May 19, 2020

Tango \$12.00 Sep 2, 2019

Tables

Membuat tabel dengan bantuan AngularJS cukup dengan menggunakan **ng-repeat** terhadap elemen yang ingin diulang (**<tr>**). Berikut adalah contoh pembuatan table yang **dinamis** dengan angularJS.

```
<p>  
  Search : <input type="text" ng-model="keyword">  
</p>
```

Input keywords (untuk searching).

```
<thead>  
  <tr>  
    <th><input type="text" ng-model="name"></th>  
    <th>  
      <select ng-model="team" ng-options="team for team  
        in teams">  
      </select>  
    </th>  
    <th><input type="number" ng-model="goal"></th>  
    <th><button ng-click="insert()">Add</button></th>  
  </tr>  
  <tr>  
    <th>Name</th>  
    <th>Team</th>  
    <th>Goal</th>  
    <th>Action</th>  
  </tr>  
</thead>
```

Table Header berisi **form input** dan **column title** bagi setiap data.

Coba perhatikan untuk **combobox** teamnya, disini kita menggunakan directive **ng-options** untuk pemilihan datanya. Kenapa tidak menggunakan **ng-repeat**? Apa bedanya? Nah, kalau kita menggunakan **ng-options**, maka data yang kita select adalah **objectnya**. Jika menggunakan **ng-repeat**, kita hanya melakukan looping terhadap **isi objectnya** saja (string).

More : <https://docs.angularjs.org/api/ng/directive/ngOptions>

```
<tbody>  
  <tr ng-show="filteredData==0">  
    <td colspan="4">  
      No Data!  
    </td>  
  </tr>  
  <tr ng-repeat="player in players | filter:{ name:keyword }  
    as filteredData">  
    <td>{{ player.name }}</td>  
    <td>{{ player.team }}</td>  
    <td>{{ player.goal }}</td>  
    <td><a href="#" ng-click="delete($index)">Delete</a></td>  
  </tr>  
</tbody>
```

Untuk bagian table **body**, disini kita bisa membuat sebuah **row hidden** untuk menampilkan messages ketika **datanya kosong**. Caranya adalah dengan menggunakan directive **ng-show** pada kondisi **filteredData==0**. Variabel **filteredData** otomatis **terupdate** sesuai **jumlah data** yang telah **difilter** di bagian bawahnya. (**perhatikan bagian filter as**).

Untuk tombol **delete**, kita akan mengarahkannya ke **function** delete dan kita akan passing **index arraynya** melalui parameter. Function tersebut akan berjalan ketika telah **ditrigger** oleh directive **ng-click**.

```
<script type="text/javascript" >
  var app = angular.module("myApps", []);
  app.controller("footballController", function($scope){
    $scope.players = [
      {name:"Hazard", goal:20, team:"Chelsea"},
      {name:"Alexis Sanchez", goal:15, team:"Barcelona"},
      {name:"Joe Hart", goal:0, team:"Manchester City"},
      {name:"Ibrahimovic", goal:17, team:"Manchester United"},
      {name:"Lionel Messi", goal:5, team:"Barcelona"},
      {name:"Cristiano Ronaldo", goal:24, team:"Real Madrid"},
    ];

    $scope.teams = [
      "Chelsea", "Barcelona", "Manchester United", "Manchester City", "
      Real Madrid", "Arema", "Inter Milan"
    ];

    $scope.insert = function(){
      $scope.players.push({
        name: $scope.name, team : $scope.team, goal : $scope.goal
      });
      $scope.clearForm();
    };

    $scope.clearForm = function(){
      $scope.name = ""; $scope.team = ""; $scope.goal = "";
    }

    $scope.delete = function(index){
      $scope.players.splice(index, 1);
    }
  });
</script>
```

Untuk menambahkan isi sebuah array, kita bisa menggunakan method **.push()** disertai **object** yang ingin dimasukkan ke dalam array players.

Sedangkan untuk function delete, bisa dilakukan dengan method **.splice(startindex, count)**

See result here :

https://bit.ly/hery_angular_4

AngularDOM

AngularJS juga mempunyai fitur untuk mengatur sebuah elemen berdasarkan **value** form component.

