MỤC LỤC

CHƯƠNG I: TỔNG QUAN VỀ ANGULAR JS FRAMEWORK

1.1. Ngôn ngữ lập trình web JavaScript

1.1.1. Lịch sử phát triển

- Được phát triển bởi Brendan Eich tại Hãng truyền thông Netscape với cái tên đầu tiên Mocha, rồi sau đó đổi tên thành LiveScript, và cuối cùng thành JavaScript.

- JavaScript được bổ sung vào trình duyệt Netscape bắt đầu từ phiên bản 2.0b3 của trình duyệt này vào tháng 12 năm 1995. JavaScript được bổ sung vào trình duyệt Internet Explorer bắt đầu từ Internet Explorer phiên bản 3.0 được phát hành tháng 8 năm 1996.

1.1.2. Javascript là gì?

- JavaScript là một ngôn ngữ lập trình kịch bản dựa trên đối tượng được phát triển từ các ý niệm nguyên mẫu. Ngôn ngữ này được dùng rộng rãi cho các trang web, nhưng cũng được dùng để tạo khả năng viết script sử dụng các đối tượng nằm sẵn trong các ứng dụng.

1.1.3. Javascript có thể làm được những gì?

- Trên trình duyệt, rất nhiều trang web sử dụng JavaScript để thiết kế trang web động và một số hiệu ứng hình ảnh thông qua DOM. JavaScript được dùng để thực hiện một số tác vụ không thể thực hiện được với chỉ HTML như kiểm tra thông tin nhập vào, tự động thay đổi hình ảnh ...

- Bên ngoài trình duyệt, JavaScript có thể được sử dụng trong tập tin PDF của Adobe Acrobat và Adobe Reader. Điều khiển Dashboard trên hệ điều hành Mac OS X phiên bản 10.4 cũng có sử dụng JavaScript.

1.2. Tổng quan về AngularJs framework

1.2.1 Lịch sử phát triển

- Dự án AngularJS được bắt đầu từ năm 2009, do lập trình viên Misko Hevery tại Google viết ra. Misko và nhóm lúc này đang tham gia vào 1 dự án của Google tên là Google Feedback. Với AngularJS, Misko đã rút ngắn số dòng code front-end từ 17000 dòng còn chỉ khoảng 1500. Với sự thành công đó, đội ngũ của dự án Google Feedback quyết định

phát triển AngularJS theo hướng mã nguồn mở. Theo thông số từ Github mà mình thấy, hiện tại dự án AngularJS đang có gần 11000 người theo dõi và hơn 2000 lượt fork.

- Công nghệ HTML hỗ trợ tốt cho các trang web tĩnh, kiểu như trước năm 2000 vậy. Khi bạn xây dựng 1 trang web với PHP, Node/Express, hay Ruby thì nó cũng chỉ là một trang web tĩnh với nội dung được thay đổi khi bạn gửi request về máy chủ, máy chủ sẽ render 1 trang với nội dung tương ứng. Tuy nhiên mọi thứ đã thay đổi nhiều từ sự phát triển của HTML5, nhất là khi có sự chống lưng từ những ông lớn như Google, Yahoo, Facebook, và sự tập hợp đông đảo của cộng đồng mã nguồn mở.

- Douglas Crockford, người được biết đến nhiều qua JSON và JSLint đã dùng sự chênh lệch về độ dày giữa 2 cuốn sách "Javascript: The definitive guide" và "Javascript: The good parts" để châm biếm 1 cách hài hước về JavaScript.

- Tuy nhiên, sự thành công của jQuery đã khiến JavaScript được nhiều người yêu thích vì tính đơn giản và dễ sử dụng. Việc phát triển 1 website sử dụng AJAX thì không khó, bạn có thể dùng jQuery để làm việc này với $.ajax tuy nhiên làm thế nào để xây dựng một phần mềm có thể mở rộng, dễ test, nâng cấp và bảo trì thì không hề đơn giản vì bản thân JavaScript không được thiết kế ngay từ đầu để làm những việc này. Do đó sự ra đời của những framework hỗ trợ lập trình viên xây dựng ứng dụng web 1 cách có hệ thống đã ra đời như Sencha, Ember, Knockout, Backbone, và AngularJS.

1.2.2. Angular JS là gì?

- Angular là framework javascript mạnh mẽ. Angular tăng cường HTML cho các ứng dụng web. Nó có chức năng để giảm bớt quá trình phát triển ứng dụng web. Từ nhiều năm trước, khi HTML mới bắt đầu, nó được dự định để xây dựng trang web hoặc có thể nói đó là một cách để hiển thị tài liệu tĩnh, không để xây dựng một ứng dụng web động. Bây giờ, hãy tưởng tượng nếu HTML lần đầu tiên được dùng để xây dựng các ứng dụng web động, đó là Angular. Angular là những gì HTML sẽ có được nếu nó được thiết kế để xây dựng các ứng dụng web . Lời giải thích này có thể khó hiểu nhưng khi đọc tiếp về sau bạn sẽ ngạc nhiên với những gì Angular có thể làm và sẽ hiểu ý nghĩa này.

