|  |
| --- |
| **Pre-Thesis** |

###### **Topic: Angular 2 getting started**



*Prepared by Đỗ Hùng Cường*

June 18, 2017

Table of Contents

**Introduction………………………………………………………………………………….……………. 2**

**Content…………………………………………………………………………****……………………….…… 2**

**1. What is Angular 2 ? ………………………………………………………………………… 2**

**2. Why using Angular 2 ? ……………………………………………………………………. 3**

**3.** **Architecture of Angular 2.…….…………………………………………………………. 5**

**4.** **Using RESTful Web Service with Angular 2………………………………………. 16**

**5.** **Securing RESTful Web Services with OAuth 2.0…………….…………………. 24**

**6.** **Install and setup environment.…….…………………………………………………. 27**

**7.** **Source code and Demo.…….………………………………………………………....... 31**

**Conclusion.…………………………………………………………………………………….............. 47**

**References.………………****…………………………………………………………………….............. 47**

1. [*Introduction*](http://www.enteract.com/~bradapp/docs/sdd.html#TOC_SEC4)

Over the last 4 years, Angular has turned into the main open source JavaScript application framework. A huge number of developers all over the world begin to learn Angular and the Angular version 1.x has been widely used and became well known for website developers to build the complex applications. Then, Angular 2.x has been released to improve and overcome the weaknesses of its previous version.

This document is written for people who want to learn about Angular 2. Thank to this report, the beginners will know what Angular 2 is, why they should use Angular 2 and its architecture. Beside, this report also introduces a little bit about RESTful Web Service and Open Authentication 2, which relate to an Angular 2 application.

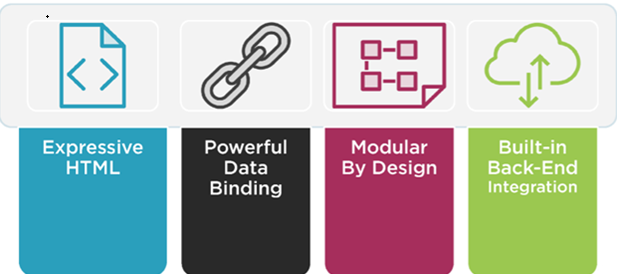
1. *Content*
2. What is angular 2?

Angular 2 is the new improve version of Google’s massively popular MV\* framework, it is actually an open source JavaScript framework that will help you to build a complex web application or mobile application in html with css, javascript or typescript. It is the new version that improves the JavaScript framework Angular 1 or AngularJS.

1. Why using angular 2?

There are many reason why we need to use Angular. The most important reason is that Angular allows us to build a single page application easily. Single Page Application is a web application that on only one single web page or only one index page contains dynamic actions which we does not need to refresh the page. Single Page Application interactions can be handle without reaching server. Single Page Application can improve performance in many cases such as loading time, using AJAX, easy to navigate pages etc. That makes the end users feel more comfortable when using Single Page Application. Recently, many frameworks, platforms or techniques were released to support building a Single Page Application. Angular is one of the most popular Single Page Application framework.

Another reason for using Angular is its features:



* Angular makes the HTML more expressive by support some features such as if-else condition, switch-case, loop and local variable.
* Angular has powerful data binding. Thank to data binding, we can easily display variables from the data model such as component, track changes, and process updates from the user.
* Angular promotes modularity by design. Every Angular applications is a set of building blocks and that is easier to create and reuse content.
* Angular has built-in support for communication with a back-end service. In Angular application, it is easy for the fontend to integrate with a backend service to get and post data or execute server-side business logic.

With so many developers already using Angular 1 or AngularJS, why do we need Angular 2?

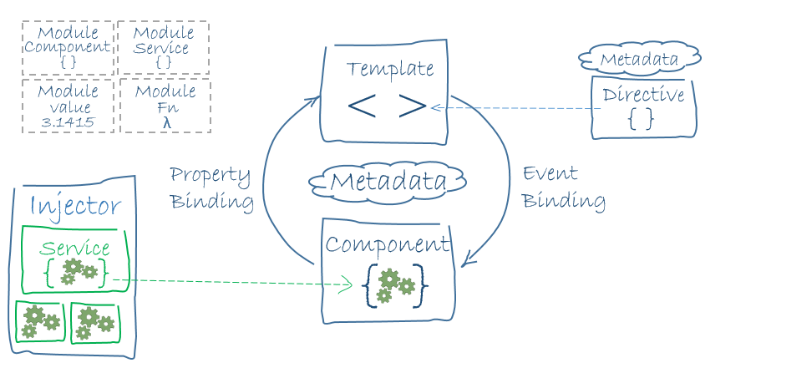
* Angular 2 is faster than Angular 1. Angular 2 is built for speed so that the initial loads is faster, the change detection and improved rendering times are also faster than Angular 1.
* The fewer concepts of Angular 2 make it easier to understand than Angular 1. The code structure is very simplified than the previous version of Angular. Therefore, it is easier for the developer to learn and use Angular 2.
* In Angular 2, we can use Typescript which is a super set of Javascript. Typescript is a form of JavaScript, in Typescript we can know types and classes. Typescript can be compiled to JavaScript. TypeScript is an open source that contains many aspects of object orientation such as interfaces and inheritance. The TypeScript’s syntax is cleaner than javascript and similar to C# or java. Because of using TypeScript, so we can use all its libraries and the functionality of TypeScript itself in Angular 2.



* Angular 2 is mainly focused on mobile apps. Angular 1.x was not built with mobile support in mind, where Angular 2 is mobile oriented.

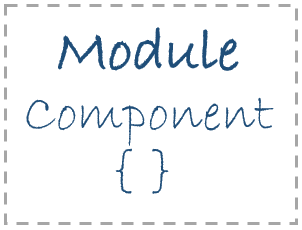
1. Architecture of angular 2 application

The architecture of angular 2 focus on the big picture below:



There are 8 main building blocks of an Angular application in this architecture diagram: Modules, Components, Templates, Metadata, Data binding, Directives, Services and Dependency injection.

Now let’s focus on each block.

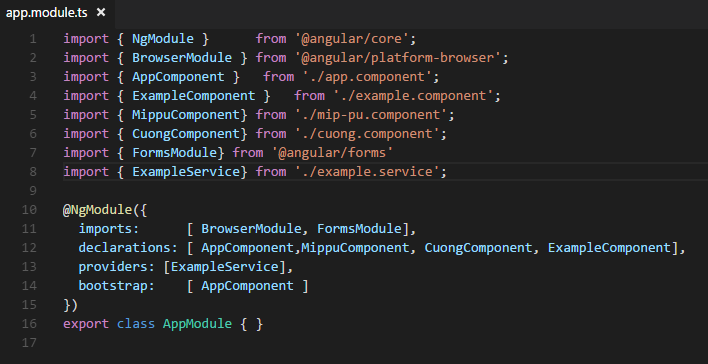
Modules

In each Angular application, there must be one or more Angular module class. The [root module](https://angular.io/docs/ts/latest/guide/appmodule.html) is always available in every Angular app.

In a small applications, they may have only one root module. However, almost larger applications may have the root module and many feature modules.

An Angular module is a class with the @NgModule decorator. Decorators are functions that modify JavaScript classes which attach metadata to those classes so that Angular can knows what they mean and how they should work.

Here's a simple root module:



NgModule is a decorator function that describes the module by taking a single metadata object. The properties of this metadata object will show what module control:

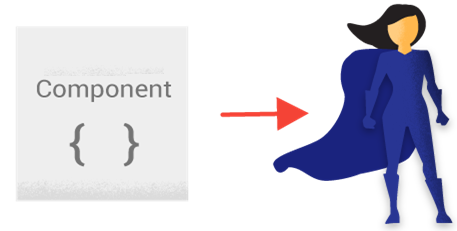
* declarations - the view classes that were controlled by this module.
* exports - the subset of declarations that the component templates of other modules can import to use.
* imports - other modules that were exported and we need to use them in the component templates declared in this module.
* providers - creators of services that this module contributes to the global collection of services. We can access it in all parts of the Angular application.
* bootstrap - the main application view, called the root component, that hosts all the views of the Angular application. We should bootstrap only a root module.

There are no reason to export the root module class (AppModule) because we do not need to import the root module in other components.

The Angular application can run by bootstrapping its root module (bootstrap the AppModule in a main.ts file like this source code below)



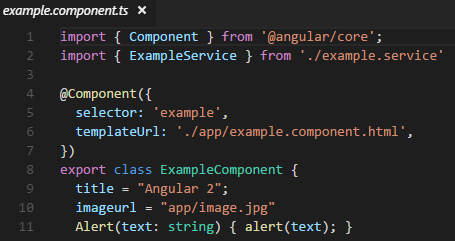
Components



A component controls the view. For instance, some views of a page are controlled by components such as the header, the footer, the sidebar of a page.

We define the application logic in class of the component. That can be fields or functions which support the view. The class and the view interacts with each other through an API of properties and methods.

