**TẬP ĐOÀN BƯU CHÍNH VIỄN THÔNG VIỆT NAM**

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

------------------------------



**BÁO CÁO THỰC TẬP**

**TỐT NGHIỆP ĐẠI HỌC**

**Đề tài:**

**“XÂY DỰNG WEBSITE TRAO ĐỔI MUA BÁN HÀNG ONLINE”**

**Người hướng dẫn : ThS. Nguyễn Anh Hào**

**Sinh viên thực hiện : Nguyễn Quốc Thành**

**Mã số sinh viên : N12DCCN041**

**Lớp : D12CQPM01-N**

**Khoá : 2012-2017**

**Hệ : ĐẠI HỌC CHÍNH QUY**

**TP.HCM, tháng 8 /2016**

**PHIẾU GIAO ĐỀ CƯƠNG THỰC TẬP TỐT NGHIỆP ĐƯỢC PHÊ DUYỆT.**

**LỜI MỞ ĐẦU**

**LỜI CẢM ƠN**

Trong quá trình thực hiện đề tài thực tập tốt nghiệp này, em đã nhận được sự hỗ trợ, giúp đỡ của rất nhiều các cá nhân và tổ chức trong và ngoài trường.

Trước hết em xin bày tỏ sự biết ơn sâu sắc tới giảng viên Ths.Nguyễn Anh Hào– Người đã trực tiếp hướng dẫn em trong suốt quá trình nghiên cứu và thực hiện đề tài.

Em xin chân thành cảm ơn các thầy giáo, cô giáo trong khoa Công Nghệ Thông Tin 2 trường Học Viện Công Nghệ Bưu Chính Viễn Thông thành phố Hồ Chí Minh đã tạo mọi điều kiện thuận lợi giúp đỡ em trong suốt quá trình học tập và rèn luyện tại trường.

Mặc dù trong quá trình nghiên cứu đề tài, bản thân em đã có những cố gắng, song do trình độ và thời gian hạn chế nên đề tài không tránh khỏi những thiếu sót. Vậy kính mong các thầy cô giáo và các bạn đóng góp ý kiến để đề tài của em được hoàn thiện hơn và rất mong bài nghiên cứu của em sẽ trở thành một trong những bài nghiên cứu có ích cho mọi người.

TP.Hồ Chí Minh, tháng 08 năm 2016

Sinh viên

NGUYỄN QUỐC THẢNH

**CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**

**Độc lập- Tự do- Hạnh phúc**

TP. Hồ Chí Minh, ngày tháng năm 2016

**NHẬN XÉT CỦA ĐƠN VỊ THỰC TẬP**

**THỰC TẬP TỐT NGHIỆP ĐẠI HỌC**

1. **Tên đề tài: “XÂY DỰNG WEBSITE TRAO ĐỔI MUA BÁN HÀNG ONLINE.”**
2. **Sinh viên: NGUYỄN QUỐC THÀNH Lớp: D12CQPM01-N**
3. **Giáo viên hướng dẫn: Thạc sĩ NGUYỄN ANH HÀO**
4. **Nợi thực tập: Học viện công nghệ bưu chính viễn thông cơ sở phía Nam**

**NỘI DUNG NHẬN XÉT**

**Xác nhận của cơ sở thực tập**

*(Ký, ghi rõ họ tên, đóng dấu)*

**CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**

**Độc lập- Tự do- Hạnh phúc**

TP. Hồ Chí Minh, ngày tháng năm 2016

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

**THỰC TẬP TỐT NGHIỆP ĐẠI HỌC**

1. **Tên đề tài: “XÂY DỰNG WEBSITE TRAO ĐỔI MUA BÁN HÀNG ONLINE.”**
2. **Sinh viên: NGUYỄN QUỐC THÀNH Lớp: D12CQPM01-N**
3. **Giáo viên hướng dẫn: Thạc sĩ NGUYỄN ANH HÀO**
4. **Nợi thực tập: Học viện công nghệ bưu chính viễn thông cơ sở phía Nam**

