



# Angular web development

Phần 3 – Ôn tập tổng hợp qua ví dụ Quản lý User



# Nội dung

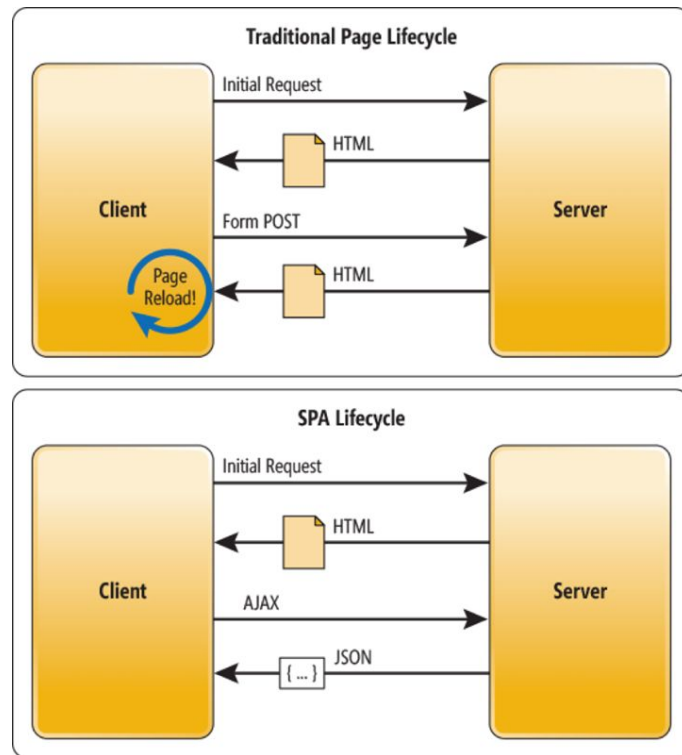
- SPA & Angular Universal
- Router & Navigation
- Angular Form
- REST API & HTTP Services
- Firebase
- NgRx

---

# SPA & Angular Universal

## Single Page Application (SPA)

- **SPA** là ứng dụng mà chỉ có 1 trang duy nhất (single page). Khi người dùng bấm chuyển trang thì không bị tải lại trang mà chỉ thay đổi một phần giao diện, dữ liệu mới được tải qua AJAX. Thông thường SPA web dùng cơ chế **Client Side Rendering** (CSR) để render ra giao diện. Ví dụ một số trang web sử dụng SPA: Gmail, Facebook, Github, ...
- Angular giúp xây dựng các trang web SPA dùng cơ chế CSR dễ dàng hơn.





## Ưu nhược điểm của website dạng SPA dùng cơ chế CSR

### Ưu điểm

- Website chạy nhanh hơn, trải nghiệm người dùng tốt hơn.
- Giảm tải bớt công việc cho Server vì Client phụ trách phần render giao diện.
- Tách biệt giữa Server và Client giúp dự án dễ bảo trì hơn.

### Nhược điểm

- Với các website thiên về hiển thị nội dung thì SEO kém hơn.
- Công việc phía Client nặng hơn do đảm nhiệm hết phần xử lý giao diện.
- Phát triển website phức tạp hơn dạng truyền thống.



## Angular Universal

- Để khắc phục nhược điểm chính của SPA là kém về SEO, Angular cung cấp thêm kỹ thuật **Server Side Rendering (SSR) - Angular Universal**. Theo cách này thì với lần đầu request từ người dùng, nội dung của Website sẽ được render từ trước phía server, sau đó từ các lần truy cập tiếp theo thì nó hoạt động tương tự CSR.
- **Angular Universal** giúp website có thể SEO tốt như các website truyền thống, và còn giúp giảm thời gian truy cập website lần đầu tiên. Tuy nhiên quy trình xây dựng ứng dụng phức tạp hơn, yêu cầu về phía server cũng cao hơn, cần có 1 server backend như NodeJS.
- Tham khảo thêm: <https://angular.io/guide/universal>



## Practice 1



- Tạo App bằng Angular CLI.
- Sử dụng một thư viện UI bất kỳ tạo giao diện trang quản trị.


















Dashboard

Users

+ New User

Search User

Avatar	Name	Email	Birthday	Action
	<a href="#">Margaret Rayworth</a>	mrayworth2r@fastcompany.com	3/11/2006	 
	<a href="#">Clio Maxstead</a>	cmaxstead2q@goodreads.com	12/15/1996	 
	<a href="#">Alina Beaglehole</a>	abeaglehole2p@patch.com	6/5/1994	 
	<a href="#">Brooke Signe</a>	bsigne2o@thetimes.co.uk	5/24/1994	 
	<a href="#">Jenna Kringe</a>	jkringe2n@ox.ac.uk	3/23/1995	 
Items per page: 5 1 - 5 of 100  < < > >				

Demo bố cục một trang quản trị và phần Quản lý User



---

# Router & Navigation

## Angular Router

- **Angular Router** cho phép tạo các điều hướng trên trang web, giúp người dùng chuyển trang và thay đổi đường dẫn nhưng không reload lại trang (chỉ thay đổi view).  
Tham khảo: <https://angular.io/guide/router>.
- Khi dùng **Angular CLI** tạo project thì chọn Yes khi được hỏi có dùng **Angular Router** hay không.



```
~/Desktop ➤ ng new angular-project
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
CREATE angular-project/README.md (1031 bytes)
CREATE angular-project/.editorconfig (246 bytes)
```

