

ChatGPT

Bạn đóng vai trò là một mentor chuyên nghiệp để dạy tôi Angular từ cơ bản đến nâng cao và luôn luôn diễn giải chi tiết cặn kẽ và code đầy đủ các bước để chạy được ví dụ. Bạn đã dạy tôi các bài học sau đây: Bài 1: Giới thiệu Angular và Cài đặt Môi trường Bài 2: Data Binding trong Angular Bài 3: Component Communication trong Angular Bài 4: Angular Routing (Điều hướng giữa các Component) Bài 5: Reactive Forms trong Angular Bài 6: Gọi API trong Angular với HttpClient Bài 7: Quản lý trạng thái trong Angular với NgRx (State Management) Bài 8: Angular Services và Dependency Injection Bài 9: Lifecycle Hooks trong Angular Bài 10: Custom Directives trong Angular Bài 11: Custom Pipes trong Angular Bài 12: Angular Guards - Bảo vệ Route trong Angular Hãy tiếp tục lộ trình trên.

Bài 13: Lazy Loading Modules trong Angular

Mục tiêu bài học:

- Hiểu về Lazy Loading trong Angular.
- Biết cách cấu trúc ứng dụng để tối ưu hóa hiệu suất.
- Triển khai Lazy Loading bằng cách sử dụng `loadChildren`.

1. Lazy Loading là gì?

Lazy Loading (Tải chậm) là kỹ thuật trong Angular giúp tải module chỉ khi cần thiết, thay vì tải tất cả các module ngay từ đầu. Điều này giúp cải thiện hiệu suất ứng dụng bằng cách giảm kích thước của file JavaScript ban đầu.

2. Tại sao cần sử dụng Lazy Loading?

- Giúp giảm thời gian tải ban đầu của ứng dụng.
 - Chỉ tải module khi cần, tiết kiệm tài nguyên trình duyệt.
 - Cải thiện trải nghiệm người dùng (UX) bằng cách giảm độ trễ khi tải trang.
-

3. Triển khai Lazy Loading trong Angular

3.1 Tạo một dự án Angular mới (Nếu chưa có)

```
sh
```

```
ng new angular-lazy-loading
cd angular-lazy-loading
ng serve
```

3.2 Tạo hai module để thử nghiệm

```
sh
```

```
ng generate module modules/admin --route admin --module app.module
ng generate module modules/user --route user --module app.module
```

Lệnh trên sẽ tự động cập nhật `app-routing.module.ts` với các route và thiết lập Lazy Loading.

3.3 Cấu trúc thư mục

pgsql

```
src
├── app
│   ├── modules
│   │   ├── admin
│   │   │   ├── admin-routing.module.ts
│   │   │   ├── admin.module.ts
│   │   │   └── admin.component.ts
│   │   ├── user
│   │   │   ├── user-routing.module.ts
│   │   │   ├── user.module.ts
│   │   │   └── user.component.ts
│   ├── app-routing.module.ts
│   ├── app.module.ts
│   └── app.component.ts
```

3.4 Cấu hình Lazy Loading trong **app-routing.module.ts**

Lazy Loading sử dụng **loadChildren**, giúp tải module khi cần thiết.

Cập nhật **app-routing.module.ts:**

typescript

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';

const routes: Routes = [
  { path: 'admin', loadChildren: () =>
import('./modules/admin/admin.module').then(m => m.AdminModule) },
  { path: 'user', loadChildren: () =>
import('./modules/user/user.module').then(m => m.UserModule) },
  { path: '', redirectTo: '/admin', pathMatch: 'full' }, // Mặc định điều
hướng đến trang admin
];

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

3.5 Cấu hình Admin Module (**admin.module.ts**)

typescript

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { AdminComponent } from './admin.component';
import { AdminRoutingModule } from './admin-routing.module';

@NgModule({
  declarations: [AdminComponent],
  imports: [
    CommonModule,
    AdminRoutingModule
  ]
})
export class AdminModule { }
```

3.6 Cấu hình Admin Routing (**admin-routing.module.ts**)

typescript

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { AdminComponent } from './admin.component';

const routes: Routes = [
  { path: '', component: AdminComponent }
];

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

3.7 Tạo và cấu hình User Module (**user.module.ts**)

typescript

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { UserComponent } from './user.component';
import { UserRoutingModule } from './user-routing.module';

@NgModule({
  declarations: [UserComponent],
  imports: [
    CommonModule,
    UserRoutingModule
  ]
})
export class UserModule { }
```

3.8 Tạo và cấu hình User Routing (**user-routing.module.ts**)

typescript

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { UserComponent } from './user.component';

const routes: Routes = [
  { path: '', component: UserComponent }
];

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

3.9 Tạo Admin Component (`admin.component.ts`)

typescript

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

@Component({
  selector: 'app-admin',
  template: `<h2>Admin Page</h2>`,
})
export class AdminComponent { }
```

3.10 Tạo User Component (`user.component.ts`)

typescript

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

@Component({
  selector: 'app-user',
  template: `<h2>User Page</h2>`,
})
export class UserComponent { }
```

4. Chạy ứng dụng và kiểm tra Lazy Loading

Bây giờ, chạy ứng dụng bằng:

sh

ng serve

- Truy cập `http://localhost:4200/admin` sẽ tải module `admin`.
- Truy cập `http://localhost:4200/user` sẽ tải module `user`.