**NỘI DUNG NHẬN XÉT**

**1. Đánh giá chung:**

**2. Đánh giá chi tiết:**

**3. Nhận xét tinh thần, thái độ làm việc:**

**4. Kết luận:**

**5. Điểm hướng dẫn:**

**GIẢNG VIÊN HƯỚNG DẪN**

*(Ký, ghi rõ họ tên)*

**MỤC LỤC**

[**CHƯƠNG I: ANGULAR JS FRAMEWOR**](#page3)**K** 1

[**1.1. Ngôn ngữ lập trình web JavaScrip**](#page3)**t** 1

[1.1.1. Lịch sử phát triể](#page3)n 1

[1.1.2. Javascript là gì](#page3)? 1

[1.1.3. Javascript có thể làm được những gì](#page3)? 1

[**1.2. Tổng quan về AngularJs framewor**](#page3)**k** 1

[1.2.1 Lịch sử phát triể](#page3)n 1

[1.2.2. Angular JS là gì](#page5)? 3

[1.2.3. Các tính năng chín](#page5)h 3

[1.2.4. Đặc trưn](#page6)g 4

[1.2.5. SPA – Single Page Applicatio](#page7)n 5

[**1.3. Tại sao phải sử dụng AngularJs**](#page8)**?** **6**

[**1.4. AngularJs được lập trình như thế nào**](#page8)**?** **6**

[1.4.1. Cài đặt AngularJ](#page8)s 6

[**1.5. Các thành phần của AngularJ**](#page10)**s** 8

[1.5.1 Angular Templat](#page10)e 8

[1.5.2 Data binding trong angularjs](#page10) 8

[1.5.3 Controller trong angularjs](#page10) 8

[1.5.4 Service trong angularjs](#page10) 8

[1.5.5. Module](#page11)s 9

[1.5.6. Scop](#page15)e  [1](#page15)3

[1.5.7. Mode](#page15)l  [1](#page15)3

[1.5.8.Expression (Biểu thức](#page16))  [1](#page16)4

[1.5.9. Filters (Bộ lọc](#page17))  [1](#page17)5

[1.5.10. Directive](#page18)s  [1](#page18)6

[1.5.11. Multiple Views and Routin](#page21)g  [1](#page21)9

[**CHƯƠNG II: ASP.NET WEB API**](#page3) 1

[**2.1.**](#page3) **Giới thiệu web api** 1

[2.1.1](#page3)  Giới thiệu 1

[2.1.2.](#page5) Những lợi ích khi sử dụng ASP.NET Web API 3

[2.1.3.](#page5)  Một số đặc điểm nổi trội. 3

[**2.2.**](#page3) **ASP.NET Web API 2.2** 1

[2.2.2.](#page5) Entity Framework 3

[2.2.3.](#page5) Identity trong ASP.NET Web API 3

[2.2.4.](#page3) Entity Framework Code First 1

[2.2.5.](#page5) Respository and unit of work 3

[2.2.6.](#page5) Bảo mật trong ASP.NET Web API 3

[**TÀI LIỆU THAM KHẢ**](#page31)**O**  [2](#page31)9

**DANH MỤC CÁC KÝ HIỆU VÀ CHỨ VIẾT TẮT**

**DANH MỤC CÁC BẢNG VẼ**

**DANH MỤC CÁC HÌNH VẼ**

**KÍ HIỆU CÁC CỤM TỪ VIẾT TẮT**

**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.

**1.2.6. 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.

**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](https://angularjs.org/) hoặc  [https://github.com/angular/angular.j](https://github.com/angular/angular.js)s 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**

C:\Users\leduc\Desktop\thanh\1.PNG

* + 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.

C:\Users\leduc\Desktop\thanh\Capture.PNG

**2.2.2. Ví dụ đơn giản**

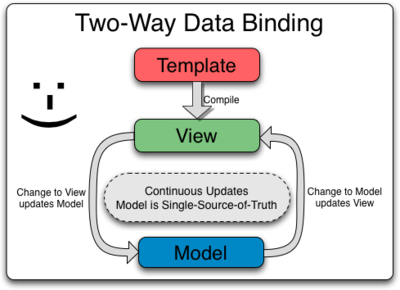


**2.2.2 Data binding trong angularjs**

### **-** Data Binding trong AngularJS là gì ?

Data Binding trong AngularJs chính là các thức tự động đồng bộ (synchronization) dữ liệu giữa Model và View trong AngularJs. Data Binding cho phép bạn tạo sự liên kết, ràng buộc giữa Model và Template thông qua một ứng dụng AngularJs (ng-app). Khi có bất kỳ sự thay đổi nào ở model, dữ liệu sẽ được phản ánh ngay tức thì ra Template và ngược lại.

### **-** Cách thức hoạt động của Data Binding trong AngularJs



Ở hình trên bạn sẽ thấy rằng, đầu tiên AngularJS compile Template ra ngoài View cho người dùng thấy được (chẳng hạn như là 1 ô textbox). Khi có bất kì thày đổi trên View này, chẳng hạn như trong ô textbox mình sẽ nhập là abc. Thì tự động dữ liệu abc này sẽ ngay lập tự được đồng bộ vào bên trong Model. Và tiếp theo bên trong Model nếu có bất kỳ sự thay đổi nào sẽ ngay lập tức được đồng bộ ngược lại ra View. Vì thế dữ liệu được trung chuyển từ Model sang View hay từ View sang Model là một thể thống nhất.

