AngularJS

Beginner's guide - part 2

POLITECNICO
DI TORINO

e-Lite

Summary of the previous lesson

1. To add AngularJS to an empty page:
   a) Download the script `angular.js` from https://angularjs.org/
   b) Link it in the header

```
<script src=
"https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angul
ar.min.js" ></script>
```

   c) Add the `ng-app` directive to the portion of the page that AngularJS should process

```
<html lang="en" ng-app>
...
```

Summary of the previous lesson

2. To create a controller that implements the logic of your app:
   a) Declare the module in your HTML page

```
<html lang="en" ng-app="sonetExample" >
```

   b) Create a JavaScript file (example.js) containing the module and the controller declaration

```
angular.module("sonetExample", [])
 .controller('MyCtrl', function ($scope) {
    //do something
})
```

Summary of the previous lesson

c) Link it in the HTML header

```
<script src= "./example.js" ></script>
```

d) Use the ng-controller directive to say to AngularJS in which part of the page it should be used

```
<body ng-controller="MyCtrl">
```

e) Modify the script (example.js) to actually implement the controller actions

```
angular.module("sonetExample", [] )
 .controller('MyCtrl', function ($scope) {
    $scope.name = "Teo";
})
```

Summary of the previous lesson

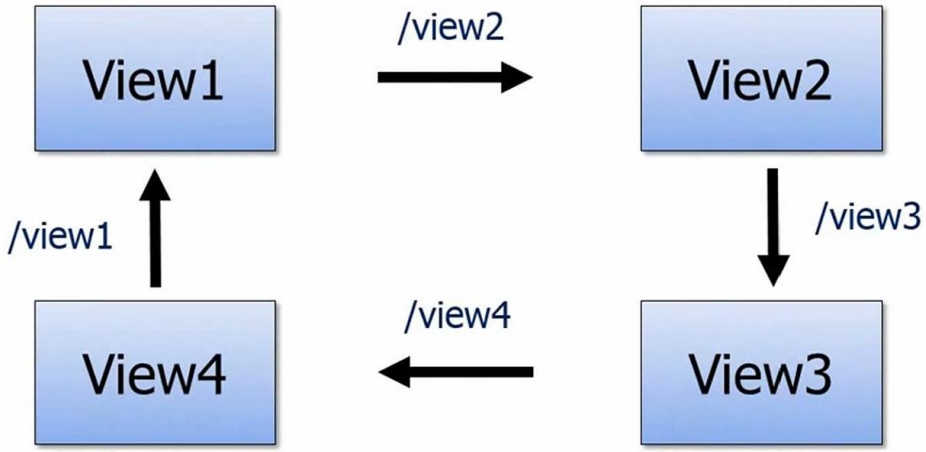
3. To share data between the controller and the view:
   - Use the $scope

```
angular.module("sonetExample", [])
 .controller('MyCtrl', function ($scope) {
    ...
})
```