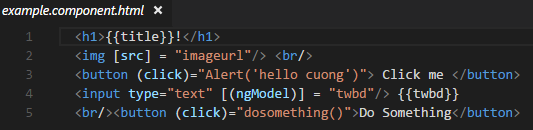
This picture below describes the source code of a component called ExampleComponent.



Templates

We define the view of a component by the **template**. We can write html code or put html code in the html file to build the template that tells Angular how to render the component.

A template is similar to HTML. However, they are different a little bit. Here is the source code of a template. It is actually the view of the ExampleComponent above:



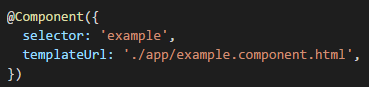
We can recognize that this template uses typical HTML elements like <h1>, <input> and <button>, it is just a html tag that any web developer has known. However, the codes like (click) = "Alert ('hello cuong')" and [(ngModel)] = "twbd" are not html codes anymore. That is the syntax of Angular’s template.

MetadataMetadata

Metadata tells Angular how to process a class.

The ExampleComponent is a normal class until we tell Angular that ExampleComponent  is a component. To do that job, we have to attach **metadata** to this class. In TypeScript, we attach metadata by using a **decorator**.

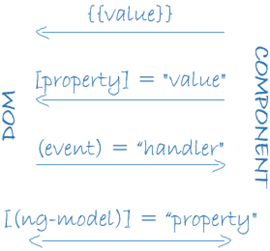
Here's some the source code of metadata that make the class ExampleComponent become a component.



Data binding

Data binding is one of the most powerful technique of Angular. For example, when we build a simple calculator on View, if using Javascript and jQuery, then we have to select each corresponding input tag, then retrieve the value in the tag, and change the data type from String to number, and calculate, then re-select the tag to display the results in the View. With Angular, we do not need to do so, just simply attach the variable in the logic code of the component to the template and everything will display automatically without having to be selective like the classical way.

Angular supports **data binding** for coordinating parts of a component with parts of a template. Data binding is added to the template HTML that tell Angular how to connect both sides.



There are 4 kinds of data binding in the diagram above.

* Interpolation Binding: This way helps you to display the fields, the variable,the value from component or another data model within the tags.



* Property Binding: This way helps you to pass the properties from the parent to child‘s properties.



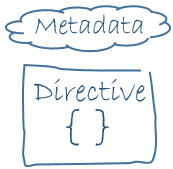
* Event Binding: This way helps you to catch the event and handle when clicking on the components method name.



* Two-way Binding: This way uses the ngModel directive in a single notation to help you to bind both property and event.



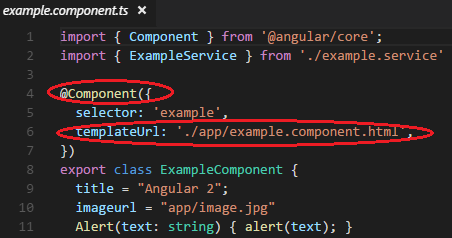
Directives



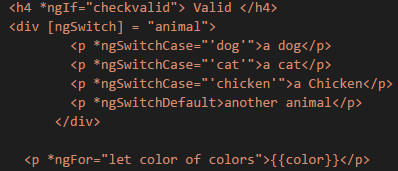
The directive is also a class, in directive, we will attach the metadata by the @Directive decorator.

**There are 3 kinds of directive: Component directive, Structural** directive and **Attribute** directive.

**Component is a directive with @Component decorator and the view (template). The @Component decorator is actually a @Directive decorator and it is extended wherein the template-oriented features.**

****

**Structural** directives add, remove or replace elements in DOM to change the layout. Some common examples of **Structural** directives are ngIf, ngSwitch (Switch-Case), ngFor.



**Attribute** directives are similar to regular HTML attributes in templates. They changes the behavior or appearance of a DOM element. One of the common example of **Attribute** directives is two way binding.



Services



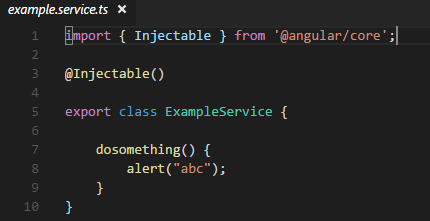
Service is a class, we use service as a category contains any feature, function ,value or what we need to use in our application.

Almost anything in Angular can be a service. We should define service for clearly purpose. Services should have a specific work and do it well.

Angular does not define service. However, services are fundamental to any Angular application. It provides function and feature for the components and so much more.

We should keep component classes clean and provide primary job for them like the medial controlers between the view which rendered by the template and the application logic or presents properties and methods for data binding. Therefore, we should delegates another job to the services.

Here's an example of a service class:



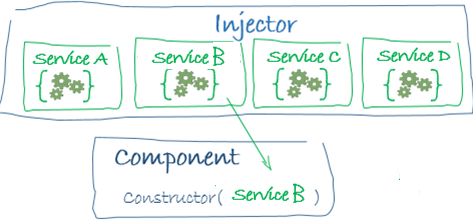
Dependency injection



In Angular, we provide new components with the services they need by using dependency injection. Therefore, Angular can tell the components that the types of their constructor parameters may include the services they need.

When created a component, Angular firstly asks an **injector** for the services, that is the parameters in the contructor of the component.

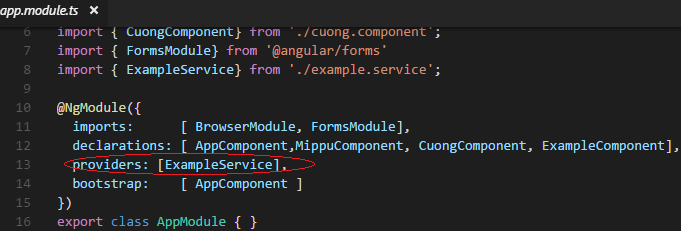
In the picture below, we will see Angular injects ServiceB to a Component. That means Angular tell that the service which the component need is ServiceB.



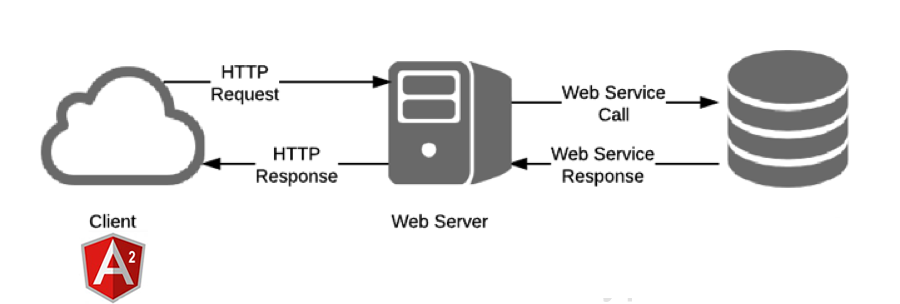
However, in the injector box above, if we do not have a ServiceB, how does Angular know how to make one?

In that case, we have to register a **provider** of the ServiceB with the injector.

We can register providers in components or modules. To create the same instance of a service everywhere in our application, we can add providers to the root module. Here is the source code that provider is added to the root module



1. Using RESTful Web Service with Angular 2



Angular 2 is simply a front-end framework for building applications. It is not the right determinant for what backend you should use for your application. There are many ways to connect angular 2 to your backend server. RESTful Web Service which is essentially REST Architecture based Web Services is one of the architectural style that helps angular 2 and your backend server communicate with each other. In REST Architecture, everything is a resource. RESTful web services are light weight services so the developers usually use RESTful web services to make APIs for web-based applications.

In some case, RESTful Web Service help us write a software application in various programming languages and we can run them on various platforms. For example, we can write a backend server in Java using RESTful web service and connect to fontend in Angular 2 using Typescript.

REST is a web standards based architecture which was first presented by Roy Fielding in 2000. The word ‘REST’ means REpresentational State Transfer. REST uses HTTP Protocol for data communication. It spins around resources where each component is a resource and a resource accessed by a typical interface utilizing HTTP standard methods.

RESTful Web Services are Web services based on REST Architecture. They use HTTP methods to implement the concept of REST architecture. URI is usually a service which a RESTful Web Service provides resource such as Text, JSON and XML.

* There are 5 HTTP methods are most commonly used in a REST based architecture.