**2.2.3. Controller trong angularjs**

<script>

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

demo.controller('ExampleController', ['$scope', function($scope) { $scope.hello = '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('ExampleController', ['$scope', function($scope) {

$scope.hello = 'Hello World!';

}]);

'ExampleController'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.hello = 'Hello World!';

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

<div ng-controller=" ExampleController">

{{ hello}}

</div>

ng-controller cho biết controller đang hoạt động. Đoạn mã này sẽ in ra trình duyệt chuỗi “Hello World! ” được khai báo trong “ExampleController”.

**2.2.4. 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 ...

**Cách để tạo service trong Angularjs.**

Cú pháp:

module.service( 'serviceName', function );

Ví dụ:

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

module.service('userService', function(){ this.users = [Thanh, 'Nguyen', 'Bao'];

});

**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']);

**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.

**2.2.6.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: "Thanh", age: 21} | json }} {{ 1304375948024 | date }}

{{ 1304375948024 | date:"MM/dd/yyyy @ h:mma" }}

**Kết quả:**

CHUỖI KÍ TỰ VIẾT THƯỜNG

{ "name": "Thanh", "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 thẻHTML ) 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>

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 <div class="my-directive"></div> sẽ có dữ liệu
* Nếu restrict:"E" thì <div my-directive></div> sẽ có dữ liệu

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

**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 #.

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.

 **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></div>.

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.

$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.

**CHƯƠNG III: ASP.NET WEB API**

**2.1 Giới thiệu Asp.net web api**

2.2.1 **Giới thiệu**

Cùng với việc giới thiệu các công nghệ dịch vụ (service) trong thập niên gần đây, Microsoft đã tạo nên một chuỗi các công nghệ cho phép dễ dàng xây dựng và hỗ trợ các dịch vụ web trên nền tảng .NET. Bắt đầu với XML Web Service, rồi sau đó được mở rộng thông qua  Windows Communication Foundation vài năm sau đó, Microsoft đã cho phép các nhà phát triển .NET có nhiều lựa chọn để xây dựng các dịch vụ trên cơ sở SOAP. Chỉ với một số thay đổi nhỏ trong cấu hình, sẽ cho phép bạn hỗ trợ rất nhiều thiết bị đa dạng về giao thức kết nối, cơ chế xác thực, định dạng thông báo cùng các chuẩn hỗ trợ bảo mật WS-\* với WCF. Nhưng khi thế giới các thiết bị kết nối phát triển, kéo theo nhu cầu về các giao tiếp đơn giản HTTP – chỉ sử dụng nền tảng dịch vụ - bỏ qua các yếu tố phức tạp như xác thực liên đoàn (federated authentication) hay mã hóa thông báo, cũng như không cần phải tạo phiên giao dịch (transaction), khám phá dịch vụ trên cơ sở Ngôn ngữ mô tả dịch vụ (Web Services Description Language – WSDL), …  
Theo một cách đơn giản, đó là các giao tiếp dịch vụ di động (mobile-to-service) và trên cơ sở trình duyệt, các ứng dụng trang đơn lẻ (Single-Page Application – SPA). Khi đó các nhà phát triển có thể tạo các ứng dụng trên nền JavaScript, các ứng dụng mobile giá rất rẻ (nhỏ hơn 1 USD) – và các ứng dụng đó chỉ cần giao thức HTTP đơn giản, giao tiếp dữ liệu backend tương thích JSON.  
Đáp lại đó, Microsoft đã tạo ra ASP.NET Web API, một nền tảng đầy hiệu quả mà rất đơn giản với giao thức HTTP và dịch vụ web theo mặc định sử dụng JSON. Các model làm việc  để xử lý và trả về được đóng gói qua các đối tượng đơn giản (Plain Old CLR Objects).

2.2.2 **Những lợi ích khi sử dụng ASP.NET Web API**

* Cấu hình

Ngay khi bạn xây dựng một website, không cần bất kỳ cấu hình gì chúng ta đã có thể cài đặt và sử dụng dịch vụ trên nền ASP.NET Web API. Khái niệm điểm đầu cuối (xuất hiện trong WCF) không còn, và cũng không cần contract. Và bạn sẽ thấy ngay sau đây, một dịch vụ trên nền ASP.NET Web API được thiết lập lỏng lẻo nếu ta so sánh với dịch vụ WCF. Theo đó bạn chỉ cần sử dụng REST ,URL, cùng các tham số đầu vào, và kết quả trả về là các bản tin JSON hay XML.

* Mặc định sử dụng REST

Việc xây dựng các dịch vụ với ASP.NET Web API cung cấp cho ta hầu hết những thành phần thiết yếu để xây dựng kiến trúc REST. Đó là bởi vì tính năng điều hướng (Route) URL tích hợp ngay trong nền tảng, các địa chỉ dịch vụ với ASP.NET Web API là các điều hướng RESTful ánh xạ với các phương thức của controller. Vì thế cho phép sử dụng các đường dẫn theo phong cách REST rất thân thiện.

* Trìu tượng hóa với các Route

Tương tự như các giao diện dịch vụ và sự thực thi chúng trên WCF, các route cho phép nhà phát triển ASP.NET Web API tạo ra lớp trìu tương giữa bên yêu cầu nhìn thấy và sự thực thi ở lớp dưới. Nói theo cách khác thì bạn có thể gắn kết bất cứ URL nào với bất cứ phương thức nào của các controller.  
Với sự thay đổi về phiên bản dịch vụ, chỉ đơn giản bằng cách thêm một route mới (ví dụ “v2”) vào trong URL cho phép ta duy trì song song nhiều phiên bản dịch vụ trên hệ thống mà không ảnh hưởng đến các dị vụ đang có.

##### Mở rộng chức năng xử lý dạng đường ống đơn giản

ASP.NET Web API cung cấp khả năng mở rộng dạng đường ống thông qua các bộ lọc (filter) cho phép xử lý trước và sau khi thực thi các action của controller.  
Bằng việc gắn các bộ lọc cho phép ta thực hiện lọc các action (theo tham số request), lọc vai trò (authorization) và lọc ngoại lệ  
Để dùng có thể đơn giản bằng cách khai báo attribute, hay cấu hình toàn cục áp dụng cho tất cả các phương thức.

2.2.3 **Một số đặc điểm nổi trội của ASP.NET Web API**

##### Thuận tiện cho các hành động (action) CRUD (create – read – update – delete)

* Các action HTTP (như GET, POST) tự động gắn kết vào các phương thức của controller (chính là các controller action) thông qua tên của chúng
* Thỏa thuận nội dung được tích hợp sẵn
* Với MVC, để cho phép các phương thức controller trả về dữ liệu JSON hay XML thì phải tùy biến kết quả trả về ở dạng nguyên gốc (raw). Nhưng với ASP.NET Web API, các phương thức controller chỉ cần trả về dữ liệu gốc, toàn bộ phần còn lại sẽ tự động được chuyển đổi sang JSON hay XML tùy theo yêu cầu từ gọi tới. Trong đó, yêu cầu từ bên gọi tới cũng chỉ cần sử dụng tiêu đề (header) Accept hay Content-Type

##### Cho phép sử dụng các attribute tiền tố điều hướng (route prefix) và route

* Với MVC, để cho phép các phương thức controller trả về dữ liệu JSON hay XML thì phải tùy biến kết quả trả về ở dạng nguyên gốc (raw). Nhưng với ASP.NET Web API, các phương thức controller chỉ cần trả về dữ liệu gốc, toàn bộ phần còn lại sẽ tự động được chuyển đổi sang JSON hay XML tùy theo yêu cầu từ gọi tới. Trong đó, yêu cầu từ bên gọi tới cũng chỉ cần sử dụng tiêu đề (header) Accept hay Content-Type

##### Quản lý lỗi toàn cục

* Đây là một cải thiện rất lớn từ phiên bản ASP.NET Web API 2.1. Theo đó, tất cả các exception không được bắt giờ có thể được ném ra và bắt thông qua một cơ chế tập trung. Việc hỗ trợ nhiều các ghi log ngoại lệ được thực hiện trong chính nền tảng (framework), tại đó cho phép ta truy xuất đến chính ngoại lệ bị ném ra cùng ngữ cảnh của nó.

##### Giao diện IhttpActionResult

* Việc thực thi giao diện này sẽ định nghĩa một HttpResponseMessage factory có thể tái sử dụng và thuận tiện khi xây dựng unit-test.

**2.2. ASP.NET Web API 2.2**

2.2.4. **Entity Framework Code First**

Entity Framework Code First được giới thiệu từ Entity Framework 4.1. Code First chủ yếu là hữu ích trong [**Domain Driven Design**](http://msdn.microsoft.com/en-us/magazine/dd419654.aspx)**(DDD)**. Trong cách tiếp cận Code First, ta có thể tập trung vào việc thiết kế **Domain**và bắt đầu tạo ra các lớp theo yêu cầu của Domain chứ không phải thiết kế cơ sở dữ liệu trước rồi sau đó tạo ra các lớp phù hợp với thiết kế cơ sở dữ liệu đó. Code First API sẽ tạo ra cơ sở dữ liệu dựa trên các lớp thực thể và lớp cấu hình của bạn.

