

ChatGPT

Chào bạn, tôi đang theo lộ trình học Angular từ cơ bản tới nâng cao. Bạn đã dạy tôi các bài 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 Hãy tiếp tục lộ trình trên.

Bài tiếp theo trong lộ trình học Angular của bạn sẽ là:

Bài 11: Custom Pipes trong Angular

Mục tiêu: Hiểu và sử dụng **Pipes** để biến đổi dữ liệu trong Angular, cũng như cách tạo **Custom Pipes** theo nhu cầu riêng.

1. Pipes là gì?

Pipes trong Angular là công cụ giúp bạn chuyển đổi hoặc định dạng dữ liệu trong template một cách dễ dàng. Chúng thường được sử dụng với `{{ expression | pipeName }}`.

Ví dụ:

html

```
<p>{{ 1234567.89 | number }}</p> <!-- Kết quả: 1,234,567.89 -->
<p>{{ 'hello world' | uppercase }}</p> <!-- Kết quả: HELLO WORLD -->
<p>{{ today | date: 'dd/MM/yyyy' }}</p> <!-- Kết quả: 16/03/2025 -->
```

Một số **built-in Pipes** phổ biến:

- **uppercase**: Chuyển đổi chuỗi thành chữ hoa.
- **lowercase**: Chuyển đổi chuỗi thành chữ thường.
- **currency**: Định dạng số thành đơn vị tiền tệ.
- **date**: Định dạng ngày tháng.
- **percent**: Hiển thị số theo định dạng phần trăm.
- **json**: Chuyển đổi đối tượng thành chuỗi JSON.

2. Tạo Custom Pipe

Mặc định, Angular có sẵn một số Pipes nhưng đôi khi bạn cần **tạo một Pipe tùy chỉnh** cho các trường hợp đặc biệt.

Bước 1: Tạo Pipe mới

Bạn có thể tạo Pipe bằng Angular CLI:

```
bash
```

```
ng generate pipe custom
```

Hoặc rút gọn:

```
bash
```

```
ng g p custom
```

Lệnh này sẽ tạo ra một file `custom.pipe.ts` với nội dung mặc định.

Bước 2: Code Pipe tùy chỉnh

Giả sử chúng ta cần một Pipe để viết hoa chữ cái đầu của mỗi từ trong chuỗi.

custom.pipe.ts:

```
typescript
```

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'capitalizeWords'
})
export class CapitalizeWordsPipe implements PipeTransform {
  transform(value: string): string {
    if (!value) return '';
    return value.replace(/\b\w/g, char => char.toUpperCase());
  }
}
```

Giải thích:

- `@Pipe({ name: 'capitalizeWords' })`: Định nghĩa một Pipe có tên là `capitalizeWords`.
- `implements PipeTransform`: Lớp phải triển khai interface `PipeTransform` để dùng trong Angular.
- `transform(value: string)`: Đây là phương thức xử lý dữ liệu.

Bước 3: Sử dụng Pipe trong Template

Sau khi tạo xong, bạn có thể sử dụng trong HTML như sau:

html

```
<p>{{ 'hello angular world' | capitalizewords }}</p>
```

Kết quả: Hello Angular World

Bước 4: Đăng ký Pipe trong Module

Nếu bạn dùng Pipe trong **toàn bộ ứng dụng**, hãy thêm nó vào `app.module.ts`:

typescript

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { CapitalizeWordsPipe } from './capitalize-words.pipe';

@NgModule({
  declarations: [
    AppComponent,
    CapitalizeWordsPipe // Đăng ký Pipe ở đây
  ],
  imports: [BrowserModule],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Nếu bạn dùng Pipe trong một **Module riêng lẻ**, chỉ cần khai báo trong `declarations` của module đó.

3. Pipe có tham số

Bạn có thể truyền tham số vào Pipe bằng cách thêm đối số vào `transform()`.

Ví dụ: Tạo một Pipe để cắt chuỗi với độ dài tùy chỉnh.

typescript

```
@Pipe({
  name: 'truncate'
})
export class TruncatePipe implements PipeTransform {
  transform(value: string, limit: number = 10): string {
    return value.length > limit ? value.substring(0, limit) + '...' : value;
  }
}
```

Sử dụng:

html

```
<p>{{ 'This is a long text' | truncate:8 }}</p>
```

Kết quả: This is...

Tóm tắt

- ✓ Pipes giúp chuyển đổi dữ liệu trong template.
 - ✓ Có nhiều built-in Pipes trong Angular như **uppercase**, **date**, **currency**, ...
 - ✓ Bạn có thể tạo **Custom Pipes** để xử lý dữ liệu theo nhu cầu riêng.
 - ✓ Pipes có thể nhận tham số để xử lý dữ liệu linh hoạt hơn.
-

Bài tập thực hành

- 1 Tạo một Pipe có tên **reverse** để đảo ngược chuỗi (**hello** -> **olleh**).
- 2 Tạo một Pipe có tên **maskEmail** để ẩn bớt email (**test@example.com** -> **t***@example.com**).

3 Thử tạo một Pipe nhận tham số để chuyển đổi số giây thành định dạng `mm:ss` (`125 -> 02:05`).

Bạn thử làm bài tập trên rồi gửi lại mình kiểm tra nhé! 🚀

tôi dùng standalone component

Vì bạn sử dụng **standalone components** trong Angular, nên cách đăng ký **Custom Pipes** sẽ hơi khác một chút so với cách làm trong module truyền thống. Dưới đây là hướng dẫn phù hợp với cách tiếp cận **standalone components**.

1. Tạo Custom Pipe trong Standalone Component

Bước 1: Tạo Pipe

Sử dụng Angular CLI để tạo một Pipe mới:

```
bash
```

```
ng g p capitalize-words --standalone
```

Lệnh này sẽ tạo ra một Pipe **standalone** không phụ thuộc vào module.

Bước 2: Viết Logic cho Pipe

Trong file `capitalize-words.pipe.ts`, cập nhật code như sau:

typescript

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'capitalizeWords',
  standalone: true // Đánh dấu Pipe là standalone
})
export class CapitalizeWordsPipe implements PipeTransform {
  transform(value: string): string {
    if (!value) return '';
    return value.replace(/\b\w/g, char => char.toUpperCase());
  }
}
```

Lưu ý rằng:

standalone: true giúp Pipe này có thể được sử dụng độc lập mà không cần đăng ký trong module.

Bước 3: Sử dụng Pipe trong Standalone Component

Trong component standalone của bạn (ví dụ `app.component.ts`):

typescript

```
import { Component } from '@angular/core';
import { CapitalizeWordsPipe } from '../capitalize-words.pipe'; // Import
Pipe vào component

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [CapitalizeWordsPipe], // Đăng ký Pipe ở đây
  template: `<p>{{ 'hello angular world' | capitalizeWords }}</p>`
})
export class AppComponent {}
```

Điểm quan trọng:

- Bạn không cần khai báo Pipe trong **declarations** nữa.
- Thay vào đó, bạn import trực tiếp Pipe vào **imports** của component.

2. Pipe có tham số trong Standalone Component

Nếu bạn muốn tạo Pipe có tham số, ví dụ một Pipe cắt chuỗi (**truncate**), bạn có thể làm như sau:

Tạo file **truncate.pipe.ts**

typescript

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'truncate',
  standalone: true
})
export class TruncatePipe implements PipeTransform {
  transform(value: string, limit: number = 10): string {
    return value.length > limit ? value.substring(0, limit) + '...' : value;
  }
}
```

Sử dụng trong Standalone Component

typescript

```
import { Component } from '@angular/core';
import { TruncatePipe } from './truncate.pipe';

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [TruncatePipe],
  template: `<p>{{ 'This is a long text' | truncate:8 }}</p>`
})
export class AppComponent {}
```

Kết quả:

This is...

Tóm tắt

- ✓ **Custom Pipes trong Angular Standalone Component** cần có `standalone: true`.
 - ✓ Không cần khai báo Pipe trong `NgModule`, chỉ cần import trực tiếp vào **component**.
 - ✓ Có thể truyền tham số vào Pipe để xử lý dữ liệu linh hoạt hơn.
-

Bài tập thực hành

- 1 Tạo một Pipe `reverse` để đảo ngược chuỗi (`hello` → `olleh`).
- 2 Tạo một Pipe `maskEmail` để ẩn bớt email (`test@example.com` → `t***@example.com`).
- 3 Tạo một Pipe nhận tham số để chuyển đổi số giây thành định dạng `mm:ss` (`125` → `02:05`).

Bạn thử làm bài tập rồi gửi lại mình kiểm tra nhé! 🚀

ChatGPT can make mistakes. Check important info.