Sebagai contoh simple adalah component **checkbox**, kita bisa mentrigger element lain berdasarkan hasil checkbox tersebut. Atau, kita juga bisa mengatur **visibility** dari sebuah element dengan menggunakan directive **ng-show** atau **ng-hide**.

```
<body ng-app="myApps">
  <div ng-controller="registerCtrl">
    <h1>Register</h1>
    <p>
      <input type="text" ng-model="name"
        placeholder="Insert Name">
    </p>
    <p>
      <input type="number" ng-model="age"
        placeholder="Insert Age">
    </p>
    <p ng-show="age<17">
      <input type="text" ng-model="name"
        placeholder="Insert Father Name">
    </p>
    <p>
      <input type="checkbox" ng-model="check">
        I Agree with Lisenca Agreement
    </p>
    <input type="submit" ng-disabled="!check">
  </div>
</body>
<script type="text/javascript">
  var app = angular.module("myApps", []);
  app.controller("registerCtrl", function($scope){
  });
</script>
```

See result here :

https://bit.ly/hery_angular_5

Angular Events

Kita juga bisa menambahkan event listener terhadap element html dengan directive-directive berikut :

Ng-blur	Ng-focus	Ng-mouseleave
Ng-change	Ng-keydown	Ng-mousemove
Ng-click	Ng-keypress	Ng-mouseover
Ng-copy	Ng-keyup	Ng-mouseup
Ng-cut	Ng-mousedown	Ng-paste
Ng-dblclick	Ng-mouseenter	

Berikut adalah sample untuk menginput **tag/kategori** dengan **tanda koma** sebagai **pemisah**. Tag yang telah diinput juga bisa **diremove** ketika **diclick**.

```
<style type="text/css">
  .result{
    max-width: 400px;
    height: 200px;
    box-sizing: border-box;
    padding: 5px;
  }
  .result span{
    cursor: pointer;
  }
  .input{
    max-width: 400px;
    border: 1px solid #000;
    box-sizing: border-box;
    padding: 10px;
  }
  .input input{
    padding: 5px;
    width: 80%;
  }
</style>
```

```
<body ng-app="myApps" ng-controller="boxCtrl">
  <div class="input">
    <input ng-keyup="check($event)" type="text" ng-model="tag" placeholder="Input tag and
    separate it with comma">
  </div>
  <div class="result" style="background-color: rgb({{ r+' '+g+' '+b }})">
    <span ng-click="remove($index)" ng-repeat="tag in tags">{{ tag + ", " }}</span>
  </div>
</body>
```

```
<script type="text/javascript" >
  var app = angular.module("myApps", []);
  app.controller("boxCtrl", function($scope){
    $scope.changeColor = function(){
      $scope.r = Math.floor(Math.random()*255);
      $scope.g = Math.floor(Math.random()*255);
      $scope.b = Math.floor(Math.random()*255);
    };
    $scope.tags = [];
    $scope.check = function(keyCode){
      if(keyCode.which==188){
        $scope.tag = ($scope.tag+ '').substr(0, ($scope.tag+ '').length-1);
        $scope.tags.push($scope.tag);
        $scope.tag = "";
      }
    }
    $scope.remove = function(index){
      $scope.tags.splice(index, 1);
      $scope.changeColor();
    }
    $scope.changeColor();
  });
</script>
```

See result here :

https://bit.ly/hery_angular_6

Forms and Validations

Untuk masalah validasi, di AngularJS juga sudah tersedia beberapa **build in function** untuk melakukan **validasi** walaupun tidak selengkap function yang ada di javascript.

- angular.isNumber
- angular.isString
- angular.isObject
- angular.isArray
- angular.isFunction
- angular.isDefined
- angular.isUndefined

```
<body ng-app="myApps">
  <div ng-controller="basicValid">
    isNumber? {{ isnumber }}<br>
    isString? {{ isstring }}<br>
    isObject? {{ isobject }}<br>
    isArray? {{ isArray }}<br>
    isFunction? {{ isfunction }}<br>
    isDefined? {{ isdefined }}<br>
    isUndefined? {{ isundefined }}<br>
  </div>
</body>
<script type="text/javascript" >
  var app = angular.module("myApps", []);
  app.controller("basicValid", function($scope){
    function a(){
      alert("a");
    }
    var temp = {name:"Bluejack"};
    var unknown;

    $scope.isnumber = angular.isNumber(2); //true
    $scope.isstring = angular.isString("2"); //true
    $scope.isobject = angular.isObject(temp); //true
    $scope.isArray = angular.isArray(temp); //false
    //(note : in javascript, array also an object but object not an array)

    $scope.isfunction = angular.isFunction(a); // true
    $scope.isdefined = angular.isDefined(unknown); //false
    $scope.isundefined = angular.isUndefined(unknown); //true
  });
</script>
```