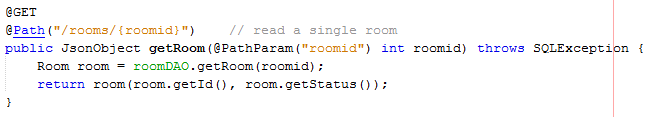
| **HTTP Verb** | **CRUD** | **Entire Collection (e.g. /rooms)** | **Specific Item (e.g. /rooms/{id})** |
| --- | --- | --- | --- |
| POST | Create | 201 (Created), 'Location' header with link to /rooms/{id} containing new ID. | 404 (Not Found), 409 (Conflict) if resource already exists. |
| GET | Read | 200 (OK), list of rooms. Use pagination, sorting and filtering to navigate big lists. | 200 (OK), single room. 404 (Not Found), if ID not found or invalid. |
| PUT | Update/Replace | 405 (Method Not Allowed), unless you want to update/replace every resource in the entire collection. | 200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid. |
| PATCH | Update/Modify | 405 (Method Not Allowed), unless you want to modify the collection itself. | 200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid. |
| DELETE | Delete | 405 (Method Not Allowed), unless you want to delete the whole collection—not often desirable. | 200 (OK). 404 (Not Found), if ID not found or invalid. |

\* For example: The source code that creates a Web service which can perform CRUD (Create, Read, Update, Delete) operations

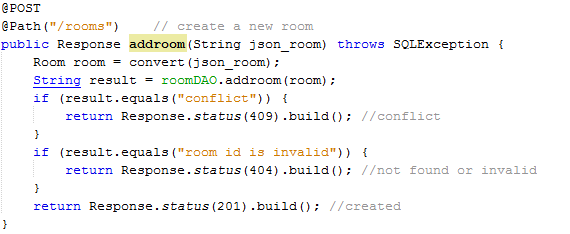
@GET a list of rooms



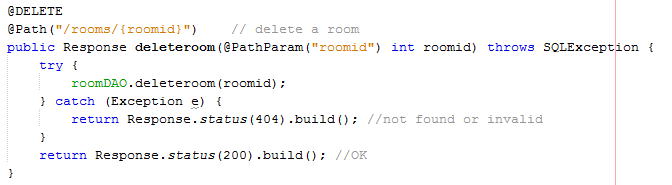
@GET a single room



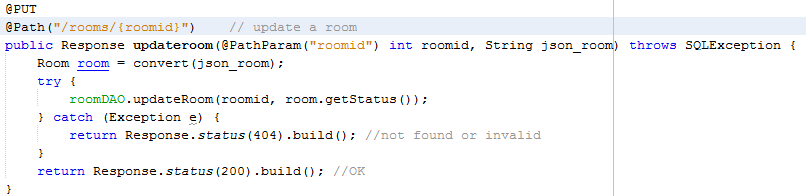
@POST a new room



@DELETE a room



@PUT a room



The method @GET with @Path(“/rooms”) will return a list of rooms in Json array to the URL: ‘http://localhost:8080/HotelManagementSystem/resources/api/RoomService/rooms’

The method @GET with @Path(“/rooms/{roomid}”) will return a single room in Json to the URL: ‘http://localhost:8080/HotelManagementSystem/resources/api/RoomService/rooms/101’ with ‘101’ is the id of the room (‘101’ can be change to other id of another room)

The method @POST will create a new room in Json to the URL: ‘http://localhost:8080/HotelManagementSystem/resources/api/RoomService/rooms’

The method @DELETE will delete a single room, exactly a room store in this this URL: ‘http://localhost:8080/HotelManagementSystem/resources/api/RoomService/rooms/101’ with ‘101’ is the id of the room (‘101’ can be change to other id of another room)

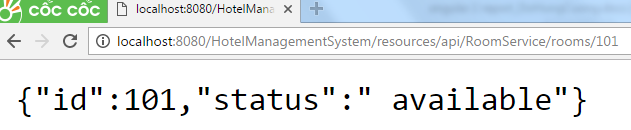
The method @PUT will update, change the information of a single room, exactly a room store in this this URL: ‘http://localhost:8080/HotelManagementSystem/resources/api/RoomService/rooms/101’ with ‘101’ is the id of the room (‘101’ can be change to other id of another room)

Here is the result link of this resource:

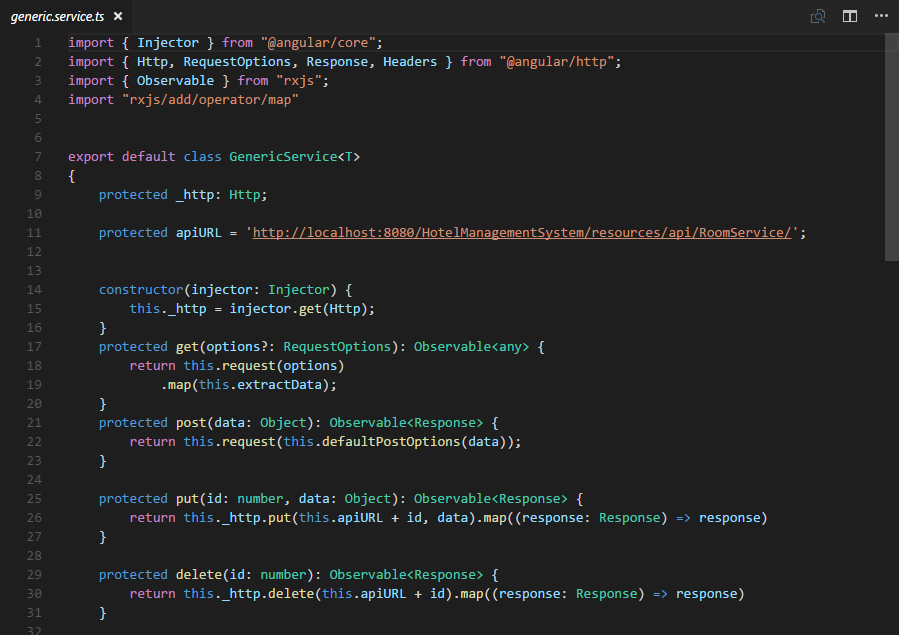
http://localhost:8080/HotelManagementSystem/resources/api/RoomService/rooms



You can get a single room by adding an id of a room to this URL, for example: http://localhost:8080/HotelManagementSystem/resources/api/RoomService/rooms/101  
(with 101 is room id)

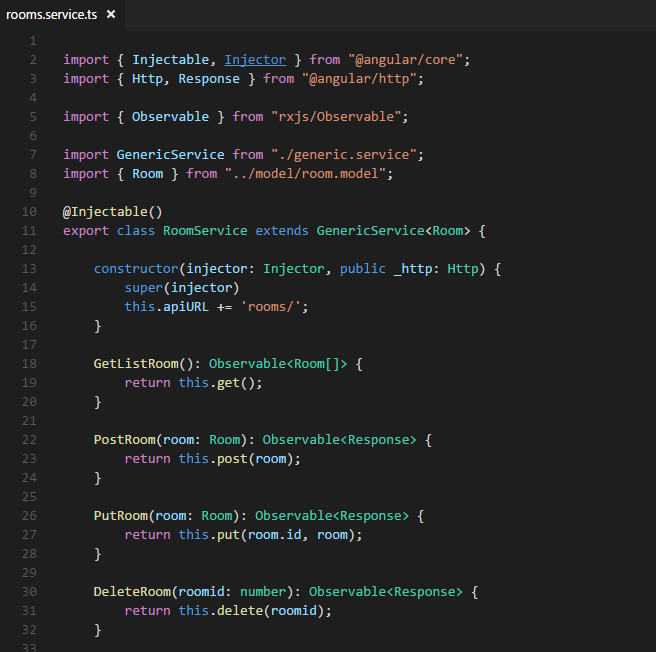


You can the use method GET/PUT/POST/DELETE from library @angular/http. Then using the api URL link that you had created from java RESTful Web Service to connect angular 2 with your backend server. Here is the service that import the library @angular/http and provides some http methods GET/PUT/POST/DELETE.



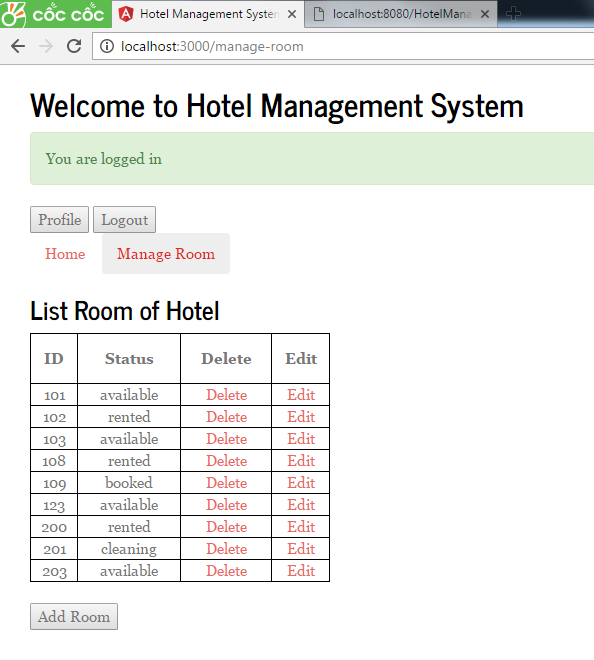
You can write a RoomService extends the service above to use those http methods GET/PUT/POST/DELETE for any purpose that you want to do with your resources.

For example (view list room, update room, add room or delete room)



Here is the result of the website write in Angular 2 with Java Restful Web Service.