1.2.3. Các tính năng chính

- Hai cách liên kết dữ liệu trong Angular là khả năng thay đổi giá trị thuộc tính của đối tượng, đồng thời giao diện người dùng đưa ra sự thay đổi đó ngay tại thời điểm đó, và ngược lại. Ví dụ, nếu bạn có thuộc tính của một đối tượng gọi là "Page.Title ', mỗi lần thay đổi giá trị' Page.Title, giao diện người dùng tự động cập nhật và hiển thị giá trị mới của 'Page.Title. Và nếu bạn nhập vào trong giao diện ứng dụng người dùng của bạn, nó có thể cập nhật giao diện người dùng và cũng cập nhật thuộc tính của đối tượng. Tất cả đều được xử lý bởi Angular do đó không cần phải viết giống như trong một số framework javascript khác. Những gì cần làm chỉ đơn giản là xác định model của chúng ta, và bây giờ bất cứ khi nào một model được thay đổi, nó sẽ thay đổi bất cứ nơi nào nó được đặt, có thể là trong đối tượng cụ thể hoặc trong các lớp đại diện. Model trong Angular chỉ là cấu trúc javascript cũ rõ ràng, không có gì lạ mắt nhưng nó lại rất hữu ích.

- Tính năng mới thực sự là đặc điểm tạo nên một khoảng cách khác nhau lớn giữa Angular và bất kỳ framework javascript khác. Directives là đề cập đến một tính năng để mở rộng nghĩa của từ HTML, đồng thời cũng có thể được xem như là một cách để chỉ cho trình duyệt mẹo mới của bạn. Chúng ta biết về header trong HTML5, đó là đại diện cho một tiêu đề nhưng nó chỉ là phần tử tĩnh. Mặc dù HTML có thể có nhiều thẻ , nhưng nó không đủ mạnh để tạo ra một ứng dụng web đẹp. Với đặc điểm chỉ dẫn , chúng ta có thể tạo ra từ vựng HTML mới để trình duyệt hiểu nó có nghĩa là gì và nó nên làm gì. Ví dụ với Angular chúng ta có thể tạo ra thẻ như sau <draggable> drag me </ draggable > . Các văn bản bên

trong tag draggable bây giờ sẽ trở thành draggable trong trình duyệt, sau đó chỉ cần viết một định nghĩa chỉ dẫn draggable trong ứng dụng Angular của chúng ta là xong.Đây là một ví dụ đơn giản của Angular, nó có thể làm rất nhiều thứ hữu ích hơn. Và nó không chỉ giới hạn một phần tử html mà có sẵn để thực thi các thuộc tính, các lớp hoặc các chú thích html. Directives trong Angular khác với directtives tạo bởi html ở chỗ chúng có thể thực hiện nhiều hành vi động hơn.

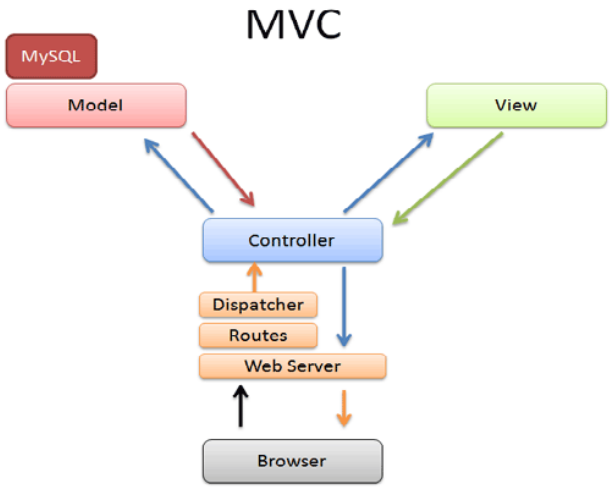
- Nhiều framework cần có template nhưng Angular template lại chỉ là html.

- Dependency injection làm cho ứng dụng Angular dễ dàng hơn để phát triển, chúng ta có thể sắp xếp mã (code) thành các module. Bằng cách này, chúng ta có thể tạo ra các thành phần tái sử dụng mà có thể được gọi về sau, bất cứ khi nào cần thiết. Ý tưởng này là đặc biệt tốt là khi ứng dụng mở rộng lớn hơn và cần phải được mở rộng, dễ dàng bảo trì và kiểm chứng. Dependency injection là một cách để ghép các đoạn mã code với nhau, điều này đòi hỏi phải phân chia mã code theo các chức năng riêng biệt như service, controller, hoặc provider nhưng không giới hạn chúng.

1.2.4. Đặc trưng

- Kiến trúc MVC, Two-way binding, Dynamic templates, Expressions, Modules, Scopes, Dependency injection, Directives, Routing, Services, Filters, Form validation, Testing in mind.