Jika ingin menggunakan validasi untuk form, maka kita akan mengenal beberapa state :

- **ng-untouched** – State dimana form component belum “disentuh” sama sekali atau sudah.
- **ng-touched** – State dimana form component telah “disentuh” atau belum.
- **ng-pristine** – State dimana form component telah di modif isinya atau belum. Kalau statusnya **pristine** true, maka sudah pasti **touched** true. Tetapi belum tentu sebaliknya.
- **ng-dirty** – State dimana form component telah di modif atau belum.
- **ng-valid** – State dimana form component telah valid atau tidak.
- **ng-invalid** – State dimana form component tidak valid atau iya.

Untuk state valid dan invalid, penentuannya sesuai dengan **attribute/type** pada component (**HTML5 version**). Ada beberapa **attribute** yang menjadi patokan apakah form tersebut **valid** atau tidak, diantaranya :

email | max | maxlength | min | minlength | number | pattern | required | url | date | datetimelocal | time | week | month

Untuk referensi, cek aturan penggunaan attribute/type diatas pada halaman ini : https://www.w3schools.com/html/html_form_input_types.asp

```
<form name="userForm">
  <table>
    <tr>
      <td>Name</td><td></td>
      <td><input type="text" name="nameInput" ng-model="user.name" required></td>
    </tr>
    <tr>
      <td>Gender</td><td></td>
      <td>
        <input type="radio" name="radioGender" ng-model="user.gender" value="Male" required> Male
        <input type="radio" name="radioGender" ng-model="user.gender" value="Female" required> Female
      </td>
    </tr>
    <tr>
      <td>Age</td><td></td>
      <td><input type="number" name="ageInput" ng-model="user.age" required></td>
    </tr>
    <tr>
      <td>Country</td><td></td>
      <td><select name="countrySelect" ng-model="user.country" ng-options="country for country in countries" required></select></td>
    </tr>
    <tr>
      <td>Address</td>
      <td></td>
      <td><textarea name="addressInput" ng-model="user.address" required></textarea></td>
    </tr>
    <tr>
      <td>Blog</td>
      <td></td>
      <td><input type="url" name="blogInput" ng-model="user.blog"*/</td>
    </tr>
    <tr>
      <td colspan="3">
        <button ng-disabled="userForm.$invalid">Save</button>
        <button ng-click="reset()">Reset</button>
      </td>
    </tr>
  </table>
</form>
<p>
  {{ user | json }}
</p>
```



```
<span ng-show="userForm.nameInput.$invalid && userForm.nameInput.$dirty">Name can't be empty<br></span>
<span ng-show="userForm.ageInput.$invalid && userForm.ageInput.$dirty">Age can't be empty<br></span>
<span ng-show="userForm.radioGender.$invalid && userForm.radioGender.$dirty">Gender must be selected<br></span>
<span ng-show="userForm.countrySelect.$invalid && userForm.countrySelect.$dirty">Country must be selected<br></span>
<span ng-show="userForm.addressInput.$invalid && userForm.addressInput.$dirty">Address must be filled<br></span>
<span ng-show="userForm.blogInput.$invalid && userForm.blogInput.$dirty">Not a url format</span>
```

Error messages akan muncul ketika kondisi di dalam **ng-show** bernilai **true**.

Untuk mengambil state dari sebuah form **component**, bisa mengakses **attribute** name-nya. Bukan mengakses **modelnya**. Misalnya untuk **nameInput**, bisa diakses dengan mengakses **name** form terlebih dahulu [dot] **nameInput**. Lalu, kondisi **\$dirty** sengaja **ditambahkan** agar **errormessages** tidak muncul langsung ketika form di load tetapi akan muncul jika **sudah** di **modif** sebelumnya.

Seperti yang saya jelaskan diatas, **state valid dan invalid** ditentukan oleh **attribute/input type** **HTML5**. Contohnya untuk **blogInput**, kondisi **error** akan muncul karena type inputan untuk **blogInput** adalah **url** (new HTML5 features).