📌 Kiểm tra Lazy Loading bằng Developer Tools

- Mở trình duyệt, vào **DevTools (F12)** → Tab **Network** → Chọn **JS**.
- Khi vào `/admin`, một file JavaScript riêng cho Admin Module sẽ được tải về.
- Khi vào `/user`, một file riêng cho User Module sẽ được tải về.

5. Kết luận

- Lazy Loading giúp tối ưu hóa hiệu suất ứng dụng Angular bằng cách tải module chỉ khi cần thiết.
- Sử dụng `loadChildren` trong `app-routing.module.ts` để thực hiện Lazy Loading.
- Tạo Routing riêng cho từng module để quản lý route tốt hơn.

👉 **Bài tiếp theo:** Học về **Angular Interceptors** để kiểm soát các request HTTP như thêm Token, Logging, và xử lý lỗi! 🚀

1/2

dùng standalone component thì áp dụng cơ chế trên như thế nào

🚀 Lazy Loading Modules với Standalone Components trong Angular

Từ Angular 14 trở đi, chúng ta có thể sử dụng **Standalone Components** mà không cần Module (`NgModule`). Trong bài này, mình sẽ hướng dẫn cách áp dụng **Lazy Loading** với Standalone Components.

📌 1. Tổng quan về Lazy Loading với Standalone Components

- Không cần tạo Module (`NgModule`), thay vào đó ta tạo Standalone Component.
- Sử dụng `loadComponent` thay vì `loadChildren` trong Router.

- **Tối ưu hóa hiệu suất**, giúp giảm kích thước ứng dụng ban đầu.

2. Cài đặt một ứng dụng Angular mới

Nếu bạn chưa có dự án Angular, tạo một dự án mới với Standalone Components:

```
sh
```

```
ng new angular-standalone-lazy --standalone
cd angular-standalone-lazy
ng serve
```

3. Cấu trúc thư mục

```
pgsql
```

```
src
├── app
│   ├── admin
│   │   └── admin.component.ts
│   ├── user
│   │   └── user.component.ts
│   ├── app.routes.ts
│   └── app.component.ts
```

4. Tạo Standalone Components

4.1 Tạo Admin Component

```
sh
```

```
ng generate component admin --standalone
```

👉 Cập nhật `admin.component.ts`:

```
typescript
```

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

@Component({
  standalone: true,
  selector: 'app-admin',
  template: `<h2>Admin Page (Standalone)</h2>`,
})
export class AdminComponent {}
```

4.2 Tạo User Component

```
sh
```

```
ng generate component user --standalone
```

👉 Cập nhật `user.component.ts`:

typescript

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

@Component({
  standalone: true,
  selector: 'app-user',
  template: `<h2>User Page (Standalone)</h2>`,
})
export class UserComponent {}
```

5. Cấu hình Lazy Loading trong Router

Trong `app.routes.ts`, sử dụng `loadComponent` để Lazy Load các Standalone Components:

👉 Cập nhật `app.routes.ts`:

typescript

```
import { Routes } from '@angular/router';

export const routes: Routes = [
  { path: 'admin', loadComponent: () =>
import('./admin/admin.component').then(m => m.AdminComponent) },
  { path: 'user', loadComponent: () =>
import('./user/user.component').then(m => m.UserComponent) },
  { path: '', redirectTo: '/admin', pathMatch: 'full' }, // Điều hướng mặc
định
];
```

6. Cấu hình `app.component.ts` để sử dụng Router

👉 Cập nhật `app.component.ts`:

typescript

```
import { Component } from '@angular/core';
import { RouterModule } from '@angular/router';
import { routes } from './app.routes';

@Component({
  standalone: true,
  selector: 'app-root',
  template: `
    <h1>Angular Lazy Loading với Standalone Components</h1>
    <nav>
      <a routerLink="/admin">Admin</a> |
      <a routerLink="/user">User</a>
    </nav>
    <router-outlet></router-outlet>
  `,
  imports: [RouterModule.forRoot(routes)]
})
export class AppComponent {}
```

✓ 7. Chạy ứng dụng và kiểm tra Lazy Loading

sh

ng serve

Kiểm tra trên trình duyệt:

- Truy cập <http://localhost:4200/admin> → Admin Component được tải.
- Truy cập <http://localhost:4200/user> → User Component được tải.
- Mở **DevTools (F12)** → **Network** → **Tab JS**:
 - Khi vào [/admin](#), file **admin.component.js** được tải.
 - Khi vào [/user](#), file **user.component.js** được tải.

👉 Xác nhận rằng Angular chỉ tải Component khi cần thiết!

🔥 8. Kết luận

- Standalone Components hỗ trợ Lazy Loading mà không cần Module (**NgModule**).
- Dùng **loadComponent** thay vì **loadChildren** để tối ưu hóa việc tải Component.
- Ứng dụng tải nhanh hơn và có hiệu suất tốt hơn.

🎯 **Bài tiếp theo:** Học về **Angular Interceptors** để kiểm soát request HTTP như thêm Token, Logging, và xử lý lỗi! 🚀

ChatGPT can make mistakes. Check important info.