## Cấu hình Routes

- Sau khi tạo ứng dụng có sử dụng **Angular Router**, chúng ta sẽ cấu hình *routes* (đường dẫn) cho website ở file **app-routing.module.ts**.
- Routes sẽ chỉ định với đường dẫn nào thì sẽ hiển thị giao diện nào, cấu hình qua phương thức **RouterModule.forRoot()** và được thêm vào phần **imports** của **AppModule**.

```
> const routes: Routes = [...];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule {}
```

```
const routes: Routes = [  
  { path: 'products', component: ProductsComponent },  
  {  
    path: 'products/:id',  Route Parameter  
    component: ProductDetailComponent  
  },  
  { path: 'about', component: AboutComponent },  
  { path: '', component: HomeComponent },  
  { path: '**', component: PageNotFoundComponent }  
];  
  
 Đại diện cho mọi đường dẫn  
  
@NgModule({  
  imports: [RouterModule.forRoot(routes)],  
  exports: [RouterModule]  
})  
  
export class AppRoutingModule {}
```

Ví dụ cấu hình Routes cho các trang: Home, About, Products, Product Detail, Not found.

```
<nav>
  <a
    routerLink="/"
    routerLinkActive="active"
    [routerLinkActiveOptions]="{ exact: true }"
  >
    Home
  </a>

  <a routerLink="/products" routerLinkActive="active">Products</a>

  <a routerLink="/about" routerLinkActive="active">About</a>
</nav>

<router-outlet></router-outlet>
```

**Đường dẫn trở đến**

**Chỉ thêm class active nếu đường dẫn trùng routerLink**

**Thêm class active khi đang ở trang đó**

**Nơi render view khi đường dẫn thay đổi**

Hiển thị đường dẫn và thiết lập chỗ View thay đổi (router-outlet).

```
import { Component, OnInit } from '@angular/core';
import { Router, ActivatedRoute } from '@angular/router';

@Component({
  selector: 'app-product-detail',
  templateUrl: './product-detail.component.html',
  styleUrls: ['./product-detail.component.css']
})
export class ProductDetailComponent implements OnInit {
  productId: string = '';

  constructor(private router: Router, private activatedRoute: ActivatedRoute) {}

  ngOnInit() {
    this.productId = this.activatedRoute.snapshot.paramMap.get('id');
  }
}
```


Ví dụ lấy dữ liệu từ *route parameter* (nếu cần tái sử dụng Component thì dùng [cách này](#)).

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { UserService } from 'src/app/services/user.service';

@Component({ ...
})
export class LoginComponent {
  constructor(private router: Router, private userService: UserService) {}

  login() {
    this.userService
      .login('email', 'password')
      .then(result => {
        if (result.user) {
          this.router.navigateByUrl('/');
        }
      })
  }
}
```

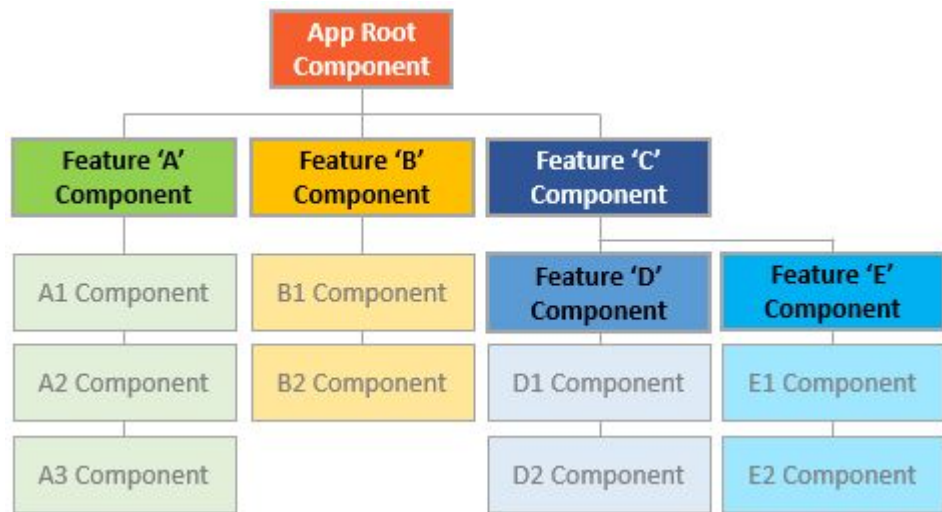
Chuyển sang trang chủ



Ví dụ chuyển trang về trang chủ sau khi người dùng Login

## Child Routes

- Trong 1 ứng dụng phức tạp có thể chia làm nhiều Feature, nhiều đường dẫn. Trong 1 Feature lại có thể có những phần nhỏ giao diện thay đổi theo đường dẫn.
- Chúng ta có thể cấu hình Child Routes để đáp ứng yêu cầu này.





```
const routes: Routes = [
  {
    path: 'profile',
    component: ProfileComponent,
    children: [
      { path: 'account', component: AccountComponent },
      { path: 'security', component: SecurityComponent }
    ]
  },
  {
    path: '',
    component: HomeComponent
  },
  {
    path: '**',
    component: PageNotFoundComponent
  }
];
```

**Annotations:**

- `/profile` (blue arrow pointing to `path: 'profile'`)
- Trong ProfileComponent có chứa `<router-outlet>`** (red arrow pointing to `ProfileComponent`)
- `/profile/account` (blue arrow pointing to `path: 'account'`)
- `/profile/security` (blue arrow pointing to `path: 'security'`)

Ví dụ cấu hình Child Routes



## Router guards

- Trong website sẽ có những trang mà chỉ cho phép người dùng có quyền mới được truy cập (authenticated user), ví dụ như các trang quản trị website.
- Cấu hình route có thể thêm các guard cho phép xử lý các trường hợp về phân quyền cho route: Kiểm tra người dùng đã đăng nhập chưa, có quyền truy cập route hay không và redirect người dùng về trang khác (ví dụ trang login).
- Tham khảo: <https://angular.io/guide/router#guards>.



## Practice 2



- Tạo các Component đại diện cho các Page của website.
- Cấu hình Router cho website.

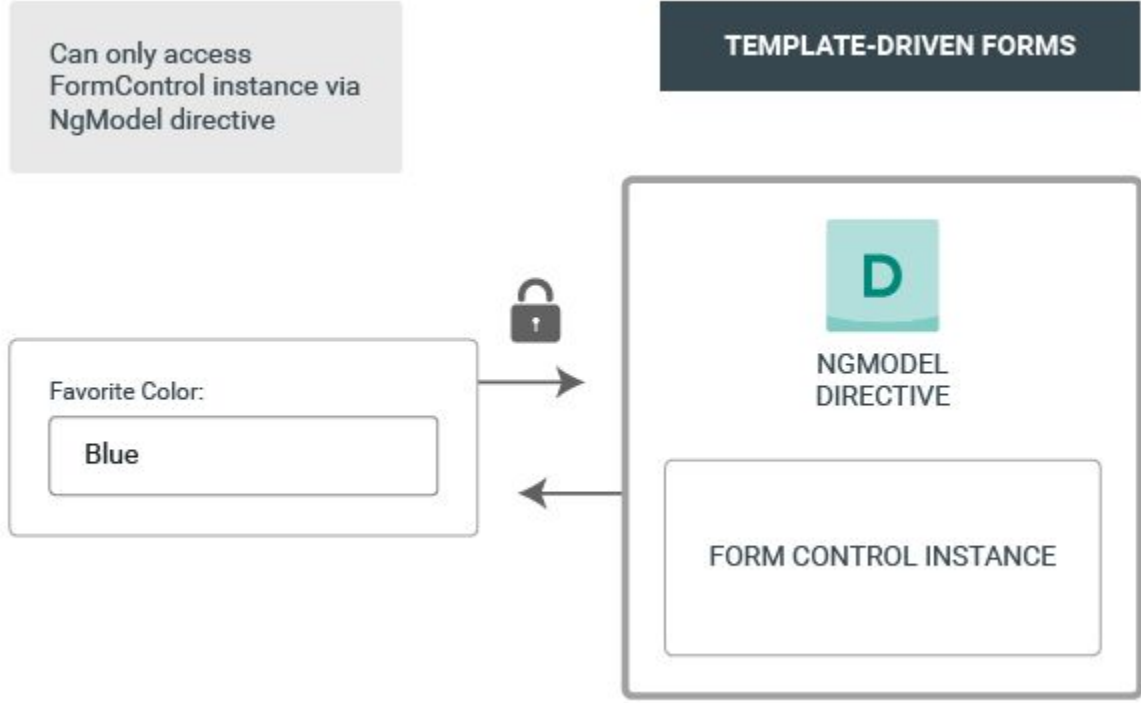
---

# Angular Form

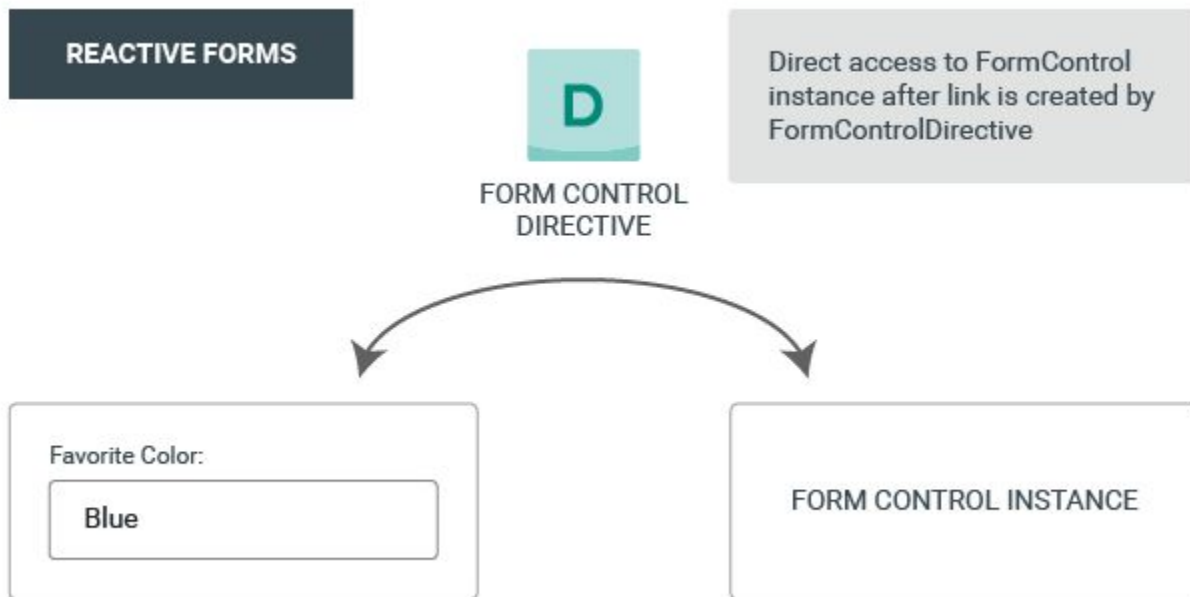


## Angular Form

- Angular cung cấp 2 cách tiếp cận với Form để xử lý dữ liệu do người dùng nhập:  
**Reactive Forms** và **Template-driven Forms**.
- **Reactive Forms** (*Model-driven*) cho phép tạo và xử lý Form từ phía Class, giúp Form có thể dễ dàng mở rộng, tái sử dụng ở nhiều nơi trong ứng dụng. Để sử dụng cần import **ReactiveFormsModule**.
- **Template-driven Forms** thì cho phép tạo Form luôn ở template, nó có tính mở rộng kém hơn **Reactive Forms** nhưng sử dụng đơn giản hơn, thường dùng cho các Form đơn giản trong ứng dụng. Để sử dụng cần import **FormsModule**.



Minh họa Template-driven Forms



Minh họa Reactive Forms



## Template-driven Forms

- Sử dụng **ngModel** để binding dữ liệu (có thể dùng trên 1 đối tượng). Ngoài ra **ngModel** còn giúp kiểm tra xem người dùng đã tương tác với form chưa, đã thay đổi dữ liệu hay chưa và dữ liệu nhập có hợp lệ hay không.
- Tham khảo: <https://angular.io/guide/forms>

State	Class if true	Class if false
The control has been visited.	ng-touched	ng-untouched
The control's value has changed.	ng-dirty	ng-pristine
The control's value is valid.	ng-valid	ng-invalid





## Reactive Forms

- Reactive Forms tạo form bằng cách khai báo trong Component dưới dạng các Object Model, sau đó sử dụng chúng để tạo form ở template và có thể lấy data form trực tiếp từ Object Model này.
- Tham khảo thêm: <https://angular.io/guide/reactive-forms>



## Tạo form bằng Reactive Forms

- **Bước 1:** Import **ReactiveFormsModule** trong **AppModule**.
- **Bước 2:** Tạo các **FormControl** (các thẻ input, textarea, select, ...) và **FormGroup** (chứa một nhóm các FormControl). Có 2 cách tạo FormControl đó là tạo instance thủ công hoặc sử dụng **FormBuilder**.
- **Bước 3:** Đăng ký FormGroup và các FormControl bên template.

```
profileForm = new FormGroup({  
  firstName: new FormControl(''),  
  lastName: new FormControl(''),  
  address: new FormGroup({  
    street: new FormControl(''),  
    city: new FormControl(''),  
    state: new FormControl(''),  
    zip: new FormControl('')  
  })  
});
```

```
profileForm = this.fb.group({  
  firstName: [''],  
  lastName: [''],  
  address: this.fb.group({  
    street: [''],  
    city: [''],  
    state: [''],  
    zip: ['']  
  })  
});
```



## Form Validation

- **Form Validation** là kiểm tra tính hợp lệ của input để đảm bảo người dùng nhập đủ và đúng định dạng dữ liệu, ví dụ như: Email, Password, ...
- Với **Template-driven Forms** ta có thể validate bằng cách khai báo các [HTML form validation](#) như: required, min-length, max-length, pattern, ...
- Với **Reactive Forms** thì validate dữ liệu bằng cách gọi các hàm bên phía Component class, có thể tự viết hoặc sử dụng luôn những hàm Validate có sẵn của Angular (built-in validators).
- Tham khảo: <https://angular.io/guide/form-validation>

## Practice 3



- Tạo một component mới để hiển thị nội dung trang Profile.
- Trên component Profile, tạo Form cập nhật thông tin cá nhân, có validate và khi bấm nút Update Profile cần lấy được toàn bộ thông tin trên form.

Dashboard

Users

## Profile

Name

Birthday


Phone

Bio

Interest 1

Interest 2

Update Profile



Change Avatar

Demo Form cập nhật thông tin cá nhân (Profile)

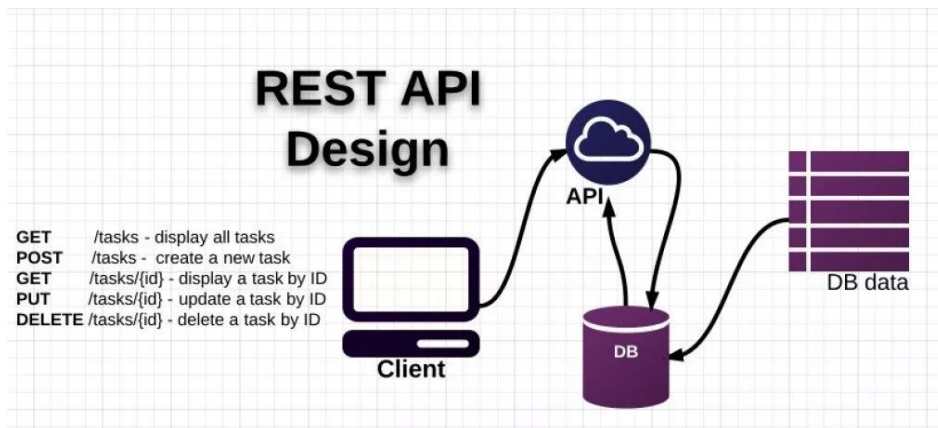
---

# REST API & HTTP Service

## REST API là gì?

- **API** (**A**pplication **P**rogramming **I**nterface) là giao diện lập trình ứng dụng giúp tạo ra các phương thức kết nối với các thư viện và ứng dụng khác nhau.
- **REST** (**R**epresentational **S**tate **T**ransfer) là 1 dạng cấu trúc dùng cho việc chuyển đổi dữ liệu.

=> **REST API**: Các **API** theo cấu trúc **REST**.



## Postman

- Là một nền tảng giúp phát triển ứng dụng kết nối với API dễ dàng và nhanh chóng hơn.
- Giúp lập trình viên thực hiện các request đến các API trước khi viết code cho ứng dụng.
- Ngoài ra còn cung cấp nhiều chức năng như: Automated Testing, Design & Mock endpoint, ...





GET https://quan-ly-hoc-vien.herokuapp.com/users

API

Untitled Request

GET https://quan-ly-hoc-vien.herokuapp.com/users

Send Save

Params Authorization Headers (9) Body Pre-request Script Tests Cookies Code Comments (0)

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Kết quả

Body Cookies Headers (14) Test Results Status: 200 OK Time: 1288ms Size: 837 B Save Response

Pretty Raw Preview JSON