Lalu, bagaimana jika ingin validasi umur **harus diatas 17** misalnya. Sebenarnya , kita juga bisa **mengakses value** dari **modelnya** lalu menambahkan **compare operator** layaknya if seperti biasa.

```
<span ng-show="user.age < 17 && userForm.ageInput.$dirty">Age must be older than 17</span>
```

Note : Directive **ng-show/ng-hide/ng-if/ng-disabled** bisa diisi dengan kondisi if seperti biasa. Jadi boleh ditambahkan **compare operator**, ataupun **negasi (!)**.

```
<script type="text/javascript" >
  var app = angular.module("myApps", []);
  app.controller("registerCtrl", function($scope){
    $scope.countries = ["Indonesia", "Singapore", "Malaysia", "Japan"];
    $scope.defaultValue = {
      name : "John Doe",
      gender : "Male",
      age : 20,
      country : "Indonesia",
      address : "Jln. Raya",
      blog : "http://www.mahirkoding.com"
    }
    $scope.reset = function(){
      $scope.user = angular.copy($scope.defaultValue);
    }
  });
</script>
```

angular.copy() merupakan salah satu build in **function** dari AngularJS untuk **menduplikasi** isi object ke object lain. Perlu diketahui, kalau seandainya kita langsung **mengassignnya** menggunakan **tanda =** maka kedua object tersebut akan **sama value dan referencenya**. Ketika object a diubah, maka object b akan **ikut berubah**. So, gunakan **angular.copy()** untuk mengcopy isinya saja tanpa menyamakan referencenya.

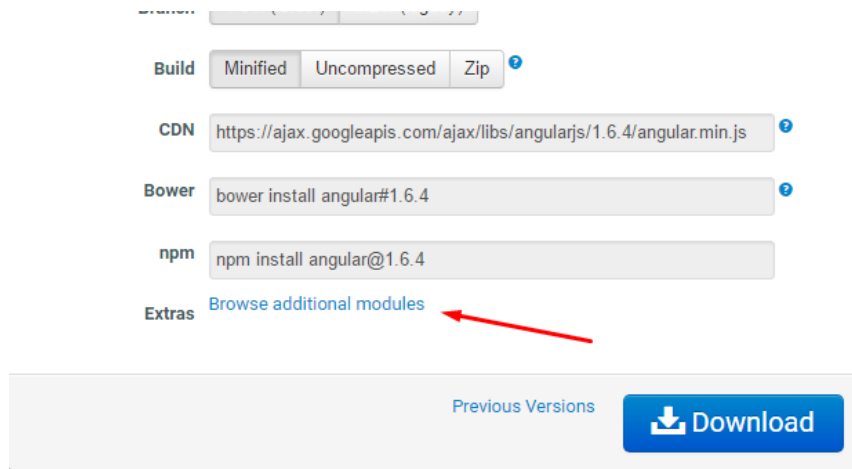
See result here :

https://bit.ly/hery_angular_7

Routings & Views

Layaknya framework-framework PHP pada umumnya, AngularJS juga memiliki fitur **routings** untuk mengontrol **view** yang akan **load** di browser berdasarkan **url**. Untuk dapat menggunakan routing di AngularJS, kita membutuhkan module **ngRoute** yang harus **diload** dalam angular module kita.

Installation



angular-resource.min.js.map	31-Mar-2017 09:31	11362
angular-route.js	31-Mar-2017 09:31	45008
angular-route.min.js	31-Mar-2017 09:31	5611
angular-route.min.js.map	31-Mar-2017 09:31	14451
angular-sanitize.js	31-Mar-2017 09:31	27725
angular-sanitize.min.js	31-Mar-2017 09:31	6101
angular-sanitize.min.js.map	31-Mar-2017 09:31	10962

Lalu, tambahkan **'ngRoute'** dalam scope pendefinisian angular modules.

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

Konsep angular route ini sama seperti **ajax load** yang akan mengambil **isi** dari **templateUrl** lalu di **render** ke dalam **directive ng-view**.

```

<body ng-app="myApps">
  <div>
    <a href="#/">Home</a>
    <a href="#view">View Data</a>
    <a href="#add">Add Data</a>
  </div>
</body>
<script type="text/javascript" >
  var app = angular.module("myApps", ['ngRoute']);

  app.config(function($routeProvider, $locationProvider){
    $locationProvider.hashPrefix('');
    $routeProvider
      .when("/", {
        template : "<h1>Welcome</h1>"
      })
      .when("/view",{
        templateUrl : "users/view.html"
      })
      .when("/add",{
        templateUrl : "users/add.html"
      })
      .when("/edit/:userID",{
        templateUrl : "users/edit.html",
        controller : "userController"
      })
      .otherwise({
        templateUrl : "404.html"
      })
  })

  app.controller("userController", function($scope, $routeParams){
    $scope.id = $routeParams.userID;
  })
</script>

```

Jika kamu menggunakan **AngularJS versi > 1.6**, pastikan menambahkan baris ini :

```
$locationProvider.hashPrefix('');
```

Karena, **secara default** angular versi > 1.6 **mempunyai default route** >> (!). Untuk itu, maka kita harus menggantinya menjadi **null** (' ')

templateUrl digunakan untuk **mengassign** halaman yang ingin diload sedangkan jika ingin langsung menuliskan htmlnya, bisa menggunakan property **template**.