1.2.5. Mô hình MVC

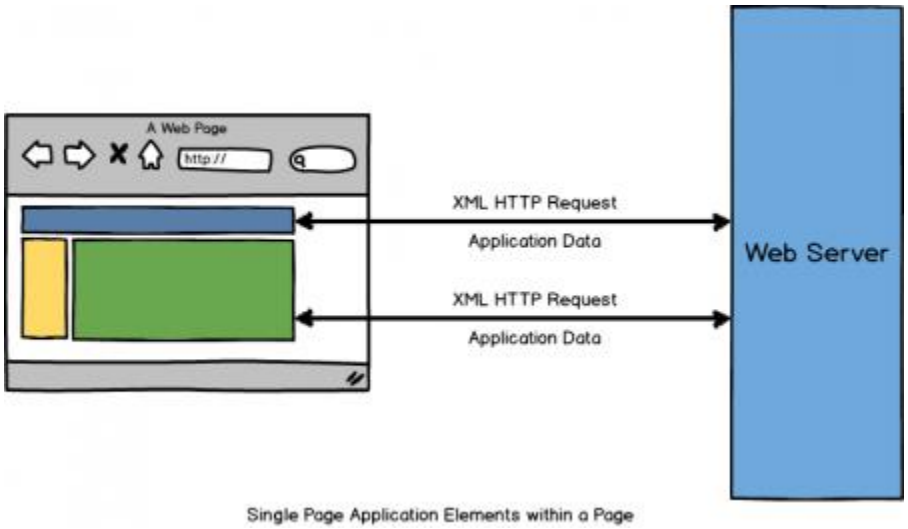


- **Mô hình MVC** (**M**odel - **V**iew - **C**ontroller) là một kiến trúc phần mềm hay mô hình thiết kế được sử dụng trong kỹ thuật phần mềm. Nó giúp cho các developer tách ứng dụng của họ ra 3 thành phần khác nhau Model, View và Controller. Mỗi thành phần có một nhiệm vụ riêng biệt và độc lập với các thành phần khác.
- Mô hình MVC được giới thiệu từ những năm 70 như một phần của Smalltalk, nhưng đối với nền tảng web, thì nó mới được thịnh hành gần đây.
- Ý tưởng đằng sau MVC là để chia rõ 3 thành phần chính là model(xử lý, truy xuất database), view(giao diện), và controller(điều hướng yêu cầu từ người dùng).
- MVC thể hiện tính chuyên nghiệp trong lập trình, phân tích thiết kế. Do được chia thành các thành phần độc lập nên giúp phát triển ứng dụng nhanh, đơn giản, dễ nâng cấp, bảo trì. Đối với Angular, View sẽ là DOM, Controller là các lớp JavaScript, còn Model sẽ là dữ liệu được lưu ở thuộc tính của các đối tượng trong JS.
- Sau khi chứng kiến nhiều tranh luận về MV*, một tác giả của Angular đã tuyên bố AngularJS là một MVW framework (Model – View – Whatever, trong đó Whatever là viết tắt của whatever works for you).

1.2.6. SPA – Single Page Application

- Một single page application hay còn được gọi là single page interface, là một web app hay website hiển thị vừa vặn trên một mặt của trang web với mục đích giúp người dùng có trải nghiệm giống như đang dùng ứng dụng trên desktop.

- Là ứng dụng chạy bên trong trình duyệt, không yêu cầu phải tải lại toàn bộ trang web mỗi lần sử dụng.



Single Page Application Elements within a Page

CHƯƠNG II: ANGULAR JS FRAMEWORK

2.1. Tại sao phải sử dụng AngularJs?

- Angular được đưa ra bởi Google. Tại sao điều này quan trọng để biết ? Như bạn có thể đoán, Google đã phát triển các tài năng và thiên tài như đội bóng của họ . Họ thực sự biết những tinh tế của trang web và những sự phát triển ứng dụng web . Ít nhất là thực tế này có thể cung cấp cho người dùng đảm bảo Angular xuất phát từ người chúng ta có thể tin tưởng . Ngoài ra, nếu bạn đã từng sử dụng sản phẩm của Google như Gmail hay Google Plus, bạn sẽ không ngạc nhiên với sự tương tác của chúng và cả ajax gửi liên tục mọi nơi mà không phải làm mới toàn bộ trang web và để sử dụng Angular kiến thức chủ yếu cần phải biết trước là Javascript.

2.2. AngularJs được lập trình như thế nào?

2.2.1. Cài đặt AngularJs

2.2.1.1. Tải AngularJS

- Truy cập vào trang web https://angularjs.org hoặc https://github.com/angular/angular.js và tải về bản angularjs mới nhất. Bản hiện mới nhất hiện tại là 1.2.16.

2.2.1.2. Chèn Angular vào ứng dụng.

*** Tự host AngularJS**

```
<script src="angular.min.js"></script>
```

- Theo cách này thì cần phải download angularjs về máy và nhúng trực tiếp vào ứng dụng.

*** Dùng phiên bản có sẵn trên server của Google**

- Ngoài cách trên ra bạn cũng có thể sử dụng phiên bản nén của AngularJS có sẵn trên server của Google. Sử dụng cách này có 2 điều lợi là tiết kiệm băng thông cho trang web của bạn và AngularJS sẽ được load nhanh hơn nếu máy của người dùng đã cache AngularJS.

- Nhưng cách này không hoạt động nếu không được kết nối mạng.

```
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.0/angular.min
.js"></script>
```

2.2.2. Ví dụ đơn giản

```
<!doctype html>
<html lang="en" ng-app>
<head>
  <meta charset="utf-8">
  <title>Ví dụ AngularJS </title>
  <script src="js/angular.js"></script>
</head>
<body>
  <p>Hello {{'World' + '!'}}</p>

</body>
</html>
```

Phân tích ví dụ:

ng-app directive:

Directives là các tùy biến gắn vào các DOM (attribute, element name, CSS class …) để HTML compiler của AngularJS có thể thêm vào hoặc biến đổi DOM element.

```
<html ng-app>
```

Ng-app cho biết phạm vi hoạt động của AngularJS. Chỉ thị này cho biết vị trí bắt đầu để sử dụng các phần tử của AngularJS.

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

- Chèn AngularJs vào ứng dụng.

```
<p> Hello {{'World' + '!'}} </p>
```

Một đặc điểm quan trọng của AngularJs là khả năng binding dữ liệu sử dụng cặp dấu ngoặc nhọn {{}}. Đoạn mã này sẽ xuất ra dữ liệu là chuỗi “Hello World !” ra trình duyệt.

Trong dấu ngoặc nhọn có thể chứa biến và các biểu thức toán học.

```
<p>1 + 2 = {{ 1 + 2 }}</p>
```

Đoạn mã này sẽ in lên trình duyệt chuỗi "1 + 2 = 3"

2.2. Các thành phần của AngularJs

2.2.1 Angular Template

Đôi lúc trong quá trình xây dựng hệ thống, file HTML trở nên phức tạp để giải quyết vấn đề này ta cần chia thành nhiều phần khác nhau, AngularJS cung cấp cho chúng ta một giải pháp hữu ích đó là template.

- Angular template là 1 đặc tả dạng declarative (khai báo), cùng với thông tin từ Model & Controller, trở thành rendered View mà User thấy trên trình duyệt. Nó là static DOM, chứa HTML, CSS, và các thành phần, thuộc tính riêng của Angular. Các thành phần Angular và các thuộc tính giúp Angular thêm các hành vi và biến đổi template DOM thành dynamic view DOM.

- Nói 1 cách đơn giản nhất thì template trong AngularJS là "HTML voi81 additional markup".

Trong Angular, chúng ta có 2 cách để tạo một template.

Dùng file ngoài:

Chúng ta có thể dùng thêm một file html bên ngoài để làm template cho file chính, ví dụ:

message.html

```
<h3>Hello, {{name}}!</h3>
```

Dùng script

Chúng ta có thể tích hợp thẳng template vào file hiện hành thông qua thẻ script với type là text/ng-template

```
<script type="text/ng-template" id="message.html">
    <h3>Hello, {{name}}!</h3>
</script>
```

Cách sử dụng template

Có nhiều cách để sử dụng template, tuy nhiên trong AngularJS có 2 cách thông dụng nhất để dùng template đó là ng-include và ngRoute (sẽ nói ở phần sau), ví dụ:

index.html

```
<!doctype html>
<html ng-app="ExampleModule">
<head>
    <script
src="http://code.angularjs.org/1.2.12/angular.min.js"></script>
    <script src="script.js"></script>
</head>
<body>
    <div ng-controller="ExampleCtrl">
        <input type="text" ng-model="name">
        <div ng-include src="template_name"></div>
    </div>
    <script type="text/ng-template" id="message.html">
        <h3>Hello, {{name}}!</h3>
    </script>
</body>
</html>
```

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

app.controller('ExampleCtrl', function($scope) { $scope.template_name = 'message.html'; $scope.name = 'World'; });

2.2.2. Modules

Trong các ứng dụng thực tế, việc phân chia ứng dụng thành các thành phần khác nhau là điều cần thiết. Dưới đây là lợi ích của việc chia nhỏ ứng dụng:

- Dễ dàng hơn cho việc quản lý và khai báo ứng dụng cũng như kiểm tra lại sau này
- Khả năng tái sử dụng cũng như tận dụng các 3rd modules
- Nạp từng phần ứng dụng sẽ nhanh hơn là buộc phải nạp toàn bộ ứng dụng vào rồi mới chạy.

Trong AngularJS, module được hỗ trợ trong khai báo ng-app bên cạnh khai báo nó trong mã nguồn script, dưới đây là một template chuẩn của angular sử dụng modules:

```
<!doctype html>
<html ng-app="myModule">
<head>
  <title>Module trong AngularJS </title>
  <script src="js/angular.js"></script>
<script src="js/app.js"></script>
</head>
<body>  <p>Ví dụ đơn giản về module p>
</body>
</html>
```

Trong ví dụ này module được khai báo ngay trong thẻ <html> với tên là myModule.

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

- ng-app="myModule": Khai báo một angular app là myModule, sử dụng myModule được khai báo trong script.
- Trong script, angular.module() là hàm khai báo cho module.
- cặp dấu ngoặc [] chính là mảng các module sẽ được nạp chung vào để chạy chung với ứng dụng (module chính được khai báo trong ng-app).Ví dụ:

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

Controller trong module.

Trong ví dụ trên chúng ta đã thấy việc khai báo module như thế nào, vậy controller khi ứng dụng sẽ được khai báo như thế nào. Hãy xem ví dụ dưới đây

index.html

```
<!doctype html>
<html ng-app="ExampleModule">
<head>
    <script
src="http://code.angularjs.org/1.2.12/angular.min.js"></script>
    <script src="script.js"></script>
</head>
<body>
    <div ng-controller="ExampleCtrl">
        Hello, {{ name }}!
    </div>
</body>
</html>
```

script.js

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

app.controller('ExampleController', function($scope) {
    $scope.name = 'World';
});
```

• Method .controller của module sẽ đóng vai trò khai báo thêm một controller cho module.

• Function đại diện cho controller được khai báo bình thường giống như controller khai báo bên ngoài module.

- AngularJS sử dụng **Dependence Injection** để tách biệt giữa các modules.

- Các dependency được đưa vào tự động bởi framework

Tổ chức của một ứng dụng thực tế

Thông thường thì tổ chức một ứng dụng thực tế sẽ được khởi tạo như sau