```
1 [
2   {
3     "name": "Winnifred Tribe",
4     "birthYear": "2000",
5     "email": "wtribe0@kickstarter.com",
6     "phone": "867-130-6017",
7     "id": 1
8   },
```

Ví dụ thực hiện request lên API từ Postman



## HTTP Services

- Để kết nối ứng dụng Angular với Server qua giao thức HTTP ta sử dụng **HTTP Services** (HttpClient).
- Để bất kỳ nơi nào trong ứng dụng đều có thể sử dụng HttpClient ta cần import **HttpClientModule** (`@angular/common/http`) vào **AppModule**. Sau đó chỗ nào cần sử dụng HttpClient thì cần import **HttpClient** và inject vào *constructor* của class đó.
- Tham khảo: <https://angular.io/guide/http>

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';

import { User } from '../models/user.model';

const API: string = 'https://quan-ly-hoc-vien.herokuapp.com/users';

@Injectable({
  providedIn: 'root'
})
export class UserService {
  constructor(private http: HttpClient) {}

  getUsersHTTP() {
    return this.http.get<User[]>(API);
  }
}
```

Thực hiện GET request lên API

Ví dụ gọi request lấy danh sách User từ API trong UserService



## Practice 4



- Dùng Postman để test API.
- Lấy dữ liệu từ API đổ vào giao diện bằng cách sử dụng HTTP Client.

---

# Firebase

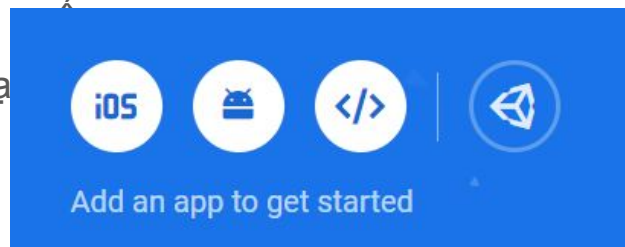
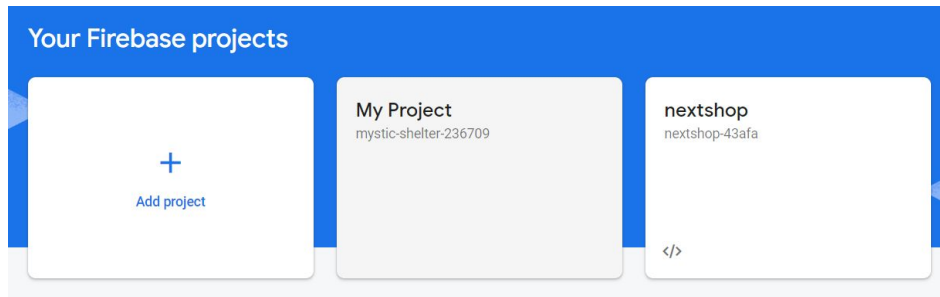
## Firestore là gì?



- Là một nền tảng phát triển các ứng dụng Web và Mobile, cung cấp rất nhiều dịch vụ mạnh mẽ cho các lập trình viên.
- Một số dịch vụ phổ biến: Cloud Firestore (Cơ sở dữ liệu), Authentication (Xác thực người dùng), Hosting, Cloud storage (Lưu trữ file), ...
- Trang chủ: <https://firebase.google.com>

## Sử dụng Firebase

- **Bước 1:** Đăng ký tài khoản trên website, sau đó truy cập vào mục **Console** để quản lý project.
- **Bước 2:** Tạo 1 project, trong Project vừa tạo sẽ có phần thêm các App (Android, iOS, Web, Unity) và có phần hình các dịch vụ cho App (dùng chung). Sau khi App sẽ có code SDK để nhúng vào App.



2

## Add Firebase SDK

Copy and paste these scripts into the bottom of your <body> tag, but before you use any Firebase services:

```
<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/7.2.1/firebase-app.js"></script>

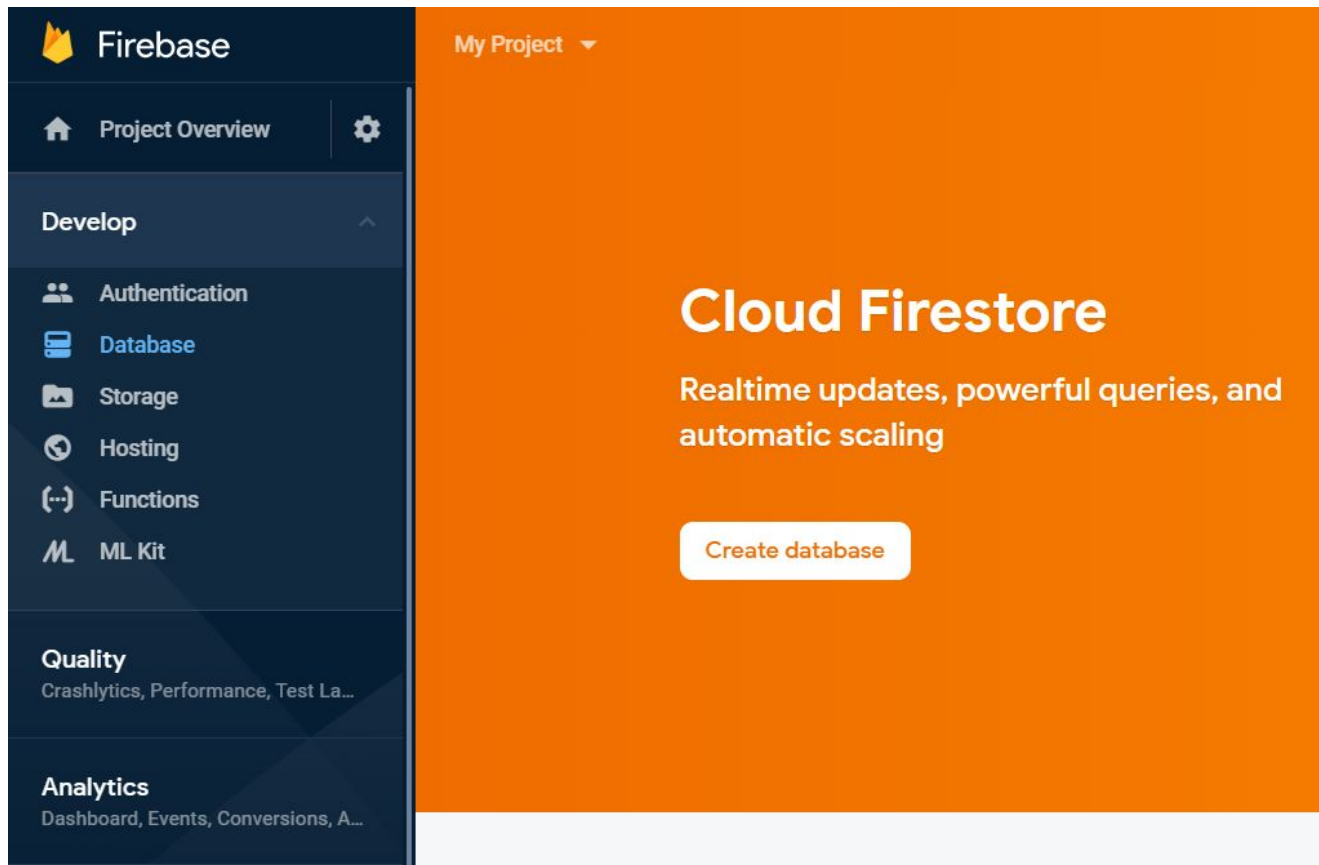
<!-- TODO: Add SDKs for Firebase products that you want to use
      https://firebase.google.com/docs/web/setup#available-libraries -->

<script>
  // Your web app's Firebase configuration
  var firebaseConfig = {
    apiKey: "AIzaSyDp6bNY7IIG0coJwB_P14T0915Z0nqm29M",
    authDomain: "mystic-shelter-236709.firebaseio.com",
    databaseURL: "https://mystic-shelter-236709.firebaseio.com",
    projectId: "mystic-shelter-236709",
    storageBucket: "mystic-shelter-236709.appspot.com",
    messagingSenderId: "45925073154",
    appId: "1:45925073154:web:bcd0fa85cfd2712166c21b"
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
</script>
```