Method **otherwise** digunakan untuk **mengkategorikan** route yang tidak terdefiniskan. (like else in selection). Di contoh diatas, method otherwise akan menampilkan tag html. Namun, penggunaan method otherwise lebih sering diarahkan untuk melakukan **redirect ke homepage**. Berikut caranya :

```

. otherwise({
  redirectTo : "/"
})

```

Jika ingin langsung **menbind** sebuah **controller** terhadap view yang akan kita load, maka tinggal tambahkan property **controller** di bawah template/templateUrl.

Lalu, seandainya ingin menambahkan parameter ke dalam routes yang kita buat, deklarasikan parameter tersebut dengan **tanda titik dua** dan kita tinggal mengaksesnya lewat **\$routeParams**.

```
.when("/edit/:userID",{
  templateUrl : "users/edit.html",
  controller : "userController"
})
```

userController :

```
app.controller("userController", function($scope, $routeParams){
  $scope.id = $routeParams.userID;
})
```

Edit.html :

```
<h1>Edit User yang akan diedit : {{ id }}</h1>
```

See result here :

https://bit.ly/hery_angular_8

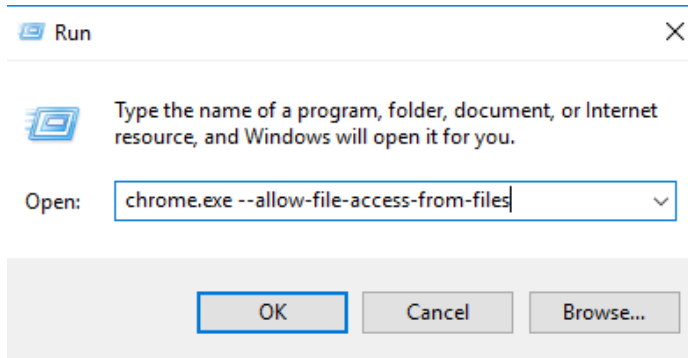
Lalu, kita juga bisa melakukan include file html lain seperti yang bisa dilakukan di php.

```
<div ng-include="'menu.html'"></div>
```

Pastikan path menu.html diberi **tanda kutip** agar filenya dapat di load oleh AngularJS. Mengapa? Karena directive **ng-include** menerima value berupa **expressions**. Oleh karena itu, menu.html harus kita ubah menjadi **string** terlebih dahulu.

```
1 <a href="#/">Home</a>
2 <a href="#view">View Data</a>
3 <a href="#add">Add Data</a>
```

NB : Jika terjadi error "Cross origin requests are only supported for protocol schemes: http, data, chrome, chrome-extension, https", pastikan dahulu web apps kita sudah berjalan di web server (localhost). Atau solusi lain adalah dengan cara force allow ketika menjalankan browser.




Cookies

AngularJS juga mempunyai module cookies yang dapat membantu kita dalam management cookies di browser.

Installation

```
angular-aria.min.js
angular-aria.min.js.map
angular-cookies.js
angular-cookies.min.js
angular-cookies.min.js.map
angular-csp.css
```



```
<script type="text/javascript" src="js/angular-cookies.min.js"></script>
```

Jika ingin menggunakan 2 module sekaligus, tinggal ditambahkan saja value barunya di dalam scope arraynya seperti gambar di bawah ini.

```
var app = angular.module("myApps", ['ngRoute', 'ngCookies']);
```

```
<body ng-app="myApps" ng-controller="userController">
  <input type="text" ng-model="key" placeholder="Key">
  <input type="text" ng-model="value" placeholder="Value">
  <button ng-click="store()">Store</button>
  <p>
    <ul>
      <li ng-repeat="(key,value) in allCookies">
        {{ key + " : " + value }}
        <a href="#" ng-click="alertData(key)">Alert</a>
        <a href="#" ng-click="deleteData(key)">Delete</a>
      </li>
    </ul>
  </p>
</body>
```

```

<script type="text/javascript" >
  var app = angular.module("myApps", ['ngCookies']);

  app.controller("userController", function($scope, $cookies){
    $scope.allCookies = $cookies.getAll();

    $scope.store = function(){
      $cookies.put($scope.key, $scope.value);
      //clear form
      $scope.key = "";
      $scope.value = "";
      $scope.allCookies = $cookies.getAll();
    }
    $scope.alertData = function(key){
      alert($cookies.get(key));
    }
    $scope.deleteData = function(key){
      $cookies.remove(key);
      $scope.allCookies = $cookies.getAll();
    }
  })
</script>

```

Ada beberapa method yang bisa digunakan untuk mengolah cookies yang ada di browser.