```
├── index.html
├── css
│   └── style.css
└── scripts
    ├── module_1.js
    ├── module_2.js
    ├── module_3.js
    └── main.js
```

- **index.html** chính là html documents
- **style.css** chính là mã nguồn css cho document
- **main.js** chính là mã nguồn cho module chính
- các files js khác là các modules được module chính sử dụng, như vậy template của chúng ta sẽ được khai báo lại như sau

index.html

```
<!doctype html>
<html ng-app="ExampleModule">
<head>
    <link rel="stylesheet" href="css/style.css">
    <script
src="http://code.angularjs.org/1.2.12/angular.min.js"></script>
    <script src="scripts/module_01.js"></script>
    <script src="scripts/module_02.js"></script>
    <script src="scripts/module_03.js"></script>
    <script src="main.js"></script>
</head>
<body>
    <div ng-controller="ExampleCtrl">
        Hello, Angular's Module!
    </div>
</body>
</html>
```

main.js

```
var app = angular.module('ExampleModule', ['Module1', 'Module2', 'Module3']);

app.controller('ExampleController', function($scope) {
    $scope.name = 'World';
});
```

2.2.3. Scope

'$scope' là đối tượng tham chiếu tới model được sử dụng trong controller. Là thành phần gắn kết giữa View và Controller. Scope cung cấp các APIs để theo dõi các thay đổi của Model : **$watch**

Scope cung cấp các APIs để truyền bất kỳ thay đổi nào của Model tới View : $**apply**

Tự động đồng bộ dữ liệu giữa Model và View

2.2.4. Model

```
<input type="text" ng-model="greeting">
  {{ greeting }}
```

Ng-model cho biết đối tượng nhận giá trị của input. {{greeting}} sẽ in ra giá trị được nhập vào input.

2.2.5. Controller

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


demo.controller('GreetingController', ['$scope', function($scope) {
  $scope.greeting = 'Hello World!';
}]);
</script>
```

Tạo một module cho ứng dụng:

```
<html ng-app="myApp">
```

- Đặt biến "demo" chứa module "myApp" của ứng dụng.

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

- Tạo một controller mới cho module này

```
demo.controller('GreetingController', ['$scope', function($scope) {
  $scope.greeting = 'Hello World!';
}]);
```

'GreetingController' là tên của controller, cách đặt tên nên chứa tên và đuôi "Controller" để dễ nhận biết.

```
['$scope', function($scope)
```

Viết hàm sử dụng với biến $scope.

```
$scope.greeting = 'Hello World!';
```

Gán giá trị cho biến greeting. Để in được giá trị của greeting ra trình duyệt.

```
<div ng-controller="GreetingController">
  {{ greeting}}
</div>
```

Ng-controller cho biết controller đang hoạt động. Đoạn mã này sẽ in ra trình duyệt chuỗi "Hello World! " được khai báo trong "GreetingController".

2.2.6.Expression (Biểu thức)

Expression được sử dụng trong tất cả các phần của AngularJs, vì vậy cần phải nắm vững về expression và cách mà Angular sử dụng, tính toán. Nó được bao bởi kí hiệu {{ }}. Một biến được khởi tạo từ controller thông qua $scope cũng được coi là một expression.

- Expression các snippets giống JS được đặt trong bindings, kiểu như {{expression}}.

- Được xử lý bởi $parse service.

Ví dụ: <body> 2 + 2 = {{2 + 2}} </body>

Angular sẽ tính toán giá trị trong biểu thức và xuất ra màn hình kết quả "2 + 2 = 4".

2.2.7. Filters (Bộ lọc)

Trong Angular, một filter cung cấp một định dạng dữ liệu để hiển thị tới người dùng.

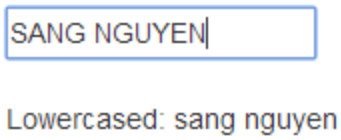
Theo tinh thần của UNIX filters và sử dụng các cú pháp tương tự | (pipe).

Angular cung cấp một số filter được xây dựng sẵn như: lowercase, date, number, currency, limitTo, orderBy…

Ví dụ:

```
<input ng-model="userInput"> Uppercased: {{ userInput | uppercase }}
```

Khi người dùng nhập dữ liệu vào input thì giá trị đó sẽ được chuyển thành chữ viết thường.



Một số filter khác:

```
{{ "chuỗi kí tự viết thường" | uppercase }}
{{ {name: "Sang", age: 21} | json }}
{{ 1304375948024 | date }}
{{ 1304375948024 | date:"MM/dd/yyyy @ h:mma" }}
```

Kết quả:

CHUỖI KÍ TỰ VIẾT THƯỜNG

{ "name": "Sang", "age": 21 }

May 3, 2011

05/03/2011 @ 5:39AM

2.2.8. Directives

Những thứ như thuộc tính, class, tên ... của 1 DOM element gọi chung là directive. AngularJS sẽ dựa vào chúng để attach các chỉ thị hoặc các sự kiện tới DOM element đó.

Bản thân AngularJS đã có rất nhiều các directive : ng-bind, ng-model, ng-click, ng-controller...

Việc dùng directive sẽ giảm thiểu được số lượng thẻ HTML , code HTML nhìn sẽ gọn gàng và sáng sủa hơn.

