



HƯỚNG DẪN THỰC HÀNH ANGULAR 4.0 WORKSHOP

Mục tiêu

Trải nghiệm việc tạo một ứng dụng Angular hoàn chỉnh để hiểu rõ hơn về ứng dụng front-end, các công cụ lập trình front-end và các bước để phát triển một ứng dụng Angular 4.0.

Mô tả

Phát triển ứng dụng Zing Radio sử dụng công nghệ Angular phiên bản 4.0.

Ứng dụng Zing Radio này là phiên bản giản lược của ứng dụng Zing Radio chính thức của Zing, được thiết kế để người tham gia có thể hoàn thành được trong vòng 2 giờ mà vẫn trải nghiệm được đầy đủ vòng đời phát triển một ứng dụng front-end.

Ứng dụng Zing Radio có giao diện cơ bản như sau:



Hướng dẫn

Phần 1: Tạo ứng dụng và các cài đặt cần thiết

1. [Tạo thư mục của ứng dụng](#)
2. [Tìm hiểu cấu trúc thư mục của ứng dụng](#)
3. [Cài đặt Bootstrap để hỗ trợ tạo giao diện cho ứng dụng](#)
4. [Tải các file tài nguyên của ứng dụng \(hình ảnh, dữ liệu...\)](#)
5. [Nhúng các file css và javascript vào trong ứng dụng](#)

Phần 2: Tạo giao diện cho ứng dụng

1. [Tạo giao diện cơ bản cho trình chơi nhạc](#)
2. [Trang trí giao diện của trình chơi nhạc](#)
3. [Tạo giao diện cho playlist](#)
4. [Trang trí giao diện cho playlist](#)

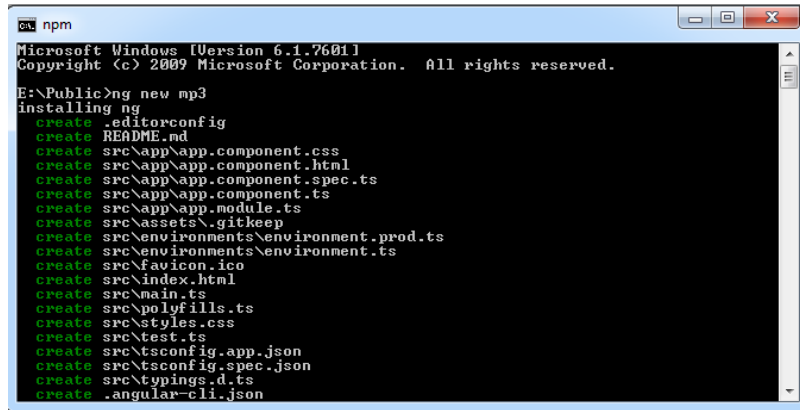
Phần 3: Triển khai các chức năng cho ứng dụng

1. [Tạo thành phần Player để chơi nhạc](#)
2. [Sử dụng thành phần PlayerComponent](#)
3. [Tạo thành phần để điều khiển playlist](#)
4. [Hoàn thiện ứng dụng](#)

Phần 1: Tạo ứng dụng và các cài đặt cần thiết

1. Tạo thư mục của ứng dụng

Mở cửa sổ `cmd` ở thư mục muốn tạo project sau đó gõ lệnh: **ng new mp3**



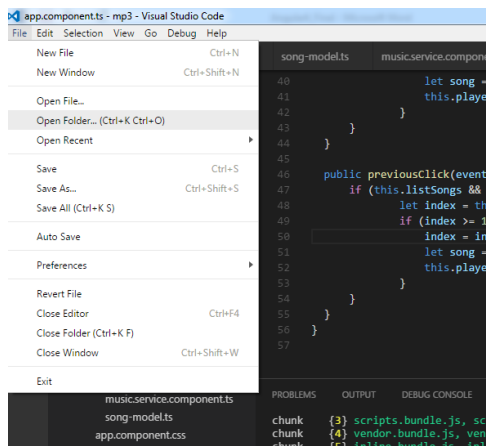
```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

E:\Public>ng new mp3
Installing ng
create .editorconfig
create README.md
create src\app\app.component.css
create src\app\app.component.html
create src\app\app.component.spec.ts
create src\app\app.component.ts
create src\app\app.module.ts
create src\assets\gitkeep
create src\environments\environment.prod.ts
create src\environments\environment.ts
create src\favicon.ico
create src\index.html
create src\main.ts
create src\polyfills.ts
create src\styles.css
create src\test.ts
create src\tscconfig.app.json
create src\tscconfig.spec.json
create src\typings.d.ts
create .angular-cli.json
```