- **getAll()** – digunakan untuk mengambil semua object cookies yang ada (key dan value).
- **get(key)** – digunakan untuk mengambil value dari cookies berdasarkan keynya.
- **put(key, value)** – digunakan untuk inialisasi atau mengupdate sebuah cookies. Kalau key yang dituju belum ada, maka otomatis akan membuat cookies yang baru. Namun juga terdapat kesamaan nama, maka put() akan bertugas mengupdate isi dari cookies berdasarkan key yang dituju.
- **remove(key)** – digunakan untuk menghapus cookies berdasarkan keynya.

See result here :

https://bit.ly/hery_angular_9

Value, Factory dan Services

Tiga buah materi yang akan dibahas dalam bagian ini akan bermanfaat untuk membuat kodingan kita **menjadi lebih rapi dan terstruktur**. Jika sebelumnya kita sudah **memecah** setiap flow program ke dalam masing-masing **controller**, bagaimana kalau “**seandainya**” ada sebuah **fungsi** atau **variabel** yang dipakai **berkali-kali** di beberapa **controller**? Apakah kita harus

menulisnya kembali? Nah, **Value**, **Factory** dan **Service** merupakan **solusi** atas permasalahan kita tadi.

Ketiganya bertugas seakan-akan sebagai **"Helper Class"** dimana di dalam **class** tersebut didefinisikan **method/variabel "pembantu"** yang akan **diinject** lalu digunakan secara bersama-sama oleh beberapa controller.

Value

Value di AngularJS biasanya digunakan untuk mendefinisikan variabel **global**. Nantinya, value ini dapat digunakan di beberapa **controller/factory/service** jika telah **diinject**.

Pendefinisian value dilakukan terhadap module app kita.

```
var app = angular.module("myApps", []);

app.value("PI", 3.14);
app.value("title", "Aplikasi Penghitung Luas");

app.controller("shapeController", function(PI, title){
    alert(PI);
    alert(title);
})
```

Method value() menerima 2 parameter, yang pertama adalah **key**/namanya, dan yang kedua adalah **valuenya**. Jangan lupa, ketika ingin menggunakan value yang telah didefinisikan maka kita wajib **menginjectnya** ke dalam controller melalui **parameter**.

Factory

Jika value hanya bisa menyimpan **variabel**, maka di factory kita bisa membuat object yang berisi variabel dan method. Perlu diingat, ketika kita menggunakan factory, **yang akan di inject ke controller adalah return value dari factory tersebut**.

Untuk mendeklarasikan sebuah factory, kita dapat menggunakan method factory dari module yang telah dibuat.

```
<body ng-app="myApps" ng-controller="shapeController">
  <h1>{{ judulAplikasi }}</h1>
  <p>Sisi 1 : <input type="text" ng-model="sisi1"></p>
  <p>Sisi 2 : <input type="text" ng-model="sisi2"></p>
  <p>
    <button ng-click="segitiga()">Luas Segitiga</button>
    <button ng-click="persegiPanjang()">Luas Persegi Panjang</button>
  </p>
  <p>
    Result : {{ result }}
  </p>
</body>
```

```

<script type="text/javascript" >
  var app = angular.module("myApps", []);

  app.factory("luasFactory", function(){
    var obj = {};
    obj.factoryName = "Luas Factory";
    obj.luasSegitiga = function(a, b){
      return (a*b)/2;
    }
    obj.luasPersegiPanjang = function(a,b){
      return a*b;
    }
    return obj;
  })

  app.controller("shapeController", function($scope, luasFactory){
    $scope.judulAplikasi = luasFactory.factoryName;
    $scope.clearForm = function(){
      $scope.sisi1 = "";
      $scope.sisi2 = "";
    }

    $scope.persegiPanjang = function(){
      $scope.result = luasFactory.luasPersegiPanjang($scope.sisi1, $
        scope.sisi2);
      $scope.clearForm();
    }
    $scope.segitiga = function(){
      $scope.result = luasFactory.luasSegitiga($scope.sisi1, $scope.
        sisi2);
      $scope.clearForm();
    }
  })
</script>