AngularJS cung cấp cho chúng ta 3 loại directive đó là :

- Directive dạng element (là 1 thẻ HTML) viết tắt là E.

```
<my-directive></my-directive>
```

- Directive dạng attribute (thuộc tính của 1 thẻ HTML) viết tắt là A, dạng này là mặc định.

```
<div my-directive></div>
```

- Directive dạng class(class CSS) viết tắt là C.

```
<div class="my-directive"></div>
```

Ví dụ:

Chuẩn bị tài liệu HTML với 3 đủ 3 loại directive:

```
<!doctype html>
<html ng-app>
<head>
  <title>Ví dụ AngularJS </title>
  <script src="js/angular.js"></script>
</head>
<body>
<div ng-controller="myController">
        <my-directive></my-directive>
        <div my-directive></div>
        <div class="my-directive"></div>
```

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

AngularJS

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

    app.controller('myController', function($scope) {
        $scope.customer = {
            name: 'Sang',
            address: 'Quang Trung, Go Vap'
        };
    })
    .directive('myDirective', function() {
        return {
            restrict : 'C',
            template: 'Name: {{customer.name}}
                      Address: {{customer.address}}'
            // templateUrl: 'directive_template.html'
        };
    });
```

Trong myCtrl khai báo thông tin customer với name và địa chỉ.

Trong myDirective sẽ return ra 1 object còn việc hiển thị hay đổ dữ liệu vào directive như nào là việc của Angular chúng ta không cần quan tâm.

Lưu ý: Tên directive phải khai báo dạng camelCase còn khi gọi thì có thể dùng dấu - để ngăn cách giữa các từ hoặc để nguyên như khi khai báo cũng được

- restrict : loại directive mặc định là E(element), C là class, A là attribute
- template: Nội dung của directive

- templateUrl: Liên kết tới 1 file template bên ngoài. Template này chứa nội dung của directive.

```
Name: {{customer.name}} Address: {{customer.address}}
```

- Nếu restrict:"A" thì <my-directive></my-directive> sẽ có dữ liệu
- Nếu restrict:"C" thì sẽ có dữ liệu
- Nếu restrict:"E" thì <div my-directive> sẽ có dữ liệu

Khi thay đổi restrict cần lưu ý infect element để xem nó đổ dữ liệu vào thẻ nào.

2.2.9. Services

AngularJS service là một object hoặc một function phụ trách một tác vụ nào đó. Việc viết service chỉ là việc gom các đoạn xử lý logic vào object hoặc function để dễ quản lý hơn. Cũng giống như mô hình MVC tách phần xử lý, điều hướng và hiển thị riêng biệt.

Trong AngularJS service sẽ chứa tất cả các phần xử lý, Controller nhận yêu cầu và gọi các service cần thiết để xử lý, Model handle dữ liệu từ các control, View hiển thị dữ liệu, Route điều hướng request tới controller.

Bản thân AngularJS cũng chứa những service như: $http, $scope, $window, $RouteProvider ...

Có 2 cách để tạo service trong Angularjs.

Cách 1:

Cú pháp:

```
module.service( 'serviceName', function );
```

Ví dụ:

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

module.service('userService', function(){
   this.users = ['Sang', 'Nguyen', 'Minh'];
});
```

Cách 2:

Cú pháp:

```
module.factory('userService', function(){
```

Ví dụ:

```
    var fac = {};
    fac.users = ['John', 'James', 'Jake'];
    return fac;
});
```

Cả 2 cách đều tạo ra được service.

Với cách 1 sau khi đăng ký tên service xong bạn sẽ được cung cấp 1 thể hiện của function mà bạn truyền vào module.service.

Với cách 2 sau khi đăng ký tên factory, trong function truyền vào trong module.factory cần return thể hiện của đối tượng tạo bên trong nó.

2.2.10. Multiple Views and Routing

Đôi khi trong một trang, nhiều khi chúng ta chỉ muốn hiển thị một phần HTML ứng với mỗi chức năng cụ thể mà ta không cần chuyển đổi trang, Angular là một full-stack framework hiệu quả giúp chúng ta có thể làm được việc này nhanh chóng và dễ dàng.

Route là bộ điều hướng các yêu cầu từ phía người dùng tới các controller tương ứng để xử lý dựa theo các đối số truyền trên thanh url. Route trong AngularJS là thành phần quan trọng giúp AngularJs tạo được ứng dụng SPA, chuyển trang mà không cần tải lại trình duyệt.

Trong AngularJs chúng ta sẽ sử dụng $routeProvider để bắt các yêu cầu. Biểu thức route được tính từ sau dấu #.

Xét ví dụ

```
<!doctype html>
<html lang="en" ng-app="demoApp" >
<head>
  <meta charset="utf-8">
  <title>Ví dụ AngularJS </title>
  <script src="js/angular.min.js"></script>
<script src="js/angular-route.min.js"></script>
</head>
<body>
  <div ng-view></div>
</body>
</html>
```

Từ phiên bản AngularJs 1.0.7 thì Route đã được tách thành một file js riêng biệt, và để sử dụng được nó ta phải nhúng file angular-route.min.js vào ứng dụng.

Directive ng-view được sử dụng để hiển thị dữ liệu.