→ Example 5 and 6

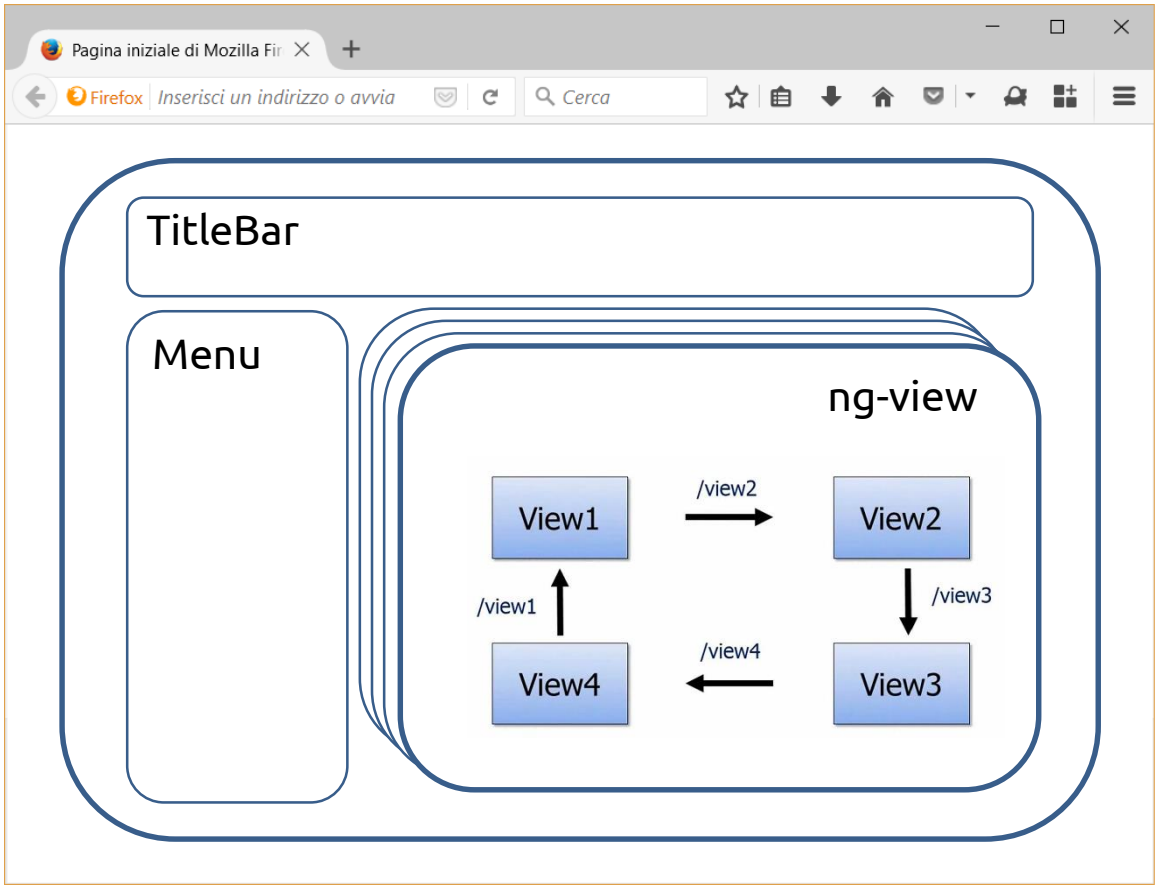
Routing

- Single page applications need routing
  - to mimic static addresses (http://mysite.com/...)
  - to manage browser history
- Goal: load different views into the single page app



Routing in AngularJS

- Goal: load different views into a portion of the single page app



How: ngRoute

- Angular API for routing
- Provides
  - a directive to indicate where view component should be inserted in the template (`ngView`)
  - a service that watches `window.location` for changes and updates the displayed view (`$route`)
  - a configuration that allows the user to specify the URL to be used for a specific view (`$routeProvider`)
- It is located in a dedicated js file
  - `angular-route.min.js`

How to use ngRoute

- In a template

```
<div ng-view></div>
```

- In your Angular module, add ngRoute as a dependency

```
angular.module('sonetExample', [ngRoute])
```

  - Now our application has access to the route module, which provides the $routeProvider

- Configure the routing table in a **config** block

```
.config(['$routeProvider',
   function($routeProvider) {
   …
}])
```

→ Example 7

How to use ngRoute

- In a template

```
<div ng-view>
```

- In your Angula
  dependency

Every time we use an external module we have to specify it in the module **dependencies**

```
angular.module('sonetExample', [ngRoute])
```

  - Now our application has access to the route module, which provides the $routeProvider

- Configure the routing table in a **config** block

```
.config(['$routeProvider',
   function($routeProvider) {
   …
}])
```

→ Example 7

How to use ngRoute

- In a template

```
<div ng-vie
```

- In your Angu
dependency

```
angular.mod
```

  - Now our ap dule, which provides the $routeProvider

Dependencies are important for other components, too! In this case we have to use methods provided by the routeProvider, so we declare its dependency

- Configure the routing table in a **config** block

```
.config(['$routeProvider',
   function($routeProvider) {
   …
}])
```

→ Example 7

How to use ngRoute: Example7

- example7.html:

```
<!DOCTYPE html>
<html lang="en" ng-app="sonetExample7">
<head>
    <meta charset="UTF-8">
    <title>Routing and Views</title>
    <script src="./angular.min.js"></script>
    <script src="./angular-route.min.js"></script>
    <script src="./example7.js"></script>
</head>

<body>
    <div ng-view></div>
</body>
</html>
```

How to use ngRoute: Example7

- enter-name.html:

```
<div>
    <label>Name</label>
    <input type="text" ng-model="name" placeholder="Enter
your name">
    <h1>{{ greeting }} {{name}}!</h1>
</div>
```

How to use ngRoute: Example7

- example7.js:

```
angular.module("sonetExample7", ['ngRoute'])

.config(['$routeProvider', function ($routeProvider) {
    $routeProvider
        .when('/name', {
            templateUrl: 'enter-name.html',
            controller: 'MyCtrl'
        })
        .otherwise({redirectTo: '/name'});
}])

.controller('MyCtrl', function ($scope) {
    $scope.name = "";
    $scope.greeting = "Ciao";
});
```

How to use ngRoute: Example7

- example7.js:

```
angular.module("sonetExample7", ['ngRoute'])

.config(['$routeProvider', function ($routeProvider) {
    $routeProvider
        .when('/name', {
            templateUrl: 'enter-name.html',
            controller: 'MyCtrl'
        })
        .otherwise({redirectTo: '/name'});
}])

.controller('MyCtrl', fu
    $scope.name = "";
    $scope.greeting = "C
});
```