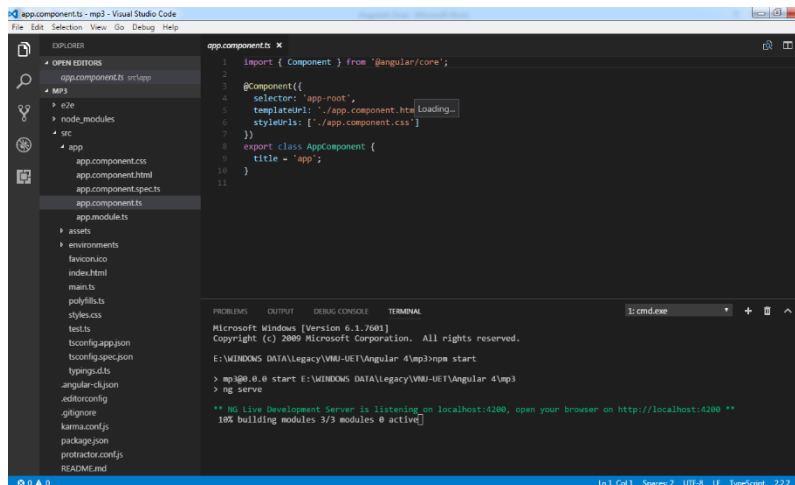
Chờ cho tới khi chúng ta cài đặt xong.

2. Tìm hiểu cấu trúc thư mục của ứng dụng

Chọn file và open folder



Chọn thư mục mp3 (Ứng dụng nghe nhạc trực tuyến mp3). Kết quả như hình vẽ bên dưới:



Cấu trúc ứng dụng Angular 4:



Files	Ý nghĩa
app/app.component.ts, app/app.component.css, app/app.component.html, app/app.component.spec.ts	Định nghĩa những AppComponent cùng với HTML template, CSS stylesheet và unit test. Nó sẽ là component root để phát triển ứng dụng của chúng ta sau này
app/app.module.ts	Khai báo AppModule. Đây là root module, là nơi ta sẽ khai báo những component mà ta sẽ dùng trong ứng dụng
assets/	Folder mà bạn để ảnh hoặc bất cứ thứ gì để copy trong ứng dụng của bạn. VD : fonts, css, js...
index.html	Khi client truy cập vào ứng dụng của bạn, thì tức là họ đang truy cập vào file này đấy. Phần lớn thời gian thì bạn sẽ không cần phải sửa file này. Angular CLI tự động thêm những file js, css khi mà ta build app ra bản product.
polyfills.ts	Support cho những browser để có thể chạy được ứng dụng
styles.css	File css của toàn ứng dụng

3. Cài đặt Bootstrap để hỗ trợ tạo giao diện cho ứng dụng

Mở cửa sổ terminal của visual studio code và gõ lệnh:

```
npm i bootstrap@next --save
```

```
chunk {0} polyfills.bundle.js, polyfills.bundle.js.map (polyfills) 177 kB {4} [initial]
chunk {1} main.bundle.js, main.bundle.js.map (main) 3.52 kB {3} [initial] [rendered]
chunk {2} styles.bundle.js, styles.bundle.js.map (styles) 10.5 kB {4} [initial]
chunk {3} vendor.bundle.js, vendor.bundle.js.map (vendor) 2.19 MB [initial]
chunk {4} inline.bundle.js, inline.bundle.js.map (inline) 0 bytes [entry]
webpack: Compiled successfully.
webpack: Compiling...
Hash: c81bae6be9bef1721b5d
Time: 470ms
chunk {0} polyfills.bundle.js, polyfills.bundle.js.map (polyfills) 177 kB {4} [initial]
chunk {1} main.bundle.js, main.bundle.js.map (main) 9.33 kB {3} [initial] [rendered]
chunk {2} styles.bundle.js, styles.bundle.js.map (styles) 10.5 kB {4} [initial]
chunk {3} vendor.bundle.js, vendor.bundle.js.map (vendor) 2.19 MB [initial]
chunk {4} inline.bundle.js, inline.bundle.js.map (inline) 0 bytes [entry]
webpack: Compiled successfully.
Terminate batch job (Y/N)? y
E:\WINDOWS DATA\Legacy\VNU-UET\Angular 4\mp3>npm i bootstrap@next --save
```