Vì vậy, đầu tiên bạn bắt đầu viết các lớp thay vì tập trung vào thiết kế cơ sở dữ liệu, sau đó khi bạn chạy ứng dụng, Code First API sẽ tạo ra cơ sở dữ liệu mới hoặc ánh xạ các lớp của bạn vào cơ sở dữ liệu đã tồn tại trước khi chạy ứng dụng của bạn.

Vì vậy, các công việc cơ bản sẽ là viết các lớp  → Nhấn F5 để chạy ứng dụng → Code First API tạo ra cơ sở dữ liệu mới hoặc ánh xạ các lớp với cơ sở dữ liệu đã tồn tại → Thêm dữ liệu mặc định vào cơ sở dữ liệu hoặc kiểm tra dữ liệu trong cơ sở dữ liệu → Cuối cùng khởi chạy ứng dụng.

Ví dụ Entity Framework Code First

Tạo 1 lớp Student

public class Student

{

public Student()

{ }

public int StudentId { get; set; }

public string StudentName { get; set; }

......

}

Lớp Standard

public class Standard

{

public Standard()

{ }

public int StandardId { get; set; }

public string StandardName { get; set; }

public string Description { get; set; }

}

Bây giờ ta sẽ tạo lớp context kế thừa từ lớp DBContext với 2 thuộc tính DbSet, một cho lớp Student và một cho lớp Standard.

namespace SchoolDataLayer

{

public class Context: DbContext

{

public Context(): base()

{ }

public DbSet<Student> Students { get; set; }

public DbSet<Standard> Standards { get; set; }

}

}

Trong hàm main, bạn viết đoạn mã sau:

using (var ctx = new Context())

{

Student stud = new Student() { StudentName = "New Student" };

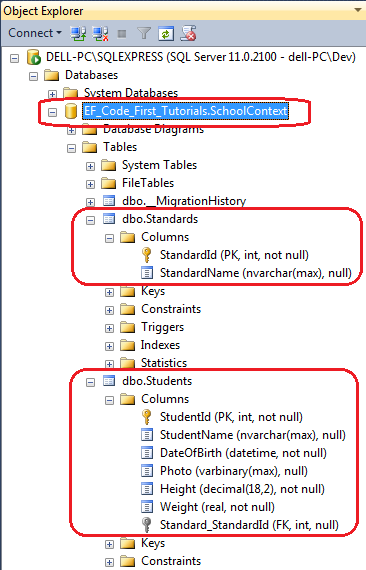
ctx.Students.Add(stud);

ctx.SaveChanges();

}

Nhấn F5 để chạy chương trình, Code First API của Entity Framework sẽ tạo database. Database này được tạo trong SQLEXPRESS.

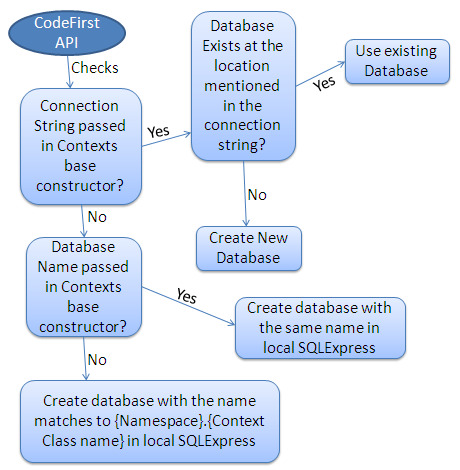
Code First API cũng sẽ tạo 2 bảng trong database này là Students và Standards. Hai bảng này được tạo dựa trên 2 lớp Student và Standard mà chúng ta đã tạo ở trên. Code First API sẽ tạo khóa chính cho bảng nếu trong lớp có bất kỳ thuộc tính nào có tên chứa chữ "Id". Ví dụ lớp Student có thuộc tính StudentId sẽ được sử dụng để tạo khóa chính trong bảng Students. Nó cũng sẽ tạo các cột có tên và kiểu dữ liệu giống với các thuộc tính của lớp.



Với cách thiết kế này bạn sẽ không tạo database trước, thay vào đó bạn sẽ viết các lớp trước, Code First API sẽ tạo database cho bạn.

* Làm thế nào Code First API tạo ra được database?

Hình dưới đây thể hiện luồng khởi tạo database dựa trên đối số truyền vào phương thức khởi tạo của lớp context kế thừa từ lớp DbContext:



Như hình trên, bạn có thể truyền các đối số vào phương thức khởi tạo của lớp context như sau:

**Không có đối số truyền vào:** Nếu bạn không truyền đối số cho phương thức khởi tạo thì Code First API sẽ tạo database của bạn trong SQLEXPRESS với tên được đặt theo quy tắc sau {Namespace}.{Tên\_lớp\_Context}. Ví dụ: database sẽ có tên là SchoolDataLayer.Context nếu chúng ta định nghĩa lớp Context như sau:

public class Context: DbContext

{

public Context(): base()

{

}

}

**Truyền vào tên của database:** Nếu bạn truyền tên của database vào đối số của phương thức khởi tạo, Code First API sẽ tạo database có tên như vậy trong SQLEXPRESS. Ví dụ: lớp Context dưới đây sẽ tạo database tên là "SchoolDB-CodeFirst" trong SQLEXPRESS.

public class Context: DbContext

{

public Context(): base("SchoolDB-CodeFirst")

{

}

}

**Truyền vào tên của database:** Nếu bạn truyền tên của database vào đối số của phương thức khởi tạo, Code First API sẽ tạo database có tên như vậy trong SQLEXPRESS. Ví dụ: lớp Context dưới đây sẽ tạo database tên là "SchoolDB-CodeFirst" trong SQLEXPRESS.

public class Context: DbContext

{

public Context(): base("SchoolDB-CodeFirst")

{

}

}

**Truyền vào chuỗi kết nối:** Nếu bạn truyền vào tên của chuỗi kết nối được định nghĩa trong file app.config hoặc web.config thì Code First API sẽ tạo database dựa trên tên và máy chủ được định nghĩa trong chuỗi kết nối. Ví dụ: lớp Context dưới đây sẽ tạo database tên là "SchoolDB-ByConnectionString" đặt trên máy chủ 192.168.1.2:

public class Context: DbContext

{

public SchoolDBContext() : base("SchoolDBConnectionString")

{

}

}

App.config:

xml version="1.0" encoding="utf-8" ?>

<configuration>

<connectionStrings>

<add name="SchoolDBConnectionString"

connectionString="Data Source=192.168.1.2;Initial Catalog=SchoolDB-ByConnectionString;Integrated Security=true"

providerName="System.Data.SqlClient"/>

</connectionStrings>

</configuration>

2.2.5 **Respository và unit of work trong ASP.NET Web API.**

**Repository** là lớp trung gian giữa lớp data và lớp business. Trong project sử dụng Entity Framework với ASP.NET MVC thì Data chính là tầng chứa các lớp dbContext và class enttiy. Còn business logic chính là tầng xử lý nghiệp vụ của dự án. Repository hầu hết được sử dụng những chỗ cần điều chỉnh dữ liệu trước khi truyền xuống tầng data hoặc truyền lên trên business logic.

## Lợi ích của việc sử dụng Repository Pattern

* Tập trung hóa được các logic về xử lý dữ liệu hoặc business logic.
* Dễ dàng unit test
* Đưa ra kiến trúc linh hoạt
* Khi thay đổi logic của tầng data hoặc business logic, không cần thay đổi Repository

## Generic Repository Pattern

Trong các trường hợp khác, dự án lại đòi hỏi một cách tốt nhất để tạo ra tất cả các repository logic ở cùng một nơi. Chúng ta cần tạo 1 và chỉ 1 repository cho việc thao tác với toàn bộ các class entity. Vậy để giải quyết điều này chúng ta phải sử dụng Generic Repository Pattern.

## Lợi ích của Generic Repository Pattern:

* Giảm thiểu sự trùng lặp code
* Đảm bảo các coder dùng chung 1 pattern
* Ít lỗi hơn
* Dễ dàng bảo trì sau này

# Unit Of Work trong Repository Pattern

Unit Of Work được sử dụng để đảm bảo nhiều hành động như insert, update, delete...được thực thi trong cùng một transaction thống nhất. Nói đơn giản hơn, nghĩa là khi một hành động của người dùng tác động vào hệ thống, tất cả các hành động như insert, update, delete...phải thực hiện xong thì mới gọi là một transaction thành công. Gói tất cả các hành động đơn lẻ vào một transaction để đảm bảo tính toàn vẹn dữ liệu.

2.2.6. **Bảo mật trong ASP.NET Web API**

* Token based:

Token based authentication là cách xác thực nổi bật ở khắp mọi nơi trên các web hiện

nay. Với hầu hết các công ty web sử dụng 1 API, các token là cách tốt nhất để xử lý

xác thực cho người dùng. Có một số yếu tố rất quan trọng khi lựa chọn token based authentication cho ứng dụng của bạn. Đó là:

* Phi trạng thái và khả năng mở rộng server.
* Ứng dụng di động
* Nhận xác thực cho các ứng dụng khác
* Tăng bảo mật
* Token based hoạt động thế nào:

Token based authentication là phi trạng thái (stateless). Chúng tôi không lưu giữ

bất kỳ thông tin và người sử dụng của chúng tôi trên server hoặc trong một

session.