Here the **controller** is specified in an internal method

Passing parameters in URLs

URLs:

- #/pizzas/my-pizza
- #/pizzas/another-pizza

Routing table configuration:

JS

```
.config(['$routeProvider', function ($routeProvider){
  $routeProvider
    .when('/pizzas/:pizzaId', {
      templateUrl: 'single-pizza.html',
      controller: 'PizzaCtrl',
    })
    [...]
}])
```

Passing parameters in URLs

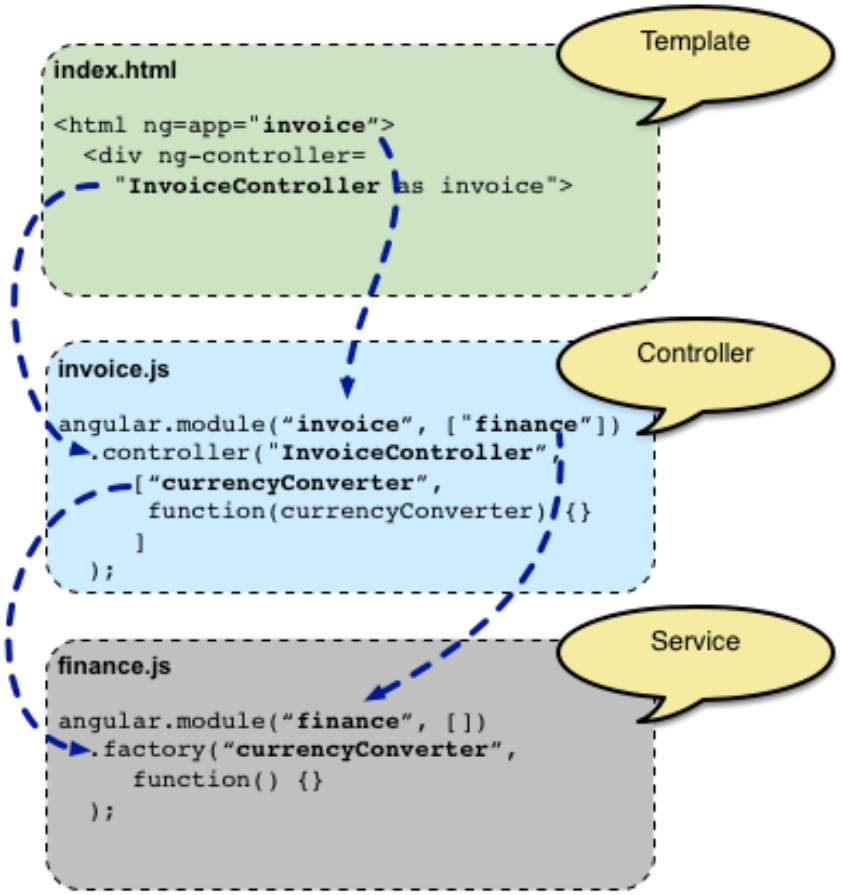
Controller:

JS

```
.controller('PizzaCtrl', ['$routeParams',
  function ($routeParams) {
    $routeParams.pizzaId
    // will be my-pizza or another-pizza
})
```

Services

- Responsible to hold the model and communicate with a server
- Singleton objects that are instantiated only once per app and lazy loaded
  - controllers are instantiated only when they are needed and discarded otherwise



Built-in Services

- Server communication
  - $http, $resource, …
- Wrapping DOM access
  - $location, $window, $document, …
- JavaScript functionality
  - $animate, $log, …

The $http Service

- Uses the XMLHttpRequest object (or JSONP) to communicate with a server
- How it works
  - takes a single argument: a configuration object
  - returns a promise with two methods: `success()` and `error()`
- It can use the `.then()` method, too
  - it registers a dedicated callback for the success and error method

The $http Service

promise

JS

```
$http({
    method: 'GET',
    url: '/someUrl'})
   .success(function(data, status, headers, config) {
     // this callback will be called asynchronously
     // when the response is available
   })
   .error(function(data, status, headers, config) {
     // called asynchronously if an error occurs
     // or server returns response with an error status.
   });
});
```

callbacks

The $http Service

- Shortcut methods

JS

```
var sonet = angular.module('sonetExample', []);

sonet.controller('MyCtrl', ['$scope', '$http',
    function ($scope, $http) {
        var promise = $http.get('/pizzas.json');
        promise.success(function(data) {
            $scope.pizzas = data;
        });
    }
]);
```

Shortcut methods:
.get(), .post(), .put(), .delete(), etc.