You can see the list of rooms by method @GET, and when you want to add room, edit room or delete a room you will use in order these methods @POST, @PUT, @DELETE



1. Securing RESTful Web Services with OAuth 2.0

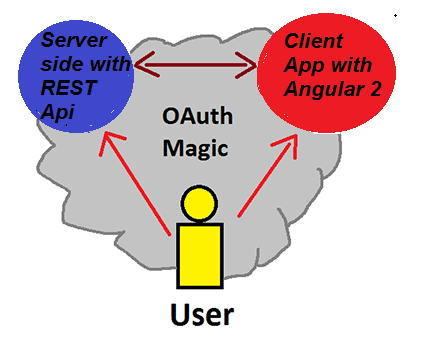


To manage the resource effectively and protect the resource carefully, we can control our resource by OAuth (Open Authentication). When using OAuth, an access token is provided by the client to the resource server to access protected resources. An access token is a string that identifies a user, a page or an application. The token contains the information such as which application created that token and when the token will expire. Then the access token must be validated and verified by resource server that this token is valid and has not expired.

OAuth 2.0 is produced by the IETF OAuth Working Group in October 2012.

OAuth2.0 is an open authorization protocol that allows you to access the resources of the resource owner by enabling the client apps on HTTP services such as Facebook, Google, Twitter ... OAuth2.0 allows you to share the resources stored on a site to other site without giving the credentials. It uses access tokens instead.

This picture below describe how Angular 2 communicate with server side contains the resource through OAuth





**Step 1** – User wants to give the client app access.

**Step 2** – The client app wants access to this user’s data store in the server side.

**Step 3** – The server side asks user to grant access to the client app.

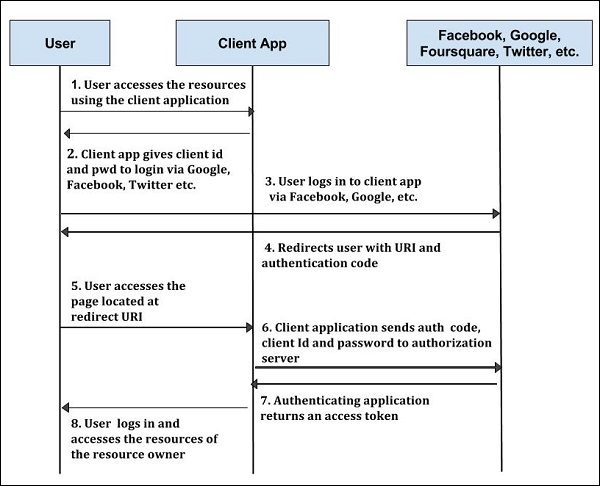
**Step 4** – The user accepts and the server side grants access to the client app.

**Step 5** – The server side creates an access token and gives it to the client app.

**Step 6** – The server side authenticates the access token provided by the client app if it's valid and authorized.

**Step 7** – The server side send requested information of the user stored in server side to the client app.

This sequence diagram below will describe the architectural style of OAuth 2.0

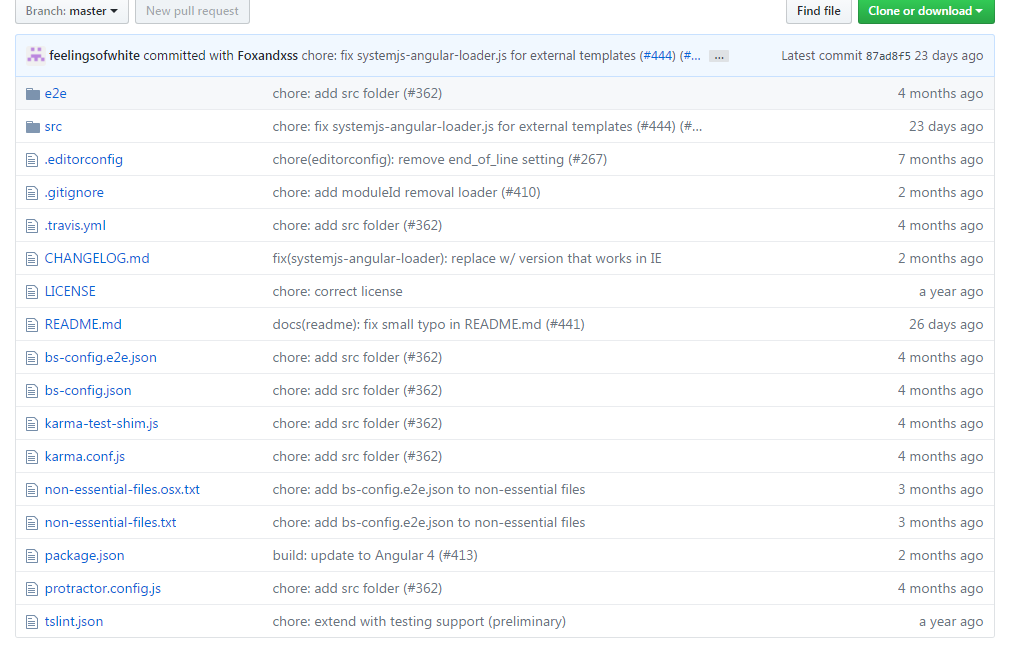


1. Install and setup environment
2. Download Node.js and install

You can go to this link to download <https://nodejs.org/en>

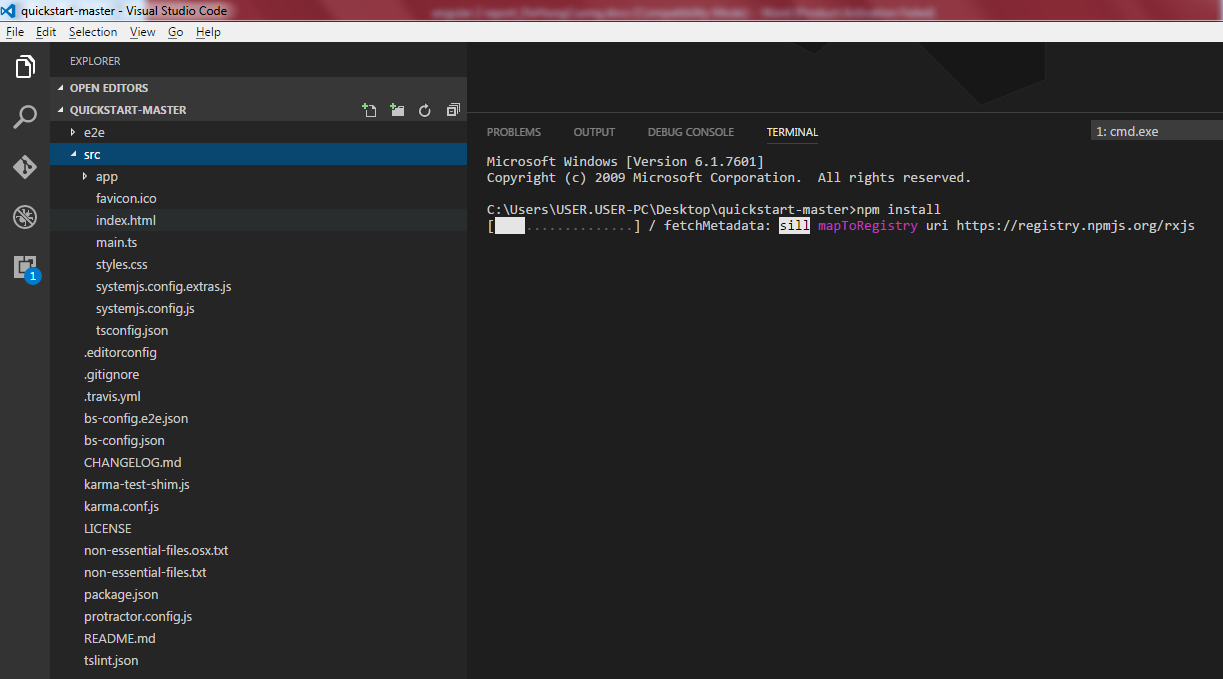


1. Setting up a new project, you can download it from <https://angular.io>. This page will give you a github link to download a sample project Angular 2 quick-start <https://github.com/angular/quickstart/>

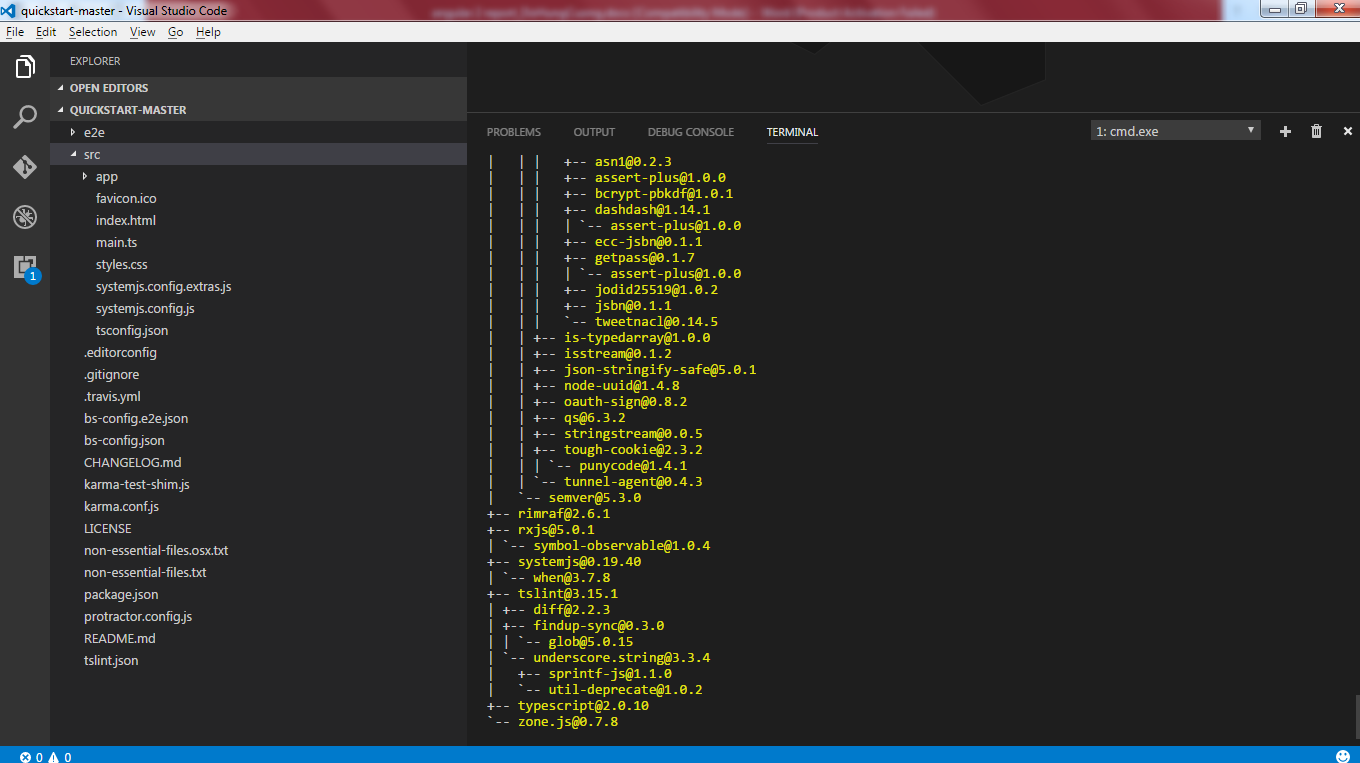


1. After download and install node.js, then unzip source code above. Now you can use VS code ( <https://code.visualstudio.com/> ) or you can run the cmd or terminal to install npm.

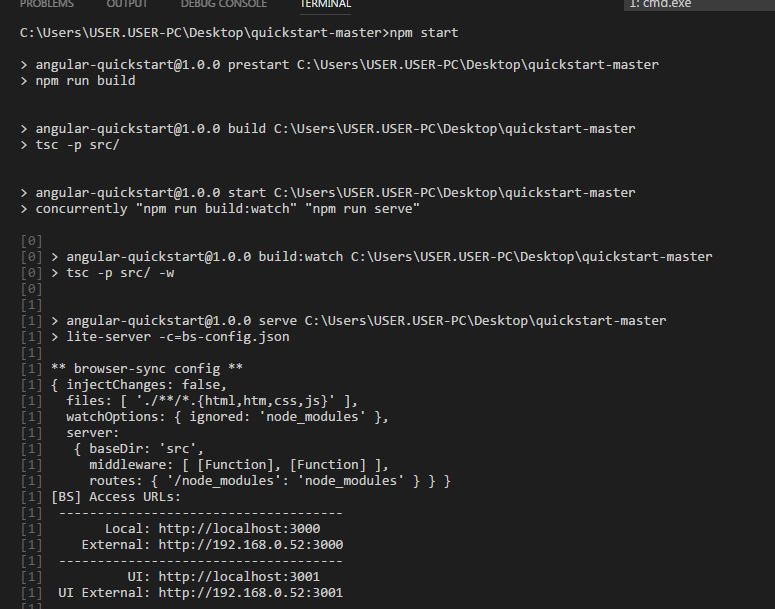
Input ‘npm install’ and wait for a few minute(make sure that the directory is correct)



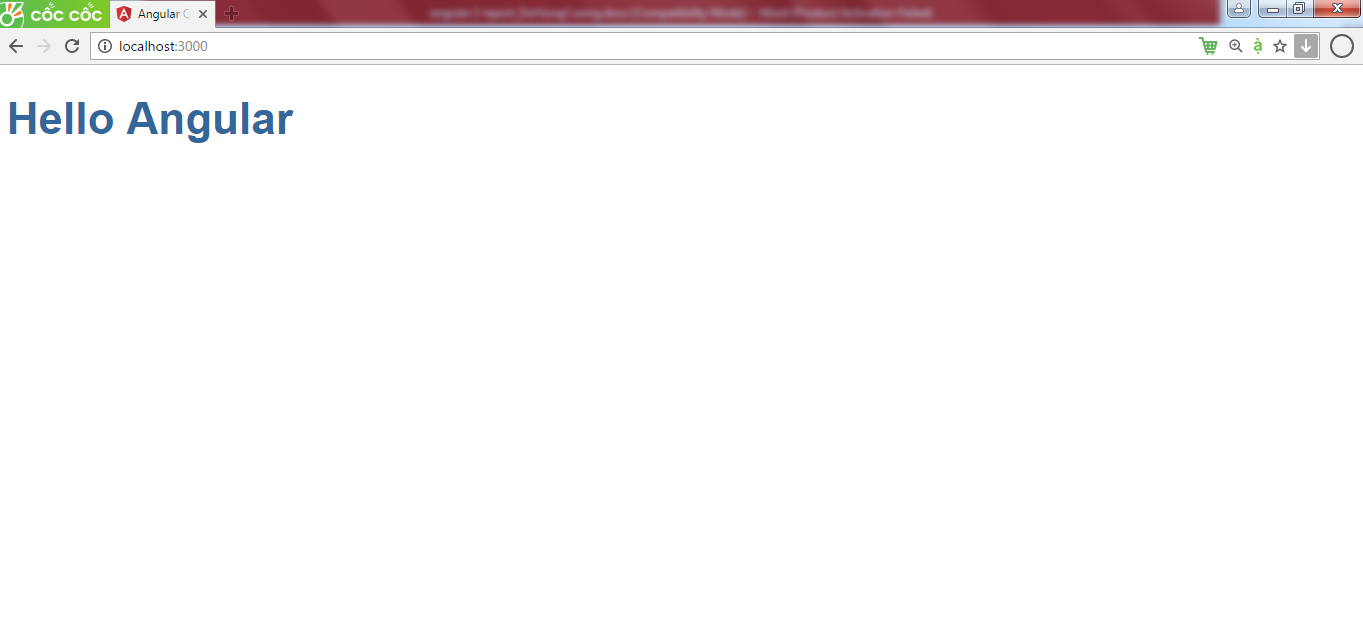
After finish, you can see the list of installed package or library



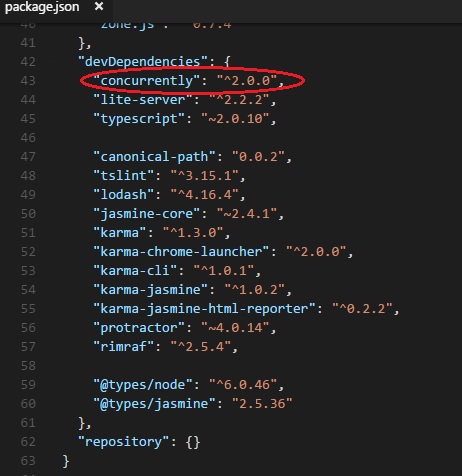
Now you can run your project by the command ‘npm start’ or ‘lite-server’



Go to <http://localhost:3000/> on your browser and view your achievement



If you command ‘npm install’ not work, you should change the "concurrently": "^3.2.0", to "concurrently": "^2.0.0", in the file package.json



I have put the files node-v6.9.5-x64.msi (to install node.js) and VSCodeSetup-1.9.1.exe (to install VS code) and quickstart-master.zip in the setup folder, so you can easily to set up a new angular 2 project.