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

Đăng ký biến app thành một module trong AngularJS và thiết lập route cơ bản. Để làm việc với Route thì bạn cần gọi và sử dụng một extends module của angular là ngRoute.

```
var app = angular.module('demoApp', ['ngRoute']);
app.config(['$routeProvider', function($routeProvider){
    $routeProvider
        .when('/, {
            templateUrl : index.html',
            controller : 'homeController'
        })
       .when('/post/:id', {
            templateUrl : 'detail.html',
```

```
            controller : 'postController'
        })
        .otherwise({
            redirect: '/'
        })
}]);
```

- **app.config** là method cho phép khai báo các Controller, View tương ứng với url

Với route thế này thì khi truy cập vào đường dẫn chính của trang web thì trình duyệt sẽ load file index.html và bind nó vào <div ng-view> .

```
.when('/post/:id', {
```

Trong route tới chi tiết bài đăng có sử dụng $routeParams service dùng để nhận các đối số là id của bài viết, giúp xác định được chính xác bài viết cần hiển thị. Đường dẫn truy cập tới chi tiết bài đăng sẽ có dạng #/post/id. Trong đó id là id của bài đăng. Khi cấu hình route không cần phải ghi dấu "#".

Khi gặp đường dẫn có dạng #/post/id thì route sẽ gọi tới controller là postController và sử dụng template từ file detail.html.

```
       .when('/post/:id', {
            templateUrl : 'detail.html',
            controller : 'postController'
        })
```

$routeProvider.otherwise xử lý cho route mặc định. Khi đường dẫn không khớp với những route đã được thiết lập thì sẽ được tái điều hướng về trang chủ.

```
        .otherwise({
            redirect: '/'
        });
```

2.2.11. Form validation

- Controls (input, select, textarea) là các cách mà người dùng nhập dữ liệu.

- Form là một tập các controls với mục đích nhóm các controls liên quan với nhau.

- Form và controls cung cấp các validation services, để người dùng được báo các lỗi liên quan đến nhập dữ liệu.

- Server – side validation cũng cần thiết để đảm bảo độ an toàn của ứng dụng.

- Sử dụng thuộc tính "novalidate" để tắt chức năng validation mặc định của trình duyệt.

- CSS Classes:

- **ng-valid**: class được thêm vào phần tử nếu kiểm tra.
- **ng-invalid**: class được thêm vào phần tử nếu không kiểm tra.
- **ng-pristine**: class được thêm vào phần tử lúc ban đầu, trước khi AngularJS xử lý kiểm tra.
- **ng-dirty**: class được thêm vào phần tử khi AngularJS xử lý kiểm tra.

- Custom Validation

- Angular cung cấp các xử lý cơ bản cho hầu hết các kiểu input HTML5: (text, number, url, email, radio, checkbox) kèm directives để kiểm tra (required, pattern, minlength, maxlength, min, max).
- Có thể tự đưa ra validate riêng bằng cách tự tạo directive để đưa hàm validate của mình vào ngModel controller.

- Validation có thể xuất hiện ở 2 chỗ

- Model to View update: khi Model thay đổi, tất cả các hàm trong **NgModelController#$formatters** array được pipe – lined, để mỗi hàm này có thể định dạng được giá trị và thay đổi trạng thái valid của form control thông qua **NgModelController#$setValidity**.
- View to Model update: tương tự như vậy, khi người dùng tương tác với 1 control, nó gọi **NgModelController#$setViewValue**. Nó sẽ pipe – line tất cả các hàm trong

NgModelController#$parsers array, để mỗi hàm này lần lượt chuyển đổi giá trị và trạng thái thay đổi của form control thông qua **NGModelController#$setValidity**.

2.3. Lập trình AngularJS phía server với Node.js

2.3.1.Node.js là gì?

Node.js là 1 nền tảng (platform) chạy trên môi trường V8 Javascript runtime. Node.js cho phép lập trình viên xây dựng các ứng dụng có tính mở rộng cao sử dụng Javascript trên server. Và vì được porting từ C nên về mặt tốc độ xử lý thì khá nhanh.

2.3.2.Node.js có thể làm được những gì?

- Xây dựng websocket server (Chat server)

- Ứng dụng upload file rất nhanh trên client

- Ad server

- Hoặc bất kỳ ứng dụng dữ liệu thời gian thực nào.

Nodejs không phải là một web framework.Nó không dành cho người mới bắt đầu, không phải là một nền tảng thực thi các tác vụ đa luồng.

2.3.3. Block code và Non-block code

Ví dụ xây dựng chức năng đọc file và in ra dữ liệu của file.

Logic

Đọc file từ Filesystem, gán dữ liệu tương ứng với biến "contents"

In dữ liệu biến "content"

Thực hiện công việc khác tiếp theo.

Non-block code:

Đọc file từ Filesystem

Sau khi đọc xong thì in dữ liệu (callback)

Thực hiện công việc khác tiếp theo.

Code

Block code

```
var contents = fs.readFileSync('hello.txt'); // Dừng cho đến khi đọc xong file.
console.log(contents);
console.log('Thực hiện công việc khác');
```

Non-block code

```
fs.readFile('hello.txt', function(contents){
   console.log(contents);
});
console.log('Thực hiện công việc khác');
```

Ta có thể thấy ở đây, tốc độ xử lý của non-block code là cao hơn so với block code. Giả sử bạn thực hiện công việc trên ở 2 file trở lên thì tốc dộ xử lý của Non-block code sẽ nhanh hơn Block code rất nhiều.