Khái niệm này một mình nó đã giải quyết rất nhiều vấn đề với việc phải lưu trữ

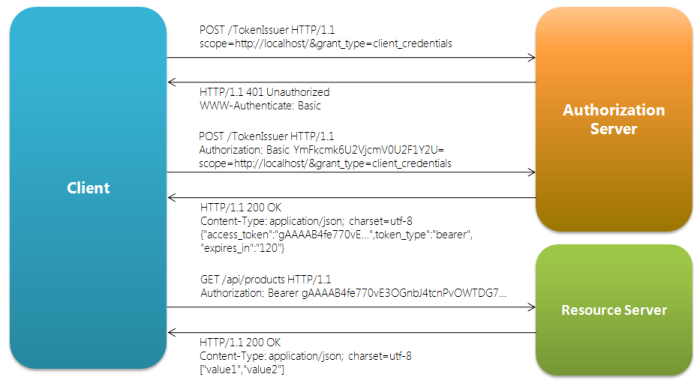
thông tin trên máy chủ. Không có thông tin về session nghĩa là ứng dụng của bạn có thể mở rộng và thêm nhiều server khi cần thiết mà không cần lo lắng về việc người dùng đăng nhập.

Những ý chính của nó như sau:

* Người dùng yêu cầu truy cập với username và password.
* Chương trình xác nhận thông tin đăng nhập.
* Chương trình cung cấp một token gửi cho khách hàng.
* Client sẽ lưu trữ cái token này và sẽ gửi nó kèm theo mỗi yêu cầu.
* Server xác minh token và trả về dữ liệu.

Mỗi một request sẽ yêu cầu token đi kèm. Token sẽ được gửi vào các HTTP

header để tiếp tục ý tưởng HTTP phi trạng thái.



**JSON Web Tokens:**

* JSON Web Tokens (JWT) là một tiêu chuẩn an toàn kể từ khi

thông tin được truyền qua JSON, những claims trong JWT được mã hóa như một đối tượng JSON và được ký điện tử sử dụng JSON Web Signature.

* JWT làm việc trên nhiều ngôn ngữ khác nhau: .NET, Python, Java, PHP, Ruby, v..v…
* JWT được khép kín: nó sẽ mang theo tất cả những thông tin cần thiết trong chính

nó. Có nghĩa là một JWT sẽ có khả năng truyền các thông tin về bản thân, payload (thường là thông tin người dùng) và chữ ký.

* JWT có thể được truyền đi một cách dễ dàng: Do JWT khép kín, hoàn toàn có thể

sử dụng nó trong một tiêu đề HTTP khi xác thực một API, hoặc có thể truyền nó

thông qua URL.

* Cấu trúc của 1 JWT

JWT rất dễ xác định, nó là một chuỗi mã hóa gồm 3 chuỗi ngăn cách nhau bởi dấu “.”

Ví dụ:

Aaaaaaaaaaaa.bbbbbbbbbbbbbbb.ccccccccccccccc

Ba phần này là header, payload và signature.

* Header

Phần header chứa 2 phần:

* + - Khai báo kiểu, nó là JWT.
    - Thuật toán băm được sử dụng.

Ví dụ:

{

"alg": "HS256",

"typ": "JWT"

}

Giờ ta sẽ mã hóa nó bằng thuật toán Base64, chúng ta có phần đầu tiên của JWT như sau:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9

* Payload

Phần Payload sẽ mang một lượng lớn trong JWT, cũng có thể gọi nó là JWT Claims. Đây là nơi đưa những thông tin muốn truyền tải và các thông tin khác về token.

Có nhiều claim mà chúng tôi có thể cung cấp. Bao gồm register claim names, public claim names, và private claim names.

* + Register Claim names:

Không bắt buộc phải có đầy đủ. Phần này bao gồm:

* + - iss: Tổ chức phát hành token.
    - sub: Tên người sử dụng token
    - exp: đây là registed claim thường được sử dụng nhất. Nó xác định hạn sử dụng của JWT.
* Signature

là chữ ký được ký bởi server. Đây là cách mà máy chủ có thể xác minh token hiện tại và ký những cái mới.

Đây là phần cuối trong JWT:

bxZ7DaOL5N6WwUnQR5jw7tkwbAGQ4sujgBy-z6xqiFE

* Và giờ chúng ta sẽ có một JSON Web Token đầy đủ:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJqd3QuaW8iLCJleHAiOiIyNTA1MjAxNiIsIm5hbWUiOiJhZG1pbiIsImFkbWluIjp0cnVlfQ.bxZ7DaOL5N6WwUnQR5jw7tkwbAGQ4sujgBy-z6xqiFE

Tiêu chuẩn JSON Web Token có thể được sử dụng trên nhiều ngôn ngữ và có thể hoán đổi cho nhau một cách dễ dàng, nhanh chóng.