```

Dalam contoh di atas, kita telah membuat sebuah object lalu kita membuat **1 buah variabel** dan **2 buah function** ke dalam object tersebut. Lalu object tersebut akan kita **return**. Sesuai dengan cara kerja **factory**, object yang telah kita buat tadi seolah-olah akan **diinject** ke dalam **shapeController**. Nantinya, kita tinggal memanggil **method/variabel** dari **factory** tersebut untuk digunakan.

See result here :

https://bit.ly/hery_angular_10

Services

Untuk services, bisa dikatakan **hampir sama persis dengan Factory**. Perbedaannya hanya terletak pada **bagian yang diinject**. Pada factory, **bagian yang diinject ke controller adalah return valuenya**. Sedangkan kalau services, yang diinject pada controllernya adalah **servicesnya**. **Service tidak memerlukan return type**, tetapi pada saat diinject ke controller, **service seakan-akan sudah langsung menjadi sebuah object**.

Dengan contoh program yang sama, berikut adalah perbedaannya : (bagian view tidak ada yang diganti)

```
<script type="text/javascript" >
  var app = angular.module("myApps", []);

  app.service("luasService", function(){
    this.factoryName = "Luas Factory";
    this.luasSegitiga = function(a, b){
      return (a*b)/2;
    }
    this.luasPersegiPanjang = function(a,b){
      return a*b;
    }
  })

  app.controller("shapeController", function($scope, luasService){
    $scope.judulAplikasi = luasService.factoryName;
    $scope.clearForm = function(){
      $scope.sisi1 = "";
      $scope.sisi2 = "";
    }

    $scope.persegiPanjang = function(){
      $scope.result = luasService.luasPersegiPanjang($scope.sisi1, $scope.sisi2);
      $scope.clearForm();
    }
    $scope.segitiga = function(){
      $scope.result = luasService.luasSegitiga($scope.sisi1, $scope.sisi2);
      $scope.clearForm();
    }
  })
</script>
```

Pendeklarasian service menggunakan method service() dari module yang telah dibuat.

Kita langsung bisa menganggap service tersebut adalah sebuah class yang nantinya akan digunakan tanpa dibuat objectnya (static).

Namun, kalau kita perhatikan, AngularJS seolah-olah otomatis menginstansiasi object ketika services tersebut **diinject** dalam controller.

```
var luasService = new luasService();|
```

Services tidak selamanya wajib dibuat sendiri. Ada beberapa service yang sudah tersediakan dari AngularJS. Beberapa diantaranya adalah **service location**, **timeout** dan **http**.

Location

Service ini digunakan untuk mengetahui detail dari location/URL yang sedang diakses.

```
<body ng-app="myApp" ng-controller="testController">
  Current URL : {{ currentUrl }}<br>
  Current Protocol : {{ currentProtocol }}<br>
  Current Host : {{ currentHost }}<br>
  Current Port : {{ currentPort }}<br>
</body>
<script type="text/javascript">
  var app = angular.module("myApp", []);
  app.controller("testController", function($scope, $location){
    $scope.currentUrl = $location.absUrl();
    $scope.currentProtocol = $location.protocol();
    $scope.currentHost = $location.host();
    $scope.currentPort = $location.port();
  })
</script>
```

Beberapa method yang tersedia dalam service location :

- **absUrl()** – Untuk mengambil url yang aktif terbuka sekarang dari address bar.
- **protocol()** – Untuk mengetahui protokol apa yang sedang digunakan saat ini.
- **host()** – Untuk mengetahui host yang sedang digunakan saat ini.
- **port()** – Untuk mengetahui port yang sedang digunakan saat ini.