$http in a Custom Service

JS

```
// custom service
sonet.factory('Pizza', function($http) {
    var pizzaService = {
        getData: function () {
            return $http.get('/pizzas.json')
              .then(function (response) {
                return response.data;
            });
        }
    };
    return pizzaService;
})

// controller
sonet.controller('PizzaCtrl', ['$scope', 'Pizza',
    function ($scope, Pizza) {
        Pizza.getData().then(function(pizzaList) {
            $scope.pizzas = pizzaList;
        });
    }
]);
```

→ Example 8

The $resource Service

- Replace the "low level" $http service
  - it has methods with high-level behaviors
  - but the server URLs must be RESTful
- REST: REpresentational State Transfer
  - works on the concept of “resource”
- CRUD: Create, Read, Update and Delete

read a collection of resources:	GET /pizzas
read a single resource:	GET /pizzas/:id
add a new resource:	POST /pizzas
update an existing resource:	PUT /pizzas/:id
delete an existing resource:	DELETE /pizzas/:id

The $resource Service

- Requires the ngResource module to be installed
- No callbacks needed!
  - e.g., `$scope.pizzas = pizzaService.query();`
- Available methods
  - get(), method: GET, single resource
  - query(), method: GET, is array (collection)
  - save(), method: POST
  - remove(), method: DELETE,
  - delete(), method: DELETE

Other best practices in AngularJS

BEST PRACTICES

A dot in the model

"There should always be a dot in your model"

Parent scope: `$scope.name = "Anon"`
Child scope: `<input type="text" ng-model="name">`

The application will display "*Anon*" initially, but once the user changes the value a *name* will be created on the child scope and the binding will read and write that value.
The parent's name will remain "*Anon*".
This can cause problems in large applications.
To avoid them:

```
<input type="text" ng-model="person.name">
```

camelCase vs dash-case

You can use variable named like

```
my-variable
myVariable
```

- The latter can cause problems in HTML
  - it is case-insensitive
- Angular solution: use either, it will map them to the same thing

Use dash-case in HTML and camelCase in JavaScript

ADDITIONAL INFORMATION

Custom Directives

- Application specific
  - replace HTML tags with “functional” tags you define e.g., DatePicker, Accordion, ...
- Naming
- Camel case naming (`ngRepeat`)
  - automatically mapped to `ng-repeat`, `data-ng-repeat` in the templates
    e.g., `<div ng-repeat></div>`

Custom Directives

- A directive returns an “object” (directive definition) with several properties
  - details: https://docs.angularjs.org/api/ng/service/$compile

JS

```
var sonet = angular.module('sonetExample', []);

sonet.directive('helloHeader', function() {
    return {
        restrict: 'E',
        template: '<h2>Hello World!</h2>'
    };
});
```

Custom Directives

JS

```
var sonet = angular.module('sonetExample', []);

sonet.directive('helloHeader', function() {
    return {
        restrict: 'E',
        template: '<h2>Hello World!</h2>'
    };
});
```

Restricts the directive to a specific HTML "item".
If omitted, the defaults (Elements and Attributes) are used.

Custom Directives

JS

```
var sonet = angular.module('sonetExample', []);

sonet.directive('helloHeader', function() {
    return {
        restrict: 'E',
        template: '<h2>Hello World!</h2>'
    };
});
```

replace or wrap the contents of an element with this template

Compile and Link

- Named after the two phases Angular uses to create the live view for your application
- Compile
  - looks for all the directives and transforms the DOM accordingly, then calls the compile function (if it exists)
- Link
  - makes the view dynamic via directive link functions
  - the link functions typically create listeners on the DOM or the model; these listeners keep the view and the model in sync at all times
  - scopes are attached to the compiled link functions, and the directive becomes live through data binding

Link Function: Example

JS

```
var sonet = angular.module('sonetExample', []);

sonet.directive('backButton', ['$window', function ($window) {
    return {
        restrict: 'A',
        link: function (scope, elem, attrs) {
            elem.bind('click', function () {
                $window.history.back();
            });
        }
    };
}])
```

References

- AngularJS official guide
  - https://docs.angularjs.org/guide
- AngularJS API documentation
  - https://docs.angularjs.org/api
- AngularJS in 60 minutes [video]
  - https://www.youtube.com/watch?v=i9MHigUZKEM
- Learn Angular in your browser
  - http://angular.codeschool.com/

01QYAPD Social Networking: technologies and applications

My contact information:
Teodoro Montanaro (teodoro.montanaro@polito.it)

Thanks to **Luigi De Russis** (luigi.derussis@polito.it), the creator the slides I used as basis!

License

- This work is licensed under the Creative Commons “Attribution-NonCommercial-ShareAlike Unported (CC BY-NC-SA 3,0)” License.
- You are free:
  - to **Share** - to copy, distribute and transmit the work
  - to **Remix** - to adapt the work
- Under the following conditions:
  - **Attribution** - You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
  - **Noncommercial** - You may not use this work for commercial purposes.
  - **Share Alike** - If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.
- To view a copy of this license, visit http://creativecommons.org/license/by-nc-sa/3.0/