Firebase SDK cho Web app (có thể xem lại trong Project Settings)





Quản lý các dịch vụ của Firebase dùng cho Project

# Database

Cloud Firestore

[Data](#)[Rules](#)[Indexes](#)[Usage](#)

nextshop-43afa

+ Start collection

categories

messages

products

users >

Danh sách Collection

users

+ Add document

GxEc81SU3g33NDG5JS6X >

fBY0ukzs9aW4qeKzkogD

vMR3R96yVjfyfeZq3Ed6

Danh sách Document trong một Collection

GxEc81SU3g33NDG5JS6X

+ Start collection

+ Add field

Anyone on the internet can write to this document

email: "robin@gmail.com"

name: "Robin"

Chi tiết một Document (có nhiều field)

Ví dụ sử dụng dịch vụ Cloud Firestore (Database)

# Database

Cloud Firestore

Data

**Rules**

Indexes

Usage



Unit test and debug your Rules by downloading and running the Emulator. Run `$ firebase emulators:start --only`



Today • 5:28 PM



Sep 16, 2019 • 4:40 PM



Sep 16, 2019 • 4:20 PM



Sep 3, 2019 • 12:29 AM



Sep 3, 2019 • 12:23 AM

```
1 rules_version = '2';
2 service cloud.firestore {
3   match /databases/{database}/documents {
4     match /{document=**} {
5       allow read: if true;
6       allow write: if request.auth.uid != null;
7     }
8   }
9 }
```

Cho phép đọc dữ liệu thoải mái, nhưng  
để ghi dữ liệu thì cần phải đăng nhập

Chú ý cập nhật Rules (phân quyền cho API)



# Kết nối vào Firebase từ ứng dụng Angular

- Sử dụng thư viện AngularFire: <https://firebaseopensource.com/projects/angular/angularfire2/>
- Hướng dẫn cài đặt:  
<https://github.com/angular/angularfire/blob/master/docs/install-and-setup.md>

AngularFire

Contents  
What is AngularFire?  
Install  
Example use:  
Developer Guide

The official library for Firebase and Angular

★ 5493

🕒 5 days ago

< > View Source

🐛 File Bug



stephangrobler and jamesdaniels docs(functions): mention proxy is for us-central only (#2191) ...

Latest commit cb5ae49 9 days ago

..

auth	docs(auth-guard): work around for source.lift for now (#2223)	9 days ago
deploy	feat(schematics): ng deploy schematic (#2046)	5 months ago
firestore	feat(firestore): Support Firestore Collection Group Queries (#2066)	5 months ago
functions	docs(functions): mention proxy is for us-central only (#2191)	9 days ago
ionic	docs(universal): getting started, serving with Cloud Functions, and p...	last year
messaging	docs(fcm): typo caused by copying from cloud storage (#1944)	last year
performance	docs(performance): Adding information on polyfilling FID (#2096)	4 months ago
rtdb	docs(rtdb): update lists.md to fix typo (#1910)	last year
storage	Add StorageBucket injection token to storage docs (#1958)	11 months ago
universal	feat(schematics): ng deploy schematic (#2046)	5 months ago

AngularFire Document: <https://github.com/angular/angularfire/tree/master/docs>

```
import { AngularFireModule } from '@angular/fire';
import { AngularFireAuthModule } from '@angular/fire/auth';
import { AngularFireAuthGuard } from '@angular/fire/auth-guard';
import { AngularFireStoreModule } from '@angular/fire/firestore';
import { AngularFireStorageModule } from '@angular/fire/storage';
import { environment } from '../environments/environment';

import { AppComponent } from './app.component';

@NgModule({
  declarations: [ ... ],
  imports: [
    BrowserModule,
    AngularFireModule.initializeApp(environment.firebase),
    AngularFireAuthModule,
    AngularFireStoreModule,
    AngularFireStorageModule
  ],
  providers: [AngularFireAuthGuard],
  bootstrap: [AppComponent]
})
```

Chú ý với **AngularFireAuthGuard** cần khai báo thêm ở phần **providers** của **AppModule**

## Practice 5



- Tạo tài khoản, project, database trên Firebase.
- Sử dụng AngularFire để kết nối ứng dụng với Firebase.
- Thêm các chức năng liên quan đến dữ liệu và kết nối với Firebase:  
Thêm, sửa, xóa dữ liệu, ảnh, ...