You can download here: [https://drive.google.com/file/d/0B8NzM\_okLiYLTWpNZ1YzTHMyMkU](https://drive.google.com/file/d/0B8NzM_okLiYLTWpNZ1YzTHMyMkU/view)

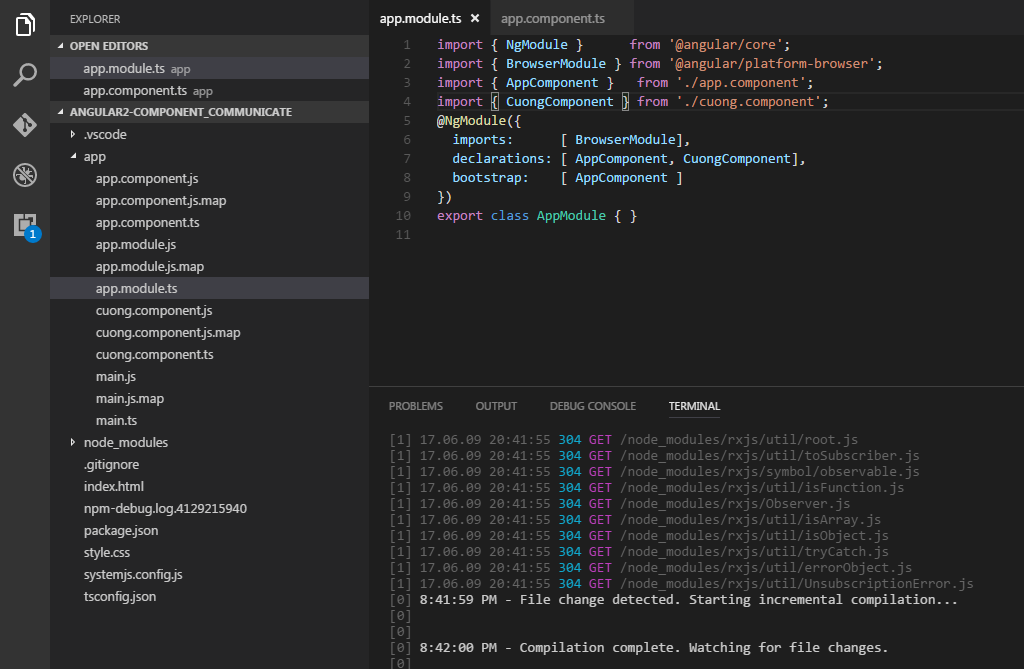
1. Source code and Demo

If you have installed node.js, you can also run my project that I have shown you every week. Here is the list of some projects I had presented:

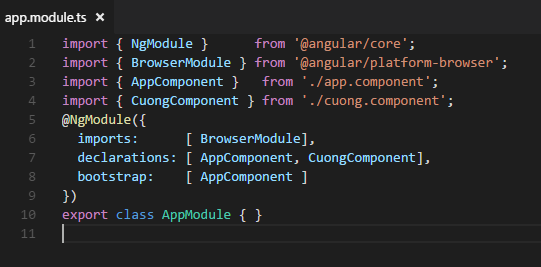
* angular2-first\_app ( first basic project with angular 2)
* angular2-basic\_databinding (data binding)
* angular2-formvalidation (Reservation Form when customer come to hotel)
* angular2-hotel-system (apply some feature of angular 2 to hotel system)
* Angular 2-Restful-OAuth-fontend (connect with java RESTful Web Service and some OAuth code, you have to install java to run the back end: HotelManagementSystem)
* HotelManagementSystem (java RESTful Web Service)

you can download here <https://drive.google.com/drive/folders/0B8NzM_okLiYLY2lDY2tHQWZyMVU>

Angular2-first\_app



Here is the root module



The root module must import the NgModule and BrowserModule to run this application on the browser

The root module declarations 2 components so we have to import 2 components AppComponent and CuongComponent



The AppComponent must import Component from @angular/core to tell Angular 2 it is a component

The AppComponent use CuongComponent so we have to import CuongComponent

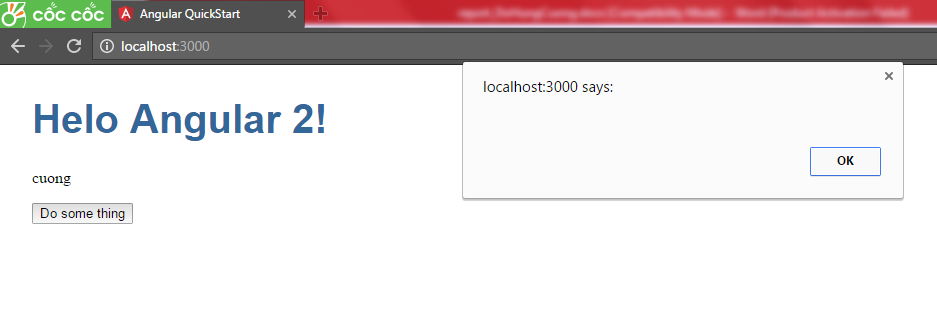
In the view, AppCompnent has variable title and AppComponent is going to use Interpolation Binding to display the variable title and call CuongComponent template by the tag <cuong></cuong> using the selector ‘cuong’ in the CuongComponent that will be shown below.



CuongCompnent has variable name and method dosomething()

In the view, CuongComponent will use Interpolation Binding to display the variable name in class CuongComponent and use Event Binding to catch and handle event when click the button ‘Do some thing’. When click this button, the localhost will alerts the variable name which is assigned to ‘cuong’

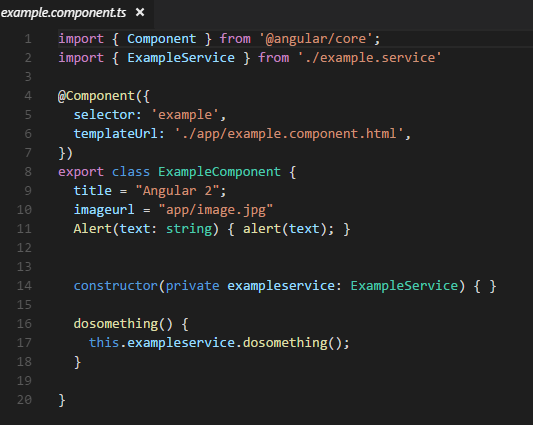
The result of this application is:



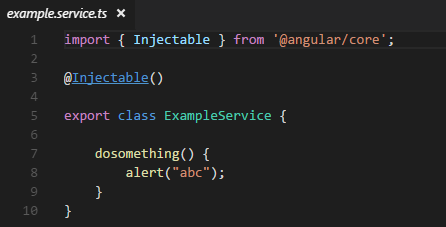
You can download source code here: <https://drive.google.com/file/d/0B8NzM_okLiYLQ3RwQUR1TVE5dDA>

Angular2-basic\_databinding

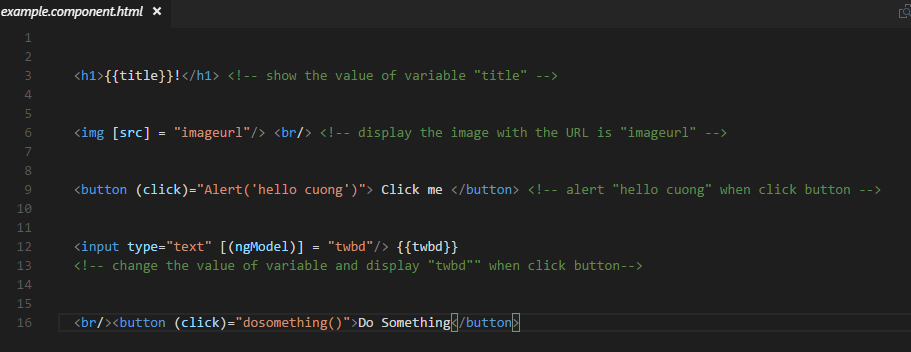
In this second project, I will only show you the important source code, I will keep something unnecessary. Here is the ExampleComponent from this project.



The ExampleComponent have a contructor with a parameter type is ExampleServie so we have to import this ExampleService in this Component. Now we can use the methods or fields in this ExampleService



And the view that provides 4 kinds of data binding that I have shown in part 3: Architecture of angular 2 application



In the view, ExampleComponent will use Interpolation Binding to display the variable title, Property Binding to display the image with src=”…” with ‘…’ is the value of varialbe imageurl, Event Binding for button ‘Click me’ and two way binding for variable twbd, The value of variable ‘twbd’ is change and also change immediately in the browser when input in textbox, click button ‘Do something’ will alert “abc” that provides by the ExampleService.

The result of this application is:

Click Do Something: alert “abc” that provides by the ExampleService.