Có thể sử dụng JSON Web Token trong một URL, tham số POST hoặc một tiêu đề HTTP. Sự linh hoạt của JSON Web Token cho chúng ta xác thực API một cách nhanh chóng và dễ dàng bằng cách truyền thông tin thông qua các token.

**CHƯƠNG IV: THỰC HÀNH – PHÂN TÍCH HỆ THỐNG**

* 1. **Phân tích yêu cầu**

Tên đề tài

“Xây dựng website trao đổi mua bán hàng online”

PHẦN 1: CHỨC NĂNG

* + Đây là website cho phép các mọi người (có tài khoản của website) có thể đăng bán những mặt hàng mà họ muốn bán.
  + Cho phép người dùng đăng lên những nhu cầu muốn mua hoặc tuyển dụng lên website, người khác có mặt hàng muốn bán hoặc nhu cầu mà phù hợp với nhu cầu đó có thể liên hệ với người đăng để thỏa thuận.
  + Cho phép người có nhu cầu có thể lên website tìm kiếm tin mà họ muốn theo địa chỉ, các loại tin, giá cả phù hợp với họ.
  + Lưu thông tin của người dùng, tất cả các tin được người dùng đăng lên.
  + Hiển thị danh sách tin của người dùng đăng lên theo thứ tự thời gian đăng (mới đăng sẽ được đẩy lên đầu trang).
  + Cho phép mọi người quan tâm đến tin nào đó có thể gửi tin nhắn cho người đăng tin đó bằng mail, xem số điện thoại người bán để liên lạc.
  + Cho phép người dùng có tài khoản cập nhật thông tin cá nhân.
  + Người dùng có thề quản lý các tin mà họ đăng.
  + Người dùng có tài khoản có thể comment vào 1 chi tiết của 1 tin bất kỳ.
  + Thông báo về cho người đăng nếu tin của họ có người dùng khác tác động vào (người khác comment vào tin mà họ đăng, admin thông báo về tin mà họ đăng, có người gửi mail cho họ, có ai đó nhắn tin cho họ) .
  + Đẩy tin liên đầu trang liên tục cứ sau 1 khoảng thời gian nhất định.

PHẦN 2: YÊU CẦU HỆ THỐNG WEBSITE

Hệ thống website gồm 2 phần:

1. Phần cho người quản trị (Admin).

Ngưởi quản trị là người có toàn quyền trên website. Website phân biệt người quản trị login vào hệ thống website bằng Role của của user đăng nhập thành công. Khi admin đăng nhập thành công thì vào trang giao diện của người quản trị.

Các quyền của người quản trị:

* Xem và duyệt các tin do người dùng đăng lên.
* Quản lí thông tin của admin.
* Thống kê các loại tin được đăng nhiều nhất.
* Xem tổng số tin được đăng lên, tổng số user.
* Thống kê số tin theo từng loại tin, thống kê tin theo địa chỉ đăng tin.
* Gửi thông báo về cho người đăng nếu tin của họ không hợp lệ.

1. Phần danh cho người sử dụng

Đối với người sử dụng bất kỳ:

* Hiển thị danh sách tin được sắp xếp theo thời gian mỗi khi người sử dụng vào trang web.
* Người dùng tìm kiếm tin theo loại tin, tiêu đề tin và nơi đăng tin.
* Xem danh sách các nhu cầu mua, tuyển dụng.
* Đăng ký tài khoản
* Gửi tin nhắn cho người đăng tin

Đối với người sử dụng có tài khoản

* Đăng tin
* Quản lý các tin đã đăng
* Xem các thông báo
* Comment vào tin bất kỳ
* Xem danh sách tin

1. Sơ đồ phân cấp chức năng mức dưới đỉnh

**Người sử dụng**

Quản lý các tin đã đăng

Đăng nhu cầu mua, thuyển dụng

Xem danh sách tin

Xem chi tiết tin

Đăng ký tài khoản

Tìm kiếm

Đăng tin

Bình luận

Nhắn tin cho người đăng tin

Xem các tin liên quan khác

Tìm theo loại tin

Tìm theo địa chỉ đăng

* Đối với người sử dụng
* Đối với người quản trị

**Người quản trị**

Thống kê

Duyệt tin

Quản lý tài khoản

Quản lý tin mà người quản trị đã chp phép lên trang chủ

Chấp nhận cho phép tin lên trang chủ

Không chấp nhận tin, gửi thông báo về cho người đăng

1. Sơ đồ ERD

C:\Users\thanh\Desktop\Untitled Diagram.png