Timeout

Digunakan untuk membuat timeout/delay seperti **setTimeout** di javascript.

```
<script type="text/javascript">
  var app = angular.module("myApp", []);
  app.controller("testController", function($scope, $timeout){
    $scope.sayHello = function(){
      alert("Hello World!");
    }
    $timeout($scope.sayHello, 3000);
  })
</script>
```

HTTP

Service http berfungsi untuk melakukan ajax request, untuk penggunaannya mirip seperti JQuery AJAX, disini saya mencoba menconsume dummy web API yang mereturn json dari :

POSTS (article)

- GET : <https://jsonplaceholder.typicode.com/posts/>
- GET : [https://jsonplaceholder.typicode.com/posts/\[id\]](https://jsonplaceholder.typicode.com/posts/[id])

COMMENTS

- GET : [https://jsonplaceholder.typicode.com/posts/\[id\]/comments](https://jsonplaceholder.typicode.com/posts/[id]/comments)

Untuk penggunaan method lain juga bisa, support beberapa HTTP method yang umum digunakan dalam REST API seperti PUT/PATCH/DELETE dll.

Basic syntax :

```
$http.get('/someUrl', config).then(successCallback, errorCallback);  
$http.post('/someUrl', data, config).then(successCallback, errorCallback);
```

Contoh di bawah ini akan saya gabungkan dengan konsep service yang telah kita pelajari sebelumnya.

View index.html :

```
<style type="text/css">  
  .posts{  
    padding: 10px;  
    border: 1px solid #000;  
    margin-bottom: 10px;  
    width: 400px;  
  }  
</style>  
<body ng-app="myApp">  
  <div ng-include="'menu.html'"></div>  
  <ng-view></ng-view>  
</body>
```

Di dalam index, saya membuat 1 div untuk **menu.html** yang **diinclude** dengan directive **ng-include**.

Lalu, siapkan juga sebuah directive **ng-view** sebagai “wadah” bagi **view** yang akan dirender ketika **routing** bekerja.

Config routes :

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

app.config(function($routeProvider, $locationProvider){
  $locationProvider.hashPrefix('');
  $routeProvider
    .when("/", {
      template : "<h1>Welcome to Simple App"
    })
    .when("/view",{
      templateUrl : "posts/view.html",
      controller : "postCtrl"
    })
    .when("/view/:postID", {
      templateUrl : "posts/detail.html",
      controller : "postDetailCtrl"
    })
  })
});
```

Untuk routingnya saya menyediakan 3 link yaitu link home, view all posts dan view detail post.

Services dan controller :

```
app.service("PostService", function($http){
  this.loadAll = function(){
    return $http.get("https://jsonplaceholder.typicode.com/posts");
  }
  this.loadDetail = function(postID){
    return $http.get("https://jsonplaceholder.typicode.com/posts/"+postID);
  }
  this.loadComment = function(postID){
    return $http.get("https://jsonplaceholder.typicode.com/posts/"+postID+"/comments");
  }
})

app.controller("postCtrl", function($scope, PostService){
  PostService.loadAll().then(function(response){
    $scope.posts = response.data;
  })
})

app.controller("postDetailCtrl", function($scope, PostService, $routeParams){
  PostService.loadDetail($routeParams.postID)
    .then(function(response){
      $scope.post = response.data;
    });
  PostService.loadComment($routeParams.postID)
    .then(function(response){
      $scope.comments = response.data;
    })
  })
});
```

Nah, bagian ini adalah bagian yang paling penting. Supaya rapi, saya membuat sebuah **services** yang akan mengurus **flow** program untuk **posts section**. **Service** ini akan **diinject** ke **postCtrl** (view all posts) dan ke **postDetailCtrl** (view post detail).

Masing-masing controller akan berjalan **otomatis** ketika url telah di load oleh **ng-routers**.

View menu.html :

```
<a href="#/">Home</a>
<a href="#view">View Data</a>
```

View view.html :

```
<div class="posts" ng-repeat="post in posts">
  Title : {{ post.title }}<br>
  Content : {{ post.body }}<br>
  <a href="#view/{{ post.id }}">Detail</a>
</div>
```

View detail.html :

```
<div class="posts">
  Title : {{ post.title }}<br>
  Content : {{ post.body }}
</div>
<h1>Comments</h1>
<table width="50%" border="1" cellspacing="0" cellpadding="5">
  <tr ng-repeat="comment in comments">
    <td>
      Name : {{ comment.name }} | Email : {{ comment.email }}
      <br>
      {{ comment.body }}
      <br>
    </td>
  </tr>
</table>
```

Untuk method-method lain penggunaannya lebih kurang sama seperti jquery ajax, jika menggunakan method POST, jangan lupa untuk mengirim datanya juga. Sintaksnya juga kurang lebih sama.

Dokumentasi lengkap tentang HTTP Service : [https://docs.angularjs.org/api/ng/service/\\$http](https://docs.angularjs.org/api/ng/service/$http)

See result here :

https://bit.ly/hery_angular_11