## Firestore Cloud Firestore REST API

- Firestore Cloud Firestore cũng cung cấp REST API cho phép gọi API trực tiếp không qua native client library.
- REST API endpoint có URL gốc là: <https://firestore.googleapis.com/v1/>
- Tham khảo: <https://firebase.google.com/docs/firestore/use-rest-api>





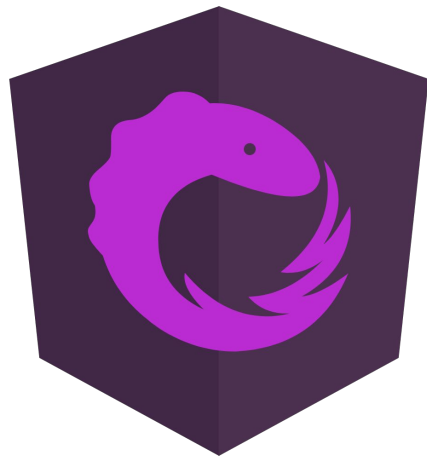
## Deploy Angular app lên Firebase

- Cài đặt Firebase CLI: **npm install -g firebase-tools**
- Đăng nhập Firebase: **firebase login**
- Cấu hình Firebase: **firebase init** (chú ý khi cấu hình phần *public directory* chọn **dist/[project-name]**)
- Build project sau đó chạy lệnh: **firebase deploy**



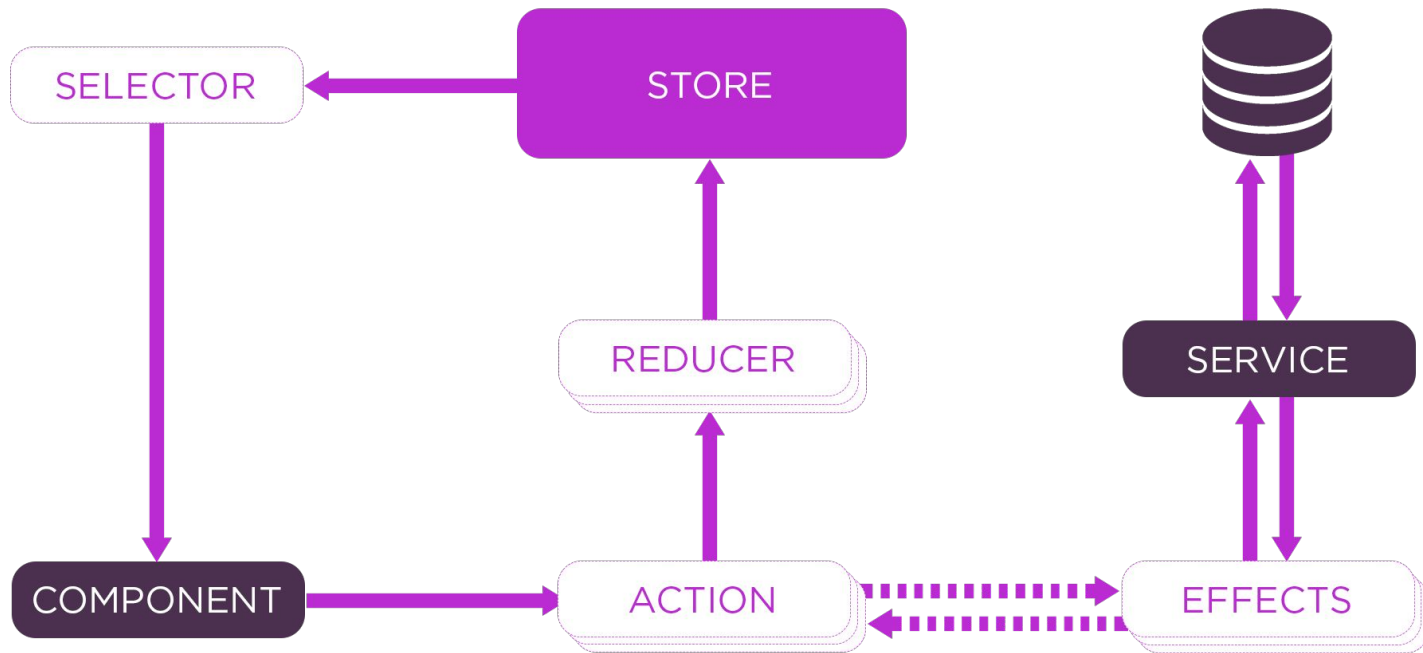
## NgRx

- Reactive State cho Angular: <https://ngrx.io/>
- Thư viện dùng để quản lý trạng thái dữ liệu (state management) cho Angular, tương tự [Redux](#). Giúp quản lý và truyền dữ liệu giữa các Component một cách dễ dàng, không bị hạn chế bởi phân cấp Component.