2.3.4.Ứng dụng đầu tiên

```
hello.js
```

```
var http = require('http'); // đây là cách chúng ta require các modules
http.createServer(function(request, response){
    response.writeHead(200, {'Content-Type':'text/plain'}); // Status code và content type
    response.write("Xin chào lập trình viên!"); // Thông điệp được gửi xuống client.
    response.end(); // Đóng kết nối
}).listen(3000); // Chờ kết nối ở cổng 3000.
console.log("Server đang chờ kết nối tại cổng 3000");
```

Chạy server: `node hello` hoặc `node hello.js` --> *Server đang chờ kết nối tại cổng 3000*

Mở trình duyệt và truy cập tới địa chỉ `http://localhost:3000` hoặc dùng terminal: `curl http://localhost:3000` --> *Xin chào lập trình viên*

2.4. Công cụ lập trình với AngularJS

2.4.1 Yeoman

- Gồm 3 ứng dụng:

- Yo
- Grunt
- Bower

- Yeoman không chỉ là tool mà còn được sử dụng như là một Workflow, tập hợp các "best practices" để giúp cho việc phát triển Web dễ dàng hơn.

- Lightning – fast scaffolding

- Dễ dàng tạo khung cho những dự án mới với các template tùy biến được (vd:HTML5 Boilerplate, Bootstrap), Angular JS...

Great buil process

- Minification and concatenation (thu nhỏ và cố kết?)
- Tối ưu tất cả image files, HTML
- Compile CoffeScript and Compass files

- Automatically lint your script

- Tất cả Script đề được tự động chạy qua JSHint để đảm bảo nó tuân theo những best-practices

- Built-in preview server

- Tối ưu ảnh cực tốt

- Dùng OptiPNG và JPEGTran

- Package management cực tốt

- Dễ dàng search được package mới thông qua command-line, cài đặt và update chúng mà không cần mở trình duyệt

- PhantomJS Unit Testing

- Dễ dàng chạy unit test trong WebKit thông qua PhantomJS

- Các công cụ hỗ trợ tốt cho việc lập trình Angular JS: Sublime Text, WebStorm, Emacs..

2.4.2. WebStorm

- Trang chủ: http://www.jetbrains.com/webstorm/

- Download: http://download-cf.jetbrains.com/webstorm/WebStorm-8.0.2.exe

- Feature: http://www.jetbrains.com/webstorm/features/

- Phiên bản mới nhất là 8.0.2

- Là phần mềm có phí.

- Là IDEs duy nhất có hỗ trợ plug-in Angular JS.

- Ở WebStorm bạn chỉ cần gõ: ngdc [TAB].

```
directive('$directiveName$', function factory($injectables$) {
var directiveDefinitionObject = {
$directiveAttrs$
compile: function compile(tElement, tAttrs, transclude) {
$END$
return function (scope, element, attrs) {
}
}
};
return directiveDefinitionObject;
});
```

2.5. Khởi chạy ứng dụng

- Sử dụng Yeoman:

- Yeoman cung cấp một cách đơn giản để chạy một Web Server và đáp ứng tất cả các yêu cầu của các tập tin liên quan đến Angular JS.
- Bạn chỉ cần gõ vào command-line: yeoman server
- Web Server sẽ tự động được cài đặt và mở trình duyệt với trang chủ Angular JS của bạn. Và chương trình sẽ tự động được cập nhật khi thay đổi code trong Angular JS.

- Không sử dụng Yeoman:

- Bạn cần một Web Server để khởi động trình duyệt (Xampp, Wamp, AppServ...)
- Bạn cần lưu ý là phải Refresh lại trình duyệt để xem những thay đổi khi có bất kỳ điều chỉnh nào về code trong Angular JS, không giống như khi sử dụng Yeoman.

2.6. Testing and Debug

- Việc Testing là thực sự cần thiết với mỗi ứng dụng cũng như mỗi Website.

- Angular JS cung cấp các công cụ test thực sự mạnh mẽ, trong số đó là Karma.

2.6.1. Karma

- Thực hiện những bài test TDD (test-driven development).

- Karma sử dụng NodeJSand SocketIO để thực thi ứng dụng của bạn và test trên nhiều trình duyệt một cách nhanh chóng.

- Trang chủ: https://github.com/vojtajina/karma/

2.6.2. Cài đặt Karma

- Cần cài đặt NodeJS và NPM, vào command-line và gõ: sudo npm install -g karma

2.6.3. Test với Karma

- Gồm 3 bước:

- Định dạng cấu hình của Angular JS:

- Nếu bạn sử dụng Yeoman, Karma sẽ tự động nhận được cấu hình của Angular JS.
- Nếu không sử dụng Yeoman, bạn chỉ cần gõ vào command-line: karma init

- Kết nối server cho Karma
  - Nhập vào command-line: karma start [optionalPathToConfigFile]
  - Karma sẽ tự động chạy server ở port 9876 (đây là port mặc định, nếu bạn muốn đổi thì chỉ cần thay đổi trong file karma.conf.js)
- Test
  - Nhập vào command-line: karma run

TÀI LIỆU THAM KHẢO

[1]. AngularJS Cheat Sheet (2013) – *tác giả* ProLoser – 5 trang.

[2]. Professional Node.js(2012) – *tác giả* Pedro Teixeira – 412 trang.

[3]. Ng-book The Complete Book on AngularJS (2013) – *tác giả* Ari Lerner – 608 trang.

[4]. AngularJS (2013) – *tác giả* Brad Green, Shyam Seshadri – 196 trang.

[5]. Single page web applications JavaScript end-to-end (2013) – *tác giả* Michael Mikowski, Josh Powell – 433 trang.