Input ‘1232132’: change variable twbd and display the changed value immediately beside the textbox

The title ‘Angular 2’: Interpolation Binding to display the variable title

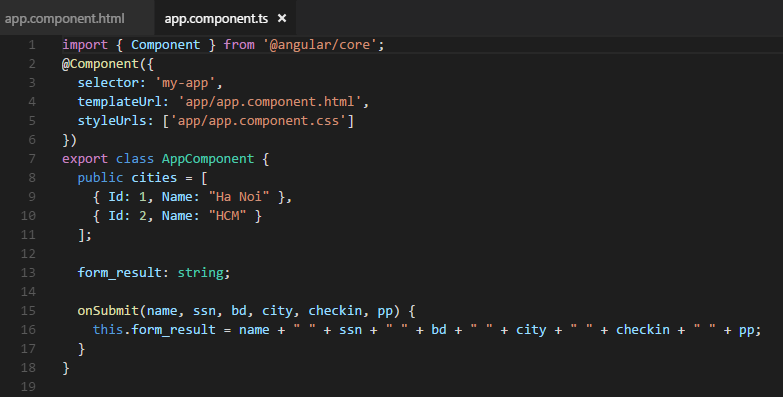
The girl image: Property Binding display the image with src=”app/image.jpg” which is the value of varialbe imageurl



You can download source code here: <https://drive.google.com/file/d/0B8NzM_okLiYLS2tfcjI5MWZFY1E>

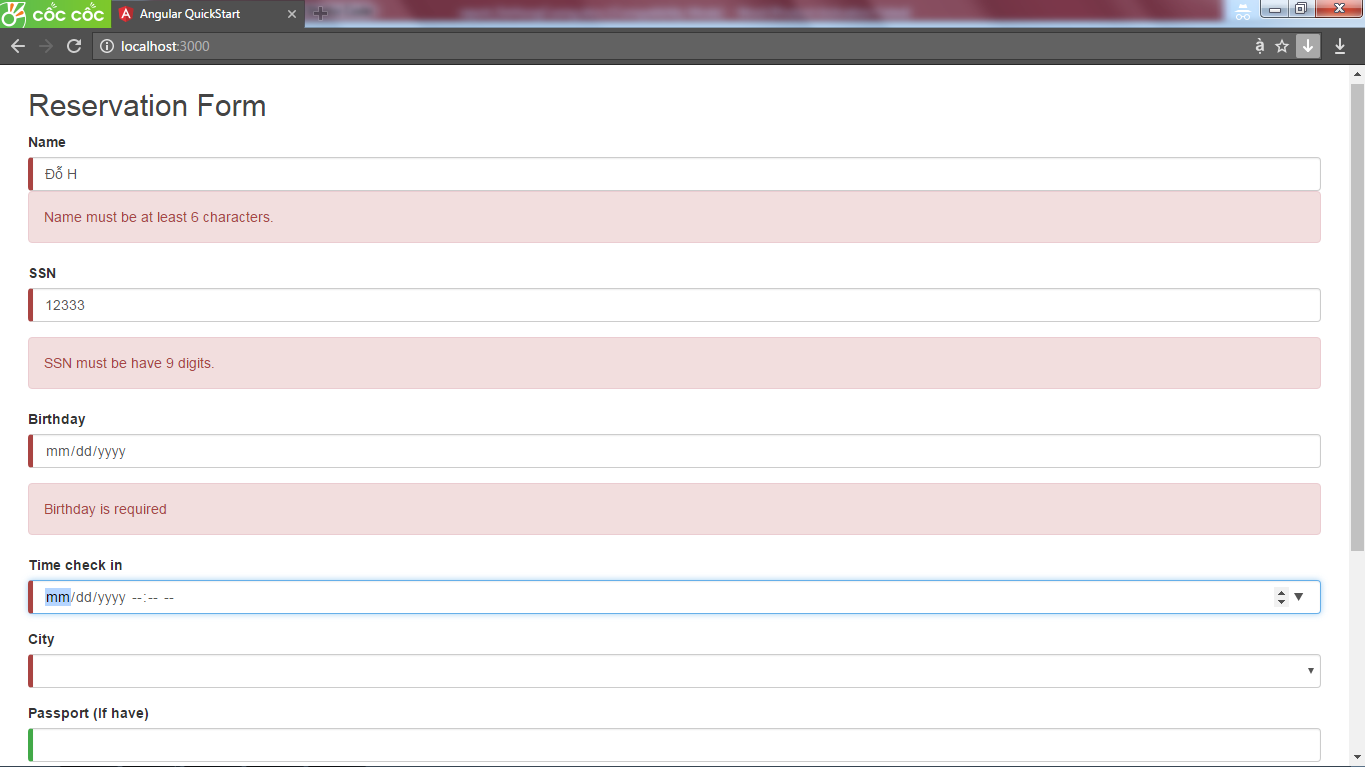
Angular2-Form\_validation

In this project, I had applied data binding to my topic hotel management system. This is a online reservation form when customer come to the hotel. And I use data binding to validate all the information that customer inputs.





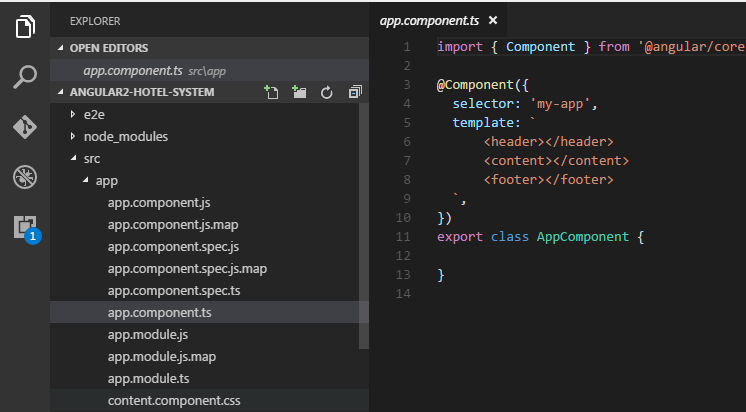
Here is the result of this application:



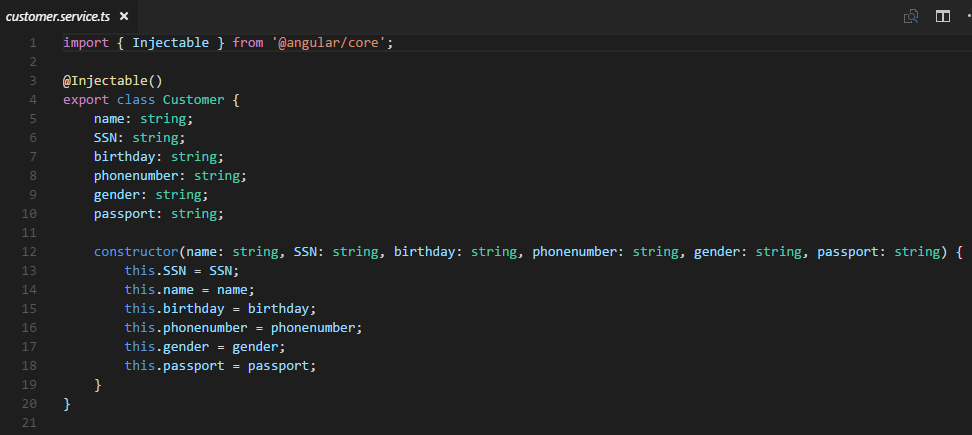
You can download source here: <https://drive.google.com/file/d/0B8NzM_okLiYLWGhRZHUzY3FJVlk>

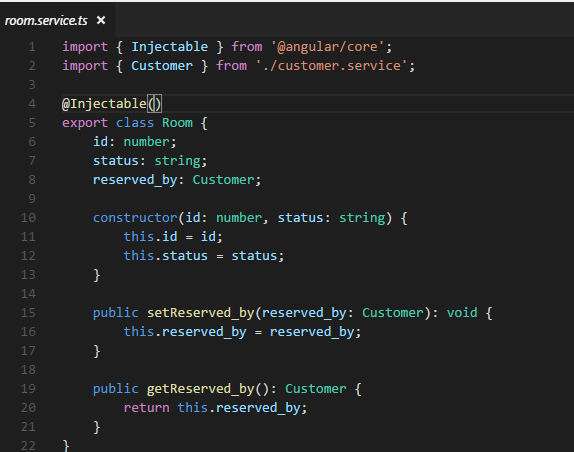
Angular2-hotel-system

In this project, I had focused on service and separated my hotel system into 3 main components header, content and footer.



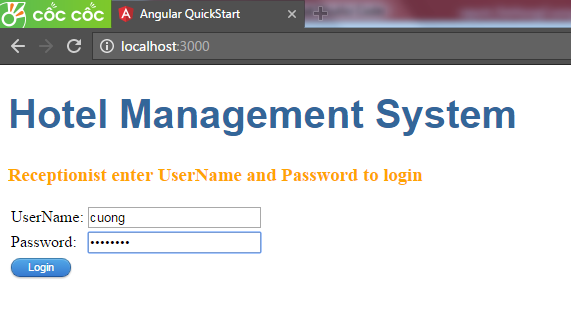


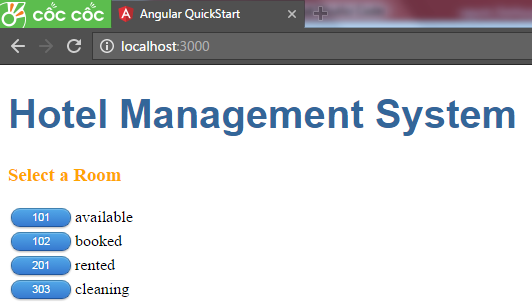


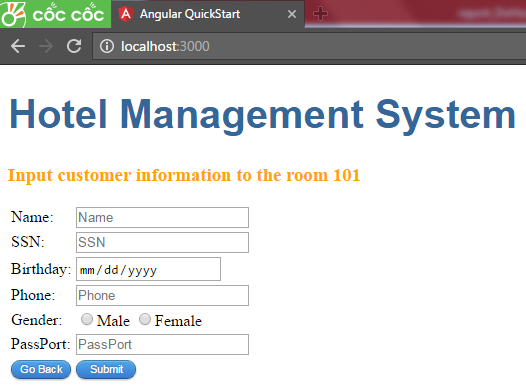


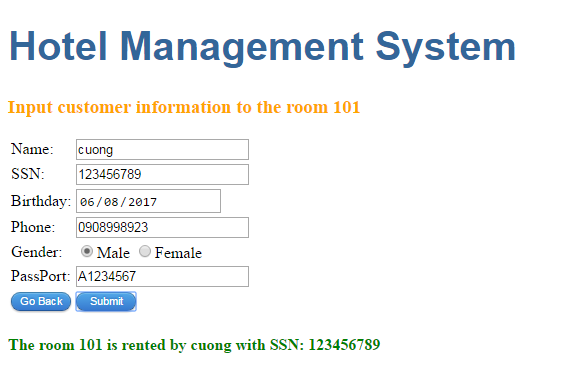


Here is the result of this application:





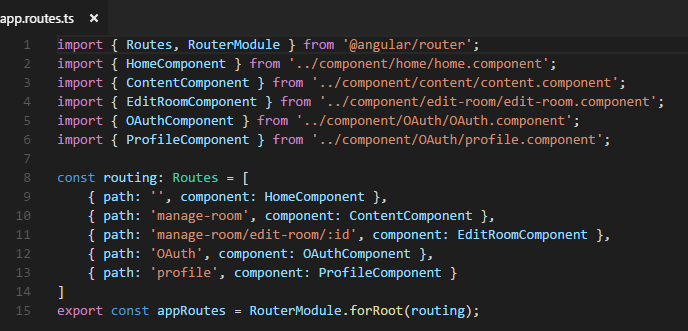




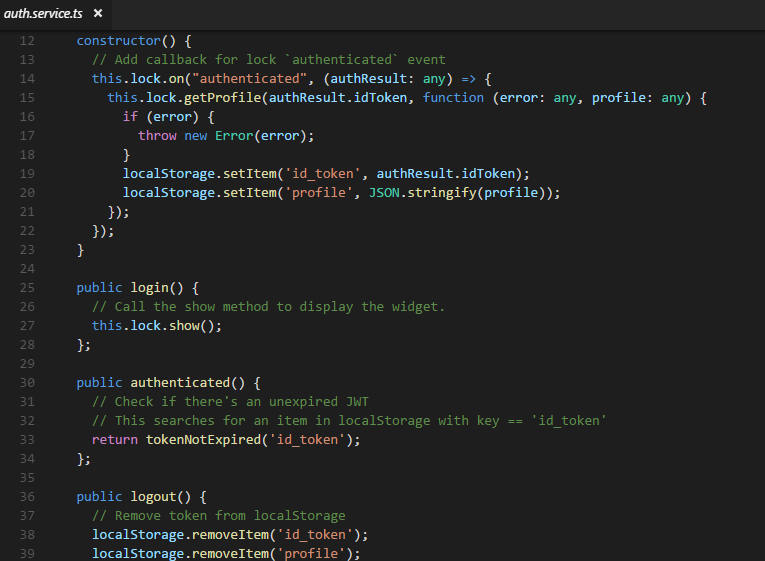
You can download source code here: <https://drive.google.com/file/d/0B8NzM_okLiYLMXpZWENOeTR1TUE>

Angular2-RESTful-Oauth-fontend & Hotel Management System

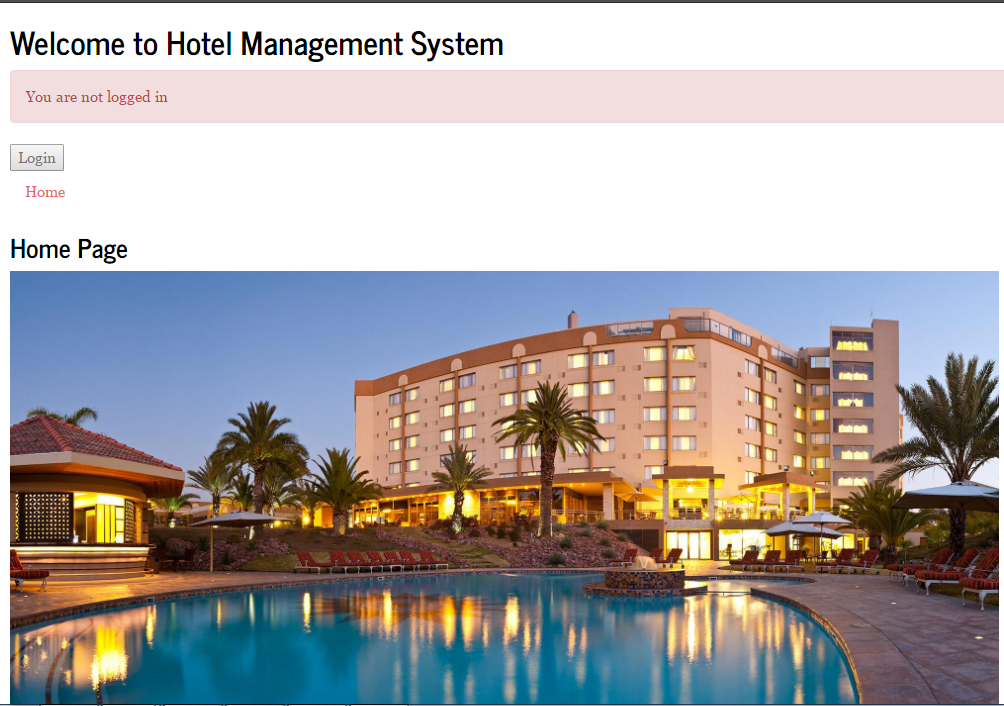
This is an application was built with the fondend using Angular 2 framework and the backend using Java RESTful WebService. I had applied RESTful Web Service (I had shown it in example of part 4: Using RESTful Web Service with Angular 2) and Open Authentication (just the Angular, when user logins with facebook, google or twitter, I did not implement Oauth for Java Restful Web Service). I also use Angular 2 router link. Here is my source code of my appRoutes

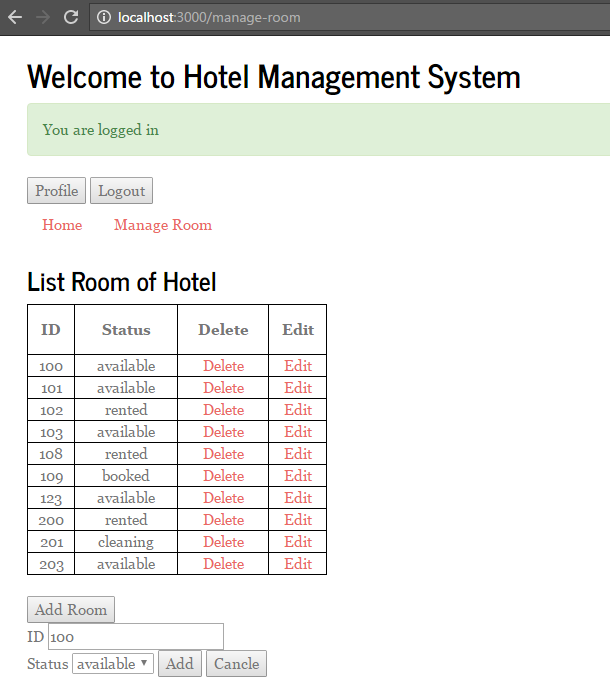


And this is my Authentication service



Here is the result of this application:





You can download source code here:

* Fontend Angular 2: <https://drive.google.com/file/d/0B8NzM_okLiYLYXVveUI4aEp6b3c>
* Backend Java: <https://drive.google.com/file/d/0B8NzM_okLiYLUzRzMVIyTldzaTQ>

1. *Conclusion*

In conclusion, as a user, after seeing the Angular 2 application loaded, I can recognize that the load time is very quick, everything in the website changes immediately. I think that Angular 2 gets something as powerful as the popular Spring framework into the front end. The only one problem I see in Angular 2 is the initial load time. However, this problem can be solved by some creative web designing. Therefore, in my opinion, Angular 2 is one of the best frameworks for building a single page application easily.

1. *References*
2. <https://www.tutorialspoint.com/angular2>
3. <http://www.zymphonies.com/blog/single-page-application-advantages-and-disadvantages>
4. <http://blog.ipointsystems.com/2016/july/angular-2-first-look>
5. <https://msdn.microsoft.com/en-us/magazine/dn890374.aspx>
6. <https://angular.io>
7. <https://blogs.msmvps.com/deborahk/why-angular-why-angular-2>
8. <https://www.tutorialspoint.com/restful>
9. <http://www.restapitutorial.com/lessons/httpmethods.html>
10. <https://www.tutorialspoint.com/oauth2.0>
11. <https://lyness.io/twitter-oauth>