Tour of Heroes application and tutorial: <https://angular.io/tutorial/tour-of-heroes>

Create a new project

**Tạo Project**: ng new angular-tour-of-heroes

Chạy Project: cd angular-tour-of-heroes; ng serve --open

Truyền biến từ .ts => .html

app.component.ts: title = 'Tour of Heroes';

app.component.html: <h1>{{title}}</h1>

Trang trí src/styles.css

h1 {

  color: #369;

  font-family: Arial, Helvetica, sans-serif;

  font-size: 250%;

}

h2, h3 {

  color: #444;

  font-family: Arial, Helvetica, sans-serif;

  font-weight: lighter;

}

body {

  margin: 2em;

}

body, input[type="text"], button {

  color: #333;

  font-family: Cambria, Georgia, serif;

}

button {

  background-color: #eee;

  border: none;

  border-radius: 4px;

  cursor: pointer;

  color: black;

  font-size: 1.2rem;

  padding: 1rem;

  margin-right: 1rem;

  margin-bottom: 1rem;

  margin-top: 1rem;

}

button:hover {

  background-color: black;

  color: white;

}

button:disabled {

  background-color: #eee;

  color: #aaa;

  cursor: auto;

}

/\* everywhere else \*/

\* {

  font-family: Arial, Helvetica, sans-serif;

}

1. The hero editor

## Create the heroes component

ng generate component heroes

heroes.component.ts

export class AppRoutingModule {

  hero = 'Windstorm';

}

heroes.component.html

<h2>{{hero}}</h2>

## src/app/app.component.html

<h1>{{title}}</h1>

<app-heroes></app-heroes>

## Tạo interface: src/app/hero.ts:

export interface Hero {

    id: number;

    name: string;

  }

Sử dụng interface

src/app/heroes/heroes.component.ts:

import { Hero } from '../hero';

export class HeroesComponent implements OnInit {

  hero: Hero = { id: 1, name: 'Windstorm' };

## Show the hero object

## heroes.component.html (HeroesComponent template)

<h2>{{hero.name}} Details</h2>

<div><span>id: </span>{{hero.id}}</div>

<div><span>name: </span>{{hero.name}}</div>

## Format with the UppercasePipe

src/app/heroes/heroes.component.html:

<h2>{{hero.name|uppercase}} Details</h2>

## Edit the hero

### Import [FormsModule](https://angular.io/api/forms/FormsModule)

app.module.ts (FormsModule symbol import):

import { FormsModule } from '@angular/forms'; // <-- NgModel lives here

imports: [

        FormsModule

### **Two-way binding**

src/app/heroes/heroes.component.html (HeroesComponent's template)

<div><span>id: </span>{{hero.id}}</div>

<div>

  <label for="name">Hero name: </label>

  <input id="name" [(ngModel)]="hero.name" placeholder="name">

</div>

## Display a selection list

## Create mock heroes

src/app/mock-heroes.ts:

import { Hero } from './hero';

export const HEROES: Hero[] = [

  { id: 12, name: 'Dr. Nice' },

  { id: 13, name: 'Bombasto' },

  { id: 14, name: 'Celeritas' },

  { id: 15, name: 'Magneta' },

  { id: 16, name: 'RubberMan' },

  { id: 17, name: 'Dynama' },

  { id: 18, name: 'Dr. IQ' },

  { id: 19, name: 'Magma' },

  { id: 20, name: 'Tornado' }

];

## Displaying heroes:

## src/app/heroes/heroes.component.ts (import HEROES)

import { HEROES } from '../mock-heroes';

export class HeroesComponent implements OnInit {

   heroes = HEROES;

### **List heroes with \***[**ngFor**](https://angular.io/api/common/NgFor)

## heroes.component.html (heroes template)

<h2>My Heroes</h2>

<ul class="heroes">

  <li>

    <button type="button">

      <span class="badge">{{hero.id}}</span>

      <span class="name">{{hero.name}}</span>

    </button>

  </li>

</ul>

## Use <li \*ngFor="let hero of heroes">

<ul class="heroes">

  <li \*ngFor="let hero of heroes">

    <button type="button">

### **Add a click event binding:**

<button type="button" (click)="onSelect(hero)">

### **Add the click event handler**

src/app/heroes/heroes.component.ts :

heroes = HEROES;

  selectedHero?: Hero;

onSelect(hero: Hero): void {

  this.selectedHero = hero;

}

### **heroes.component.css**

/\* HeroesComponent's private CSS styles \*/

.heroes {

  margin: 0 0 2em 0;

  list-style-type: none;

  padding: 0;

  width: 15em;

}

.heroes li {

  display: flex;

}

.heroes button {

  flex: 1;

  cursor: pointer;

  position: relative;

  left: 0;

  background-color: #EEE;

  margin: .5em;

  padding: 0;

  border-radius: 4px;

  display: flex;

  align-items: stretch;

  height: 1.8em;

}

.heroes button:hover {

  color: #2c3a41;

  background-color: #e6e6e6;

  left: .1em;

}

.heroes button:active {

  background-color: #525252;

  color: #fafafa;

}

.heroes button.selected {

  background-color: black;

  color: white;

}

.heroes button.selected:hover {

  background-color: #505050;

  color: white;

}

.heroes button.selected:active {

  background-color: black;

  color: white;

}

.heroes .badge {

  display: inline-block;

  font-size: small;

  color: white;

  padding: 0.8em 0.7em 0 0.7em;

  background-color: #405061;

  line-height: 1em;

  margin-right: .8em;

  border-radius: 4px 0 0 4px;

}

.heroes .name {

  align-self: center;

}

### **Add a details section**

heroes.component.html (selected hero details): Bổ sung bên dưới

 </li>

</ul>

<div \*ngIf="selectedHero">

  <h2>{{selectedHero.name | uppercase}} Details</h2>

  <div>id: {{selectedHero.id}}</div>

  <div>

    <label for="hero-name">Hero name: </label>

    <input id="hero-name" [(ngModel)]="selectedHero.name" placeholder="name">

  </div>

</div>

  </div>

</div>

### **Style the selected hero**

heroes.component.html (toggle the 'selected' CSS class)

<li \*ngFor="let hero of heroes">

    <button type="button" (click)="onSelect(hero)" [class.selected]="hero === selectedHero">

      <span class="badge">{{hero.id}}</span>

1. Create a feature component

## Make the HeroDetailComponent

Tạo detail component: ng generate component hero-detail

### **Write the template** Cắt từ phần dưới của heroes.component.html sang hero-detail.component.html và sửa

<div \*ngIf="hero">

  <h2>{{hero.name | uppercase}} Details</h2>

  <div><span>id: </span>{{hero.id}}</div>

  <div>

    <label for="hero-name">Hero name: </label>

    <input id="hero-name" [(ngModel)]="hero.name" placeholder="name">

  </div>

</div>

### **Add the @**[**Input**](https://angular.io/api/core/Input)**() hero property**

hero-detail.component.ts

import { Component, Input, OnInit } from '@angular/core';

import { Hero } from '../hero';

## Show the HeroDetailComponent

### **Update the HeroesComponent template.** heroes.component.html:

</li>

</ul>

<app-hero-detail [hero]="selectedHero"></app-hero-detail>

1. Add services

## Create the HeroService: ng generate service hero

### **Get hero data.** src/app/hero.service.ts:

import { Hero } from './hero';

import { HEROES } from './mock-heroes';

export class HeroService {

  getHeroes(): Hero[] {

    return HEROES;

  }

## Update HeroesComponent. heroes.component.ts

import { Component, OnInit } from '@angular/core';

import { Hero } from '../hero';

import { HeroService } from '../hero.service';

export class HeroesComponent implements OnInit {

  heroes: Hero[]=[];

constructor(private heroService:HeroService) { }

  getHeroes(): void {

    this.heroes = this.heroService.getHeroes();

  }

  ngOnInit(): void {

    this.getHeroes()

  }

}

## Observable data

### **Observable HeroService.** src/app/hero.service.ts (Observable imports)

import { Observable, of } from 'rxjs';

getHeroes(): Observable<Hero[]> {

    const heroes = of(HEROES);

    return heroes;

  }

### **Subscribe in HeroesComponent.** heroes.component.ts

getHeroes(): void {

    this.heroService.getHeroes().subscribe(Heroes=>this.heroes=Heroes);

  }

## Show messages

### **Create MessagesComponent:** ng generate component messages

app.component.html

<app-heroes></app-heroes>

<app-messages></app-messages>

### **Create the MessageService:** ng generate service message

src/app/message.service.ts

export class MessageService {

  messages: string[] = [];

  add(message: string) {

    this.messages.push(message);

  }

  clear() {

    this.messages = [];

  }

  constructor() { }

}

### **Inject it into the HeroService**

### **Send a message from HeroService**

import { MessageService } from './message.service';

export class HeroService {

  constructor(private messageService: MessageService) { }

  getHeroes(): Observable<Hero[]> {

    const heroes = of(HEROES);

    this.messageService.add('HeroService: fetched heroes');

    return heroes; } }

### **Display the message from HeroService**

src/app/messages/messages.component.ts (import MessageService)

import { MessageService } from '../message.service';

constructor(public messageService: MessageService) { }

src/app/messages/messages.component.html

<div \*ngIf="messageService.messages.length">

  <h2>Messages</h2>

  <button type="button" class="clear"

          (click)="messageService.clear()">Clear messages</button>

  <div \*ngFor='let message of messageService.messages'> {{message}} </div>

</div>

## Add messages to hero service

src/app/heroes/heroes.component.ts

import { MessageService } from '../message.service';

constructor(private heroService:HeroService, private messageService: MessageService) { }

onSelect(hero: Hero): void {

  this.selectedHero = hero;

  this.messageService.add(`HeroesComponent: Selected hero id=${hero.id}`);

}

1. Add navigation with routing

## Add the AppRoutingModule[link](https://angular.io/tutorial/tour-of-heroes/toh-pt5#add-the-approutingmodule)

ng generate module app-routing --flat --module=app

app-routing.module.ts

import { HeroesComponent } from './heroes/heroes.component';

const routes: Routes = [

  { path: 'heroes', component: HeroesComponent }

];

## Add [RouterOutlet](https://angular.io/api/router/RouterOutlet)

src/app/app.component.html (router-outlet)

<h1>{{title}}</h1>

<router-outlet></router-outlet>

<app-messages></app-messages>

## Add a navigation link using [routerLink](https://angular.io/api/router/RouterLink)

<h1>{{title}}</h1>

<nav>

  <a routerLink="/heroes" style="text-decoration: none;">Heroes</a>

</nav>

## Add a dashboard view

ng generate component dashboard

dashboard.component.ts

import { Hero } from '../hero';

import { HeroService } from '../hero.service';

export class DashboardComponent implements OnInit {

  heroes: Hero[] = [];

  constructor(private heroService: HeroService) { }

  ngOnInit(): void {

    this.getHeroes();

  }

  getHeroes(): void {

    this.heroService.getHeroes()

      .subscribe(heroes => this.heroes = heroes.slice(1, 5));

  }

}

dashboard.component.html

<h2>Top Heroes</h2>

<div class="heroes-menu">

  <a \*ngFor="let hero of heroes">

    {{hero.name}}

  </a>

</div>

### **Add the dashboard route**

src/app/app-routing.module.ts (import DashboardComponent)

import { DashboardComponent } from './dashboard/dashboard.component';

const routes: Routes = [

  { path: 'heroes', component: HeroesComponent },

  { path: 'dashboard', component: DashboardComponent },

];

### **Add a default route**

{ path: '', redirectTo: '/dashboard', pathMatch: 'full' },

{ path: 'heroes', component: HeroesComponent },

### **Add dashboard link to the shell**

src/app/app.component.html

<nav>

  <a routerLink="/dashboard">Dashboard</a>

  <a routerLink="/heroes" style="text-decoration: none;">Heroes</a>

</nav>

## Navigating to hero details

### **Delete hero details from HeroesComponent**

Open the heroes/heroes.component.html and delete the <app-hero-detail> element from the bottom.

### **Add a hero detail route**

app-routing.module.ts

import { HeroDetailComponent } from './hero-detail/hero-detail.component';

{ path: 'dashboard', component: DashboardComponent },

{ path: 'detail/:id', component: HeroDetailComponent }

];

### **DashboardComponent hero links**

src/app/dashboard/dashboard.component.html (hero links)

### **HeroesComponent hero links**

<h2>My Heroes</h2>

<ul class="heroes">

  <li \*ngFor="let hero of heroes">

    <a routerLink="/detail/{{hero.id}}">

      <span class="badge">{{hero.id}}</span> {{hero.name}}

    </a>

  </li>

</ul>

#### **Remove dead code – optional**. heroes.component.ts:

export class HeroesComponent implements OnInit {

  heroes: Hero[]=[];

  selectedHero?: Hero;

  constructor(private heroService:HeroService, private messageService: MessageService) { }

  getHeroes(): void {

    this.heroService.getHeroes().subscribe(Heroes=>this.heroes=Heroes);

  }

  ngOnInit(): void {

    this.getHeroes()

  }

}

## Routable HeroDetailComponent. hero-detail.component.ts:

import { ActivatedRoute } from '@angular/router';

import { Location } from '@angular/common';

import { HeroService } from '../hero.service';

export class HeroDetailComponent implements OnInit {

constructor(private route: ActivatedRoute,

    private heroService: HeroService,

    private location: Location) { }

  ngOnInit(): void {

    this.getHero();

  }

  getHero(): void {

    const id = Number(this.route.snapshot.paramMap.get('id'));

    this.heroService.getHero(id)

      .subscribe(hero => this.hero = hero);

  }

}

### **Add HeroService.getHero()**[**link**](https://angular.io/tutorial/tour-of-heroes/toh-pt5#add-heroservicegethero)

src/app/hero.service.ts (getHero)

getHero(id: number): Observable<Hero> {

    // For now, assume that a hero with the specified `id` always exists.

    // Error handling will be added in the next step of the tutorial.

    const hero = HEROES.find(h => h.id === id)!;

    this.messageService.add(`HeroService: fetched hero id=${id}`);

    return of(hero);

  }

### **Find the way back**

hero-detail.component.html

  <input id="hero-name" [(ngModel)]="hero.name" placeholder="name">

  </div>

</div>

<button type="button" (click)="goBack()">go back</button>

hero-detail.component.ts

goBack(): void {

    this.location.back();

  }

1. Get data from a server

## Enable HTTP services

src/app/app.module.ts (HttpClientModule import)

import { HttpClientModule } from '@angular/common/http';

imports: [

    HttpClientModule

## Simulate a data server

Install the In-memory Web API package

npm install angular-in-memory-web-api –save

npm i angular-in-memory-web-api@0.14.0 --save-dev

src/app/app.module.ts (In-memory Web API imports)

import { MessagesComponent } from './messages/messages.component';

import { DashboardComponent } from './dashboard/dashboard.component';

imports: [

    HttpClientModule,

// The HttpClientInMemoryWebApiModule module intercepts HTTP requests

// and returns simulated server responses.

// Remove it when a real server is ready to receive requests.

HttpClientInMemoryWebApiModule.forRoot(

  InMemoryDataService, { dataEncapsulation: false })

  ],

ng generate service InMemoryData

## Heroes and HTTP