# NGRX STATE MANAGEMENT LIFECYCLE



Mô hình quản lý State của NgRx



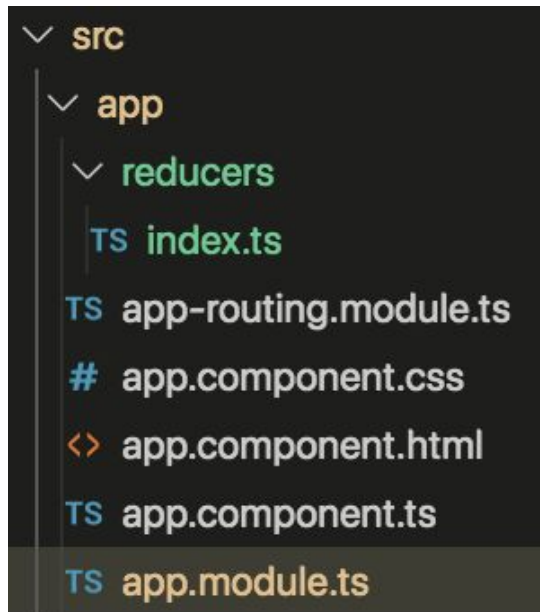
## Cài đặt

- Hướng dẫn: <https://ngrx.io/guide/store/install>

- Cài đặt bằng Angular CLI:

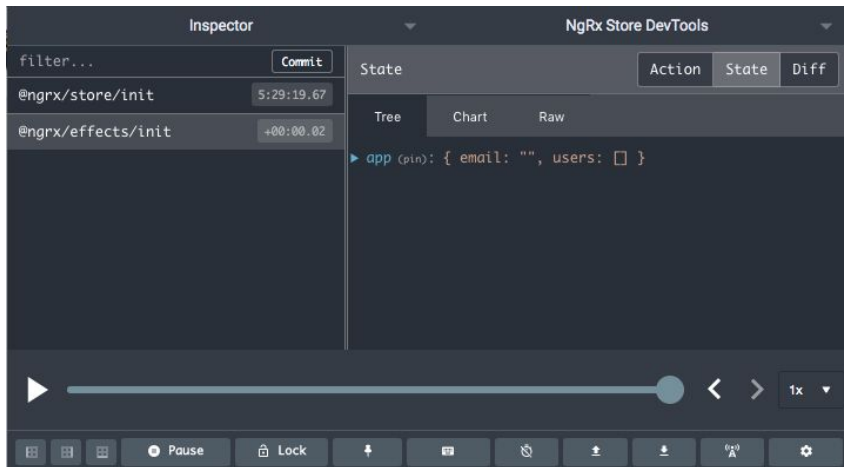
```
> ng add @ngrx/store
```

- Khi cài bằng Angular CLI sẽ có file cấu hình mẫu  
*src/app/reducers/index.ts*



## Store Devtools

- Cài thêm Extension Redux Devtools trên trình duyệt sẽ giúp debug ứng dụng dễ dàng hơn.
- Hướng dẫn cài đặt: <https://ngrx.io/guide/store-devtools/install>



```
export interface State {  
  email: string;  
  users: User[];  
}  
  
export const initialState: State = {  
  email: "",  
  users: []  
};  
  
Action  
  
export const setEmail = createAction("Set Email", props<{ email: string }>());  
export const setUsers = createAction("Set Users", props<{ users: User[] }>());  
  
const userReducer = createReducer(  
  initialState,  
  on(setEmail, (state, { email }) => ({ ...state, email })),  
  on(setUsers, (state, { users }) => ({ ...state, users })),  
);  
  
export const reducers: ActionReducerMap<any> = {  
  user: function(state, action) {  
    return userReducer(state, action)  
  }  
};
```

Kiểu dữ liệu lưu trong Store

Giá trị ban đầu lưu trong Store

Reducer

Map Action vào Reducer

Ví dụ tạo Store lưu thông tin liên quan đến user

```

import { Component } from "@angular/core";
import { Store } from "@ngrx/store";
import { setEmail } from "src/app/reducers";

@Component({...
})
export class LoginComponent {
  constructor(private store: Store<any>) {}

  saveEmailToStore(email: string) {
    this.store.dispatch(setEmail({ email }));
  }
}

```

dispatch Action vào Store  
để cập nhật dữ liệu

```

import { Component, OnInit } from "@angular/core";
import { Store } from "@ngrx/store";
import { Subscription } from "rxjs";

@Component({...
})
export class DashboardComponent implements OnInit {
  storeSubscription: Subscription;

  constructor(private store: Store<any>) {}

  ngOnInit() {
    this.storeSubscription = this.store.select("user").subscribe(userState => {
      console.log(userState);
    });
  }

  ngOnDestroy() {
    this.storeSubscription.unsubscribe();
  }
}

```

Lấy dữ liệu từ Store

Unsubscribe khi  
Component bị hủy

Ví dụ lưu và lấy thông tin từ Store giữa các Component khác nhau

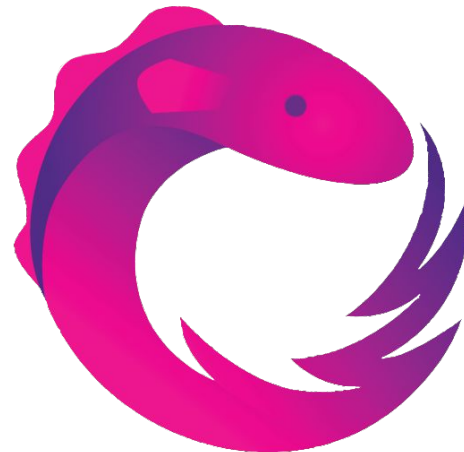


---

# Phụ lục

## RxJS

- React Extension Library cho Javascript:  
<https://rxjs-dev.firebaseapp.com/>
- Được cài đặt sẵn trong Angular, giúp lập trình theo hướng Reactive Programming dùng Observables để xử lý code asynchronous hoặc callback.
- Website học cách sử dụng RxJS: <https://www.learnrxjs.io/>.





## Awesome Angular

Những thư viện, bài viết, hướng dẫn, ... liên quan đến Angular:

<https://github.com/PatrickJS/awesome-angular>