4. Tải các file tài nguyên của ứng dụng (hình ảnh, dữ liệu...)

Download file Codegym_Angular4.zip tại địa chỉ:

<https://drive.google.com/file/d/0B6hewmCvbSq7RC0zc20xYWVWTnM/view?usp=sharing>

Giải nén file vừa tạo và copy 3 folders bên dưới đây đưa vào thư mục **assets** của project mp3 chúng ta đã tạo từ bước đầu tiên:

data	8/8/2017 1:26 AM	File folder	
fonts	8/8/2017 1:13 AM	File folder	
img	8/8/2017 1:13 AM	File folder	
angular4-style.css	8/8/2017 1:14 AM	Text Document	12 KB

Copy nội dung từ file angular4-style.css từ thư mục **Codegym_Angular4** tải về vào file style.css

data	8/8/2017 1:13 AM	File folder	
fonts	8/8/2017 1:13 AM	File folder	
img	8/8/2017 1:13 AM	File folder	
angular4-style.css	8/8/2017 1:14 AM	Text Document	12 KB

5. Nhúng các file css và javascript vào trong ứng dụng

Tiếp theo chúng ta mở file angular-cli.json thêm vào **style** và **scripts**

```
"styles": [
  "../node_modules/bootstrap/dist/css/bootstrap.min.css",
  "../node_modules/bootstrap/dist/css/bootstrap.css",
  "styles.css"
],
"scripts": [
  "../node_modules/jquery/dist/jquery.js",
  "../node_modules/tether/dist/js/tether.js",
  "../node_modules/bootstrap/dist/js/bootstrap.js"
],
```

Phần 2: Tạo giao diện cho ứng dụng

1. Tạo giao diện cơ bản cho trình chơi nhạc

Click vào app.component.ts chúng ta sẽ thấy:

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'app';
}
```

Chọn app.component.html để tạo giao diện HTML cho ứng dụng.

Tạo khung giao diện:

```
<div class="container">
  <div class="col-sm-8 col-sm-offset-2">

  </div>
</div>
```

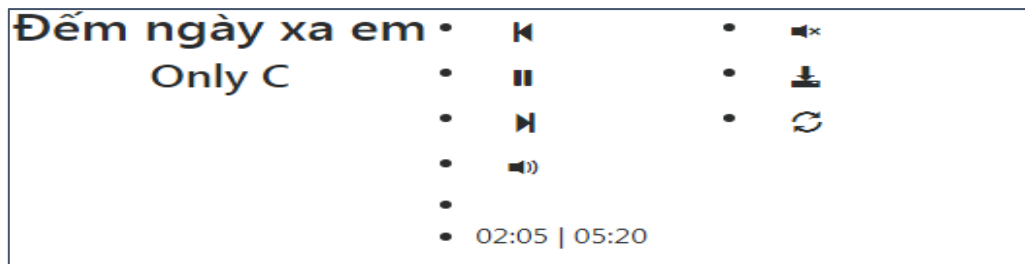
Tạo giao diện hiển thị chi tiết 1 bài hát bằng cách thêm mã nguồn sau bên trong thẻ div vừa tạo ở bước trên:

```

<div class="row allSong">
  <div class="col-sm-12 back-song">
    <div class="row">
      <div class="w100pt content content-song">
        <h3 class="text-center">Đếm ngày xa em</h3>
        <h4 class="text-center">Only C</h4>
      </div>
      <div class="w100pt timeline">
        <div class="timeOnline"></div>
      </div>
      <div class="w100pt allButton">
        <div class="row">
          <div class="col-sm-8">
            <div class="row">
              <ul>
                <li><span class="btn glyphicon
glyphicon-step-backward"></span></li>
                <li><span id="state" class="btn
glyphicon glyphicon-pause" (click)="processState($event.target)"></span></li>
                <li><span class="btn glyphicon
glyphicon-step-forward"></span></li>
                <li><span class="btn glyphicon
glyphicon-volume-up"></span></li>
                <li><span class="btn btn-
timevol"></span></li>
                <li><span class="time">02:05</span><span
class="time">05:20</span></li>
              </ul>
            </div>
          </div>
          <div class="col-sm-4">
            <div class="row">
              <ul>
                <li><span class="btn glyphicon
glyphicon-volume-off"></span></li>
                <li><span class="btn glyphicon
glyphicon-download-alt"></span></li>
                <li><span class="btn glyphicon
glyphicon-refresh"></span></li>
              </ul>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>

```

Kết quả:



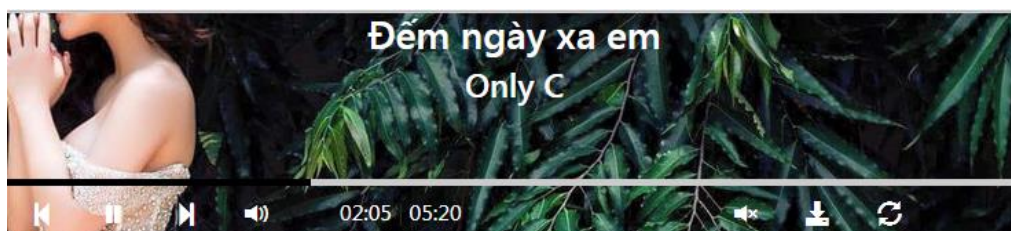
2. Trang trí giao diện của trình chơi nhạc

Chỉnh giao diện hiển thị chi tiết một bài hát bằng cách thêm đoạn mã nguồn bên dưới đây tiếp tục vào file style.css

```
.allSong{border: 1px solid #ccc;overflow: hidden;}
.back-song{background: url(assets/img/bg-song.jpg) no-repeat center center
transparent;overflow: hidden; height: 162px;}
ul li{ list-style: none;float: left; }
ul li span{ display: inline-block;}
.back-song .w100pt{ width: 100%;float: left;}
.back-song .content.content-song{ height: 120px;}
.back-song .timeline{ height: 5px;background: #ccc;}
.back-song .timeOnline{ height: 5px;background: #000;width: 30%;}
.content-song h3,
.content-song h4 { color: #fff;}

/**content song*/
.back-song ul li .glyphicon{ color: #fff; font-size: 18px;}
.back-song ul li span.time{ color: #fff;}
ul li span,
ul li span.time{ display: inline-block;}
ul li span.time{ margin-top: 7px;}
.back-song ul{ margin: 0px; padding: 0px 15px}
```

Kết quả:



3. Tạo giao diện cho playlist

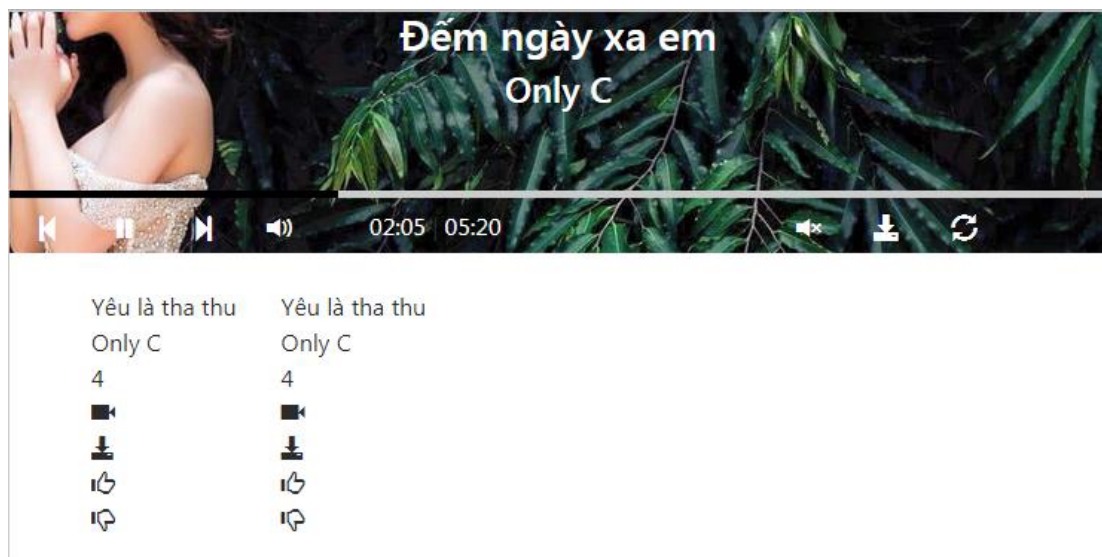
Tạo giao diện hiển thị danh sách các bài hát

```

<div class="col-sm-12">
  <div class="row scorllSong">
    <ul class="list list-song">
      <li>
        <div class="col col1 active"><span class="item-
song"></span></div>
        <div class="col col2">Yêu là tha thu</div>
        <div class="col col3">Only C</div>
        <div class="col col4">4</div>
        <div class="col col5"><span class="glyphicon glyphicon-
facetime-video"></span></div>
        <div class="col col6"><span class="glyphicon glyphicon-
download-alt"></span></div>
        <div class="col col7"><span class="glyphicon glyphicon-
thumbs-up"></span></div>
        <div class="col col8"><span class="glyphicon glyphicon-
thumbs-down"></span></div>
      </li>
    </ul>
  </div>
</div>

```

Kết quả sau khi thực hiện bước này:



4. Trang trí giao diện cho playlist

Chỉnh style hiển thị danh sách bài hát. Thêm đoạn css dưới đây tiếp tục vào file style.css


```

/**list**/
.list.list-song li{ width: 100%;float: left;overflow: hidden;display: inline-
block;}
.list-song .col{ overflow: hidden;height: 30px;float: left;}
ul.list-song{ margin: 0px; padding: 15px 15px}
ul.list-song .col{text-align: center}
ul.list-song .col1{ width: 5%;}
ul.list-song .col2{ width: 35%; text-align: left;}
ul.list-song .col3{ width: 35%; text-align: left;}
ul.list-song .col4{width: 5%}
ul.list-song .col5{width: 5%}
ul.list-song .col6{width: 5%}
ul.list-song .col7{width: 5%}
ul.list-song .col8{width: 5%}
.scorllSong{ overflow-y: scroll;height: 200px;}

.col1.active .item-song {
    content: "";
    display: block;
    width: 14px;
    height: 10px;
    background: url(assets/img/playing.png) left center;
    -webkit-animation: play .8s steps(3) infinite;
    animation: play .8s steps(3) infinite;
}

```

Kết quả:



Phần 3: Triển khai các chức năng cho ứng dụng

- **PlayerComponent:** Component play, stop, next, previous một audio
- **AppComponent:** Component chính của ứng dụng hiển thị danh sách các bài hát và chứa ứng dụng player

1. Tạo thành phần Player để chơi nhạc

Tạo thư mục music. Tiếp theo chúng ta tạo player component bao gồm 2 file: player.component.ts và player.component.html

File: player.component.ts:

```

import { Component, OnInit, ElementRef, ViewChild, Output, EventEmitter,
OnChanges } from '@angular/core';
import { Song } from "../service/index";

@Component({
  selector: 'player',
  templateUrl: './player.component.html'
})

export class PlayerComponent implements OnInit {
  @ViewChild('player') audio: ElementRef;

  public urlAudio: string;
  public isPlay: boolean = true;
  public currentAudio: Song = {
    id: "",
    name: "",
    url: "../assets/data/Dem-Ngay-Xa-Em-OnlyC-Lou-Hoang.mp3",
    singer: ""
  };

  @Output() nextEmit: EventEmitter<string> = new EventEmitter<string>();
  @Output() previousEmit: EventEmitter<string> = new EventEmitter<string>();

  public ngOnInit() {
  }

  public playAudio() {
    this.audio.nativeElement.load();
    this.audio.nativeElement.play();
  }

  public loadUrl(url: string) {
    this.urlAudio = url;
  }

  public previousAction() {
    this.previousEmit.emit(this.currentAudio.id);
  }

  public nextAction() {
    this.nextEmit.emit(this.currentAudio.id);
  }
}

```

File: player.component.html:

Cắt đoạn back-song trong app.comonent.html đưa vào player.component.html

Bổ sung thêm đoạn mã nguồn:

```
<audio #player>
  <source type="audio/mpeg" src="{{urlAudio}}">
</audio>
```

```
<div class="col-sm-12 back-song">
  <audio #player>
    <source type="audio/mpeg" src="{{urlAudio}}">
  </audio>
  <div class="row">
    <div class="w100pt content content-song">
      <h3 class="text-center">Đếm ngày xa em</h3>
      <h4 class="text-center">Only C-Lou-Hoang</h4>
    </div>
    <div class="w100pt timeline">
      <div class="timeOnline"></div>
    </div>
    <div class="w100pt allButton">
      <div class="row">
        <div class="col-sm-8">
          <div class="row">
            <ul>
              <li><span class="btn glyphicon glyphicon-step-backward"></span></li>
              <li><span class="btn glyphicon glyphicon-pause"></span></li>
              <li><span class="btn glyphicon glyphicon-step-forward"></span></li>
              <li><span class="btn glyphicon glyphicon-volume-up"></span></li>
              <li><span class="btn btn-timevol"></span></li>
              <li>
                <span class="time">02:05</span> |
                <span class="time">05:20</span>
              </li>
            </ul>
          </div>
        </div>
        <div class="col-sm-4">
          <div class="row">
            <ul>
              <li><span class="btn glyphicon glyphicon-volume-off"></span></li>
              <li><span class="btn glyphicon glyphicon-download-alt"></span></li>
              <li><span class="btn glyphicon glyphicon-refresh"></span></li>
```

```

        </ul>
      </div>
    </div>
  </div>
</div>

```

Xử lý sự kiện play hoặc pause cho ứng dụng trong file player.component.ts

Thêm chức năng hiển thị play hoặc pause Icon:

```

public setPlayIcon() {
  let control = document.getElementById("state");
  control.classList.remove("glyphicon-play");
  control.classList.add("glyphicon-pause");
}

public setPauseIcon() {
  let control = document.getElementById("state");
  control.classList.remove("glyphicon-pause");
  control.classList.add("glyphicon-play");
}

```

Thêm processState mã nguồn như bên dưới để stop hoặc play 1 audio:

```

public processState(event: any) {
  let control = event;
  if (this.isPlaying) {
    this.audio.nativeElement.pause();
    this.isPlaying = false;
    this.setPauseIcon();
  } else {
    this.audio.nativeElement.play();
    this.isPlaying = true;
    this.setPlayIcon();
  }
}

```

Sau đó khai báo sự kiện tại player.component.html:

```

<li><span id="state" class="btn glyphicon glyphicon-pause"
(click)="processState($event.target)"></span></li>

```

Thêm sự kiện nextAction:

```

<li><span class="btn glyphicon glyphicon-step-forward"
(click)="nextAction()"></span></li>

```

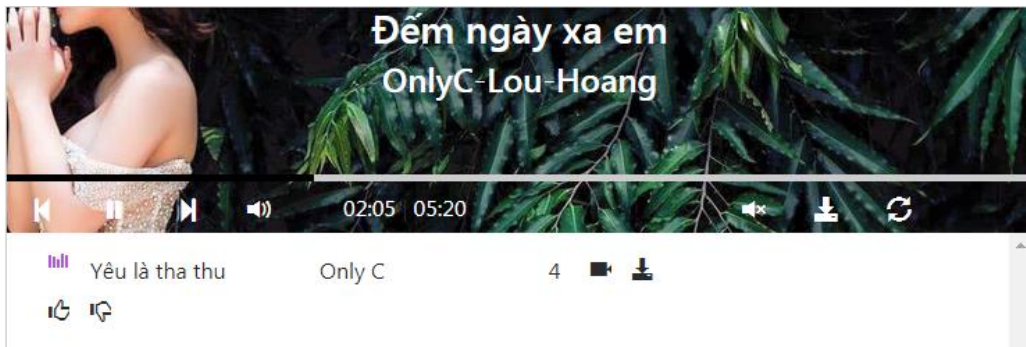
Thêm sự kiện previousAction:

```

<li><span class="btn glyphicon glyphicon-step-backward"
(click)="previousAction()"></span></li>

```

Kết quả tại bước này:



2. Sử dụng thành phần `PlayerComponent`

Mở file `app.component.ts` và import `PlayerComponent` như mã nguồn bên dưới:

```
import { Component, OnInit, ViewChild } from '@angular/core';
import { PlayerComponent } from "../music/player.component";

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent implements OnInit {
  @ViewChild(PlayerComponent) playerComponent: PlayerComponent;

  public ngOnInit() {

  }
}
```

Đăng ký `PlayerComponent` với `AppModule`:

- Mở file `app.module.ts`
- Đăng ký với module như hình ảnh bên dưới:

```
import { PlayerComponent } from "../music/player.component";

@NgModule({
  declarations: [
    AppComponent,
    PlayerComponent
```

Mở file `app.component.html`

Khai báo `<player>` `</player>` selector thay thế cho phần giao diện chúng ta đã đưa vào `player.component.html`.

Mã nguồn bên dưới:

```

<div class="container">
  <div class="col-sm-8 col-sm-offset-2">
    <div class="row allSong">
      <div style="width:100%;">
        <player></player>
      </div>
      <div class="col-sm-12">
        <div class="row scorllSong">
          <ul class="list list-song">
            <li>
              <div class="col col1 active"><span class="item-
song"></span></div>

              <div class="col col2">Yêu là tha thu</div>
              <div class="col col3">Only C</div>
              <div class="col col4">4</div>
              <div class="col col5">
                <span class="glyphicon glyphicon-facetime-
video"></span>

              </div>
              <div class="col col6">
                <span class="glyphicon glyphicon-download-
alt"></span>

              </div>
              <div class="col col7">
                <span class="glyphicon glyphicon-thumbs-
up"></span>

              </div>
              <div class="col col8">
                <span class="glyphicon glyphicon-thumbs-
down"></span>

              </div>
            </li>
            <li>
              <div class="col col1 active"></div>
              <div class="col col2">Đếm ngày xa em</div>
              <div class="col col3">Robin Hoàng Sơn</div>
              <div class="col col4">4</div>
              <div class="col col5">
                <span class="glyphicon glyphicon-facetime-
video"></span>

              </div>
              <div class="col col6">
                <span class="glyphicon glyphicon-download-
alt"></span>

              </div>
              <div class="col col7">
                <span class="glyphicon glyphicon-thumbs-
up"></span>

```

```

        </div>
        <div class="col col8">
            <span class="glyphicon glyphicon-thumbs-
down"></span>

        </div>
    </li>
</ul>
</div>
</div>
</div>
</div>
</div>
</div>

```

3. Tạo thành phần để điều khiển playlist

Hiện tại danh sách các bài hát đang được gán bằng các giá trị mặc định. Chúng ta sẽ hiển thị danh sách các bài hát từ server. Đầu tiên chúng ta tạo thư mục service trong ứng dụng.

Tạo Model trong thư mục service;

```

export interface Song {
    id: string,
    name: string,
    url: string,
    singer: string
}

```

Tạo MusicService class:

```

import { Injectable } from "@angular/core";
import { Observable } from 'rxjs/Observable';
import { Http } from '@angular/http';
import 'rxjs/add/operator/map';
import { Song } from "../song-model";

@Injectable()
export class MusicService {
    private _http: Http;
    private _baseUrl: string = "../../assets/data/data.json";
    constructor(private http: Http) {
        this._http = http;
    }

    public getListSong(): Observable<Song[]> {
        return this.http
            .get(this._baseUrl)
            .map((response) => response.json() as Song[]);
    }
}

```


Tạo index.ts trong thư mục service để export song, music service

```
export * from "./song-model";  
export * from "./music.service.component";
```

Đăng ký MusicService với app.module.ts

Mã nguồn hoàn chỉnh bên dưới:

```
import { BrowserModule } from '@angular/platform-browser';  
import { NgModule } from '@angular/core';  
  
import { AppComponent } from './app.component';  
import { PlayerComponent } from './music/player.component';  
import { MusicService } from './service/index';  
import { HttpClientModule } from '@angular/http';  
  
@NgModule({  
  declarations: [  
    AppComponent,  
    PlayerComponent  
  ],  
  imports: [  
    BrowserModule,  
    HttpClientModule  
  ],  
  providers: [MusicService],  
  bootstrap: [AppComponent]  
})  
export class AppModule { }
```

Gọi MusicService trong AppComponent và lấy danh sách bài hát

Trước tiên chúng ta import Song, MusicService trong AppComponent:

```
import { MusicService, Song } from './service/index';
```

Khai báo 1 biến private _service và danh sách bài hát:

```
public listSongs: Song[] = [];  
private _service: MusicService;
```

Khởi tạo MusicService:

```
constructor(service: MusicService) {  
  this._service = service;  
}
```

Tại hàm ngOnInit chúng ta sẽ lấy dữ liệu và hiển thị danh sách:

```

public ngOnInit() {
  this._service.getListSong().subscribe((response)=> {
    this.listSongs = response;
  });
}

```

Mở file app.component.html chúng ta sẽ thay thế các bài hát đã được đặt mặc định giá trị thay bằng *ngFor của angular 4 như mã nguồn bên dưới:

```

<ul class="list list-song">
  <li *ngFor="let item of listSongs">
    <div class="col col1 active"><span class="itemsong"></span></div>
    <div class="col col2">{{item.name}}</div>
    <div class="col col3">{{item.singer}}</div>
    <div class="col col4">4</div>
    <div class="col col5">
      <span class="glyphicon glyphicon-facetime-video"></span>
    </div>
    <div class="col col6">
      <span class="glyphicon glyphicon-download-alt"></span>
    </div>
    <div class="col col7">
      <span class="glyphicon glyphicon-thumbs-up"></span>
    </div>
    <div class="col col8">
      <span class="glyphicon glyphicon-thumbs-down"></span>
    </div>
  </li>
</ul>

```

Kết quả tại bước này:



4. Hoàn thiện ứng dụng

Thiết lập bài hát mặc định khi tải ứng dụng lần đầu và sự kiện click vào bài hát trong danh sách sẽ hiển thị và play song.

Bước 1: Chọn player.component.ts và tạo method setAudio

+ Khai báo biến currentAudio

```
public currentAudio: Song;
```

+ Tạo method: setAudio

```
public setAudio(audio: Song) {  
    this.currentAudio = audio;  
    this.urlAudio = audio.url;  
    this.playAudio();  
    this.setPlayIcon();  
}
```

+ Update lại giao diện player.component.html

```
<div class="w100pt content content-song">  
    <h3 class="text-center">{{currentAudio.name}}</h3>  
    <h4 class="text-center">{{currentAudio.singer}}</h4>  
</div>
```

Bước 2: Update lại hàm ngOnInit của app.component.ts để lấy bài hát mặc định

```
public ngOnInit() {  
    this._service.getListSong().subscribe((response)=> {  
        this.listSongs = response;  
        if (this.listSongs && this.listSongs.length > 0) {  
            let defaultSong = this.listSongs[0];  
            this.playerComponent.setAudio(defaultSong);  
        }  
    });  
}
```

Bước 3: Click một bài hát tại danh sách sẽ chạy bài hát đó

+ Tạo sự kiện click tại tên mỗi bài hát ở app.component.html

```
<div class="col col2" (click)="selectSong(item)" >{{item.name}}</div>
```

Với tên sự kiện là selectSong sẽ truyền đầu vào là một item bài hát

+ Tạo hàm xử lý sự kiện selectSong

```
public selectSong(item: Song) {  
    this.playerComponent.setAudio(item);  
}
```

Kết quả tại bước này:

Khi chọn một bài hát thì bài hát đó sẽ được hát

Khi load ứng dụng lần đầu sẽ lấy bài hát đầu tiên của danh sách để chạy.

Xử lý sự kiện next, previous bài hát

Bước 1: Xử lý sự kiện Next tại AppComponent

Tạo hàm nhận id bài hát hiện tại đang play trong player component và chuyển bài tiếp theo

+ Tại file app.component.html chúng ta cập nhật lại đoạn mã nguồn dưới đây:

```
<player (nextEmit)="clickNext($event)"></player>
```

+ Tại file app.component.ts chúng ta thêm đoạn mã nguồn dưới đây:

```
public clickNext(event: any) {
    if (this.listSongs && this.listSongs.length > 1) {
        let index = this.listSongs.findIndex(x=>x.id == event);
        if (index >= 0 && (index < (this.listSongs.length - 1))) {
            index = index + 1;
            let song = this.listSongs[index];
            this.playerComponent.setAudio(song);
        }
    }
}
```

Bước 2: Xử lý sự kiện Previous tại AppComponent

Tạo hàm nhận id bài hát hiện tại đang play trong player component và chuyển bài trước đó

+ Tại file app.component.html chúng ta cập nhật lại đoạn mã nguồn dưới đây:

```
<player (nextEmit)="clickNext($event)"
(previousEmit)="previousClick($event)"></player>
```

+ Tại file app.component.ts chúng ta thêm đoạn mã nguồn dưới đây:

```
public previousClick(event: any) {
    if (this.listSongs && this.listSongs.length > 1) {
        let index = this.listSongs.findIndex(x=>x.id == event);
        if (index >= 1) {
            index = index - 1;
            let song = this.listSongs[index];
            this.playerComponent.setAudio(song);
        }
    }
}
```

Và đây là kết quả cuối cùng:

