


nqt - Component Communication trong Angular

 chatgpt.com/c/67d828c3-38bc-8001-87c5-127b448779d6

ChatGPT

Bài 3: Component Communication trong Angular

Trong Angular, **giao tiếp giữa các component** là một phần quan trọng khi xây dựng ứng dụng. Khi ứng dụng có nhiều component, bạn cần một cách để truyền dữ liệu giữa chúng.

1. Các cách giao tiếp giữa component

Dưới đây là các phương pháp phổ biến để giao tiếp giữa các component trong Angular:

Phương pháp	Mô tả	Trường hợp sử dụng
@Input()	Truyền dữ liệu từ component cha → component con	Khi component cha muốn cung cấp dữ liệu cho component con
@Output() và EventEmitter	Truyền dữ liệu từ component con → component cha	Khi component con cần gửi sự kiện hoặc dữ liệu lên component cha
ViewChild và ViewChildren	Cho phép component cha truy cập trực tiếp vào component con	Khi component cha cần gọi phương thức hoặc truy cập thuộc tính của component con

Phương pháp	Mô tả	Trường hợp sử dụng
Service và Subject/BehaviorSubject	Dùng một service để chia sẻ dữ liệu giữa các component không có quan hệ cha-con	Khi cần giao tiếp giữa các component ở nhiều cấp khác nhau

2. Truyền dữ liệu từ Cha → Con bằng **@Input()**

Khi component cha muốn truyền dữ liệu xuống component con, ta dùng **@Input()**.

Ví dụ:

2.1. Tạo component con **child.component.ts**

typescript

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

@Component({
  selector: 'app-child',
  template: `<p>Received message: {{ message }}</p>`,
})
export class ChildComponent {
  @Input() message: string = ''; // Nhận dữ liệu từ component cha
}
```

2.2. Tạo component cha **parent.component.ts**

typescript

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

@Component({
  selector: 'app-parent',
  template: `<app-child [message]="parentMessage"></app-child>`,
})
export class ParentComponent {
  parentMessage = 'Hello from Parent!';
}
```

✓ Giải thích:

- Component cha **ParentComponent** truyền dữ liệu xuống component con **ChildComponent** thông qua **@Input()**.
- **[message]="parentMessage"** trong template cha truyền giá trị của **parentMessage** vào biến **message** trong component con.

3. Truyền dữ liệu từ Con → Cha bằng **@Output()**

Khi component con muốn gửi dữ liệu lên component cha, ta sử dụng **@Output()** kết hợp với **EventEmitter**.

Ví dụ:

3.1. Tạo component con **child.component.ts**

typescript

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

@Component({
  selector: 'app-child',
  template: `
    <button (click)="sendMessage()">Send Message to Parent</button>
  `,
})
export class ChildComponent {
  @Output() messageEvent = new EventEmitter<string>(); // Khai báo sự kiện
  để truyền dữ liệu

  sendMessage() {
    this.messageEvent.emit('Hello from Child!'); // Phát sự kiện gửi dữ liệu
    lên cha
  }
}
```

3.2. Tạo component cha **parent.component.ts**

typescript

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

@Component({
  selector: 'app-parent',
  template: `
    <app-child (messageEvent)="receiveMessage($event)"></app-child>
    <p>Message from Child: {{ childMessage }}</p>
  `,
})
export class ParentComponent {
  childMessage = '';

  receiveMessage(message: string) {
    this.childMessage = message; // Nhận dữ liệu từ component con
  }
}
```

✓ Giải thích:

- Component con **ChildComponent** phát sự kiện **messageEvent** với dữ liệu **"Hello from Child!"** khi nút được nhấn.
- Component cha **ParentComponent** lắng nghe sự kiện (**messageEvent**) và gọi **receiveMessage(\$event)**, cập nhật giá trị **childMessage**.

4. Truy cập Component Con từ Cha bằng **ViewChild**

@ViewChild() cho phép component cha truy cập vào component con để gọi phương thức hoặc thay đổi dữ liệu.

Ví dụ:

4.1. Tạo component con **child.component.ts**

typescript

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

@Component({
  selector: 'app-child',
  template: `<p>Message: {{ message }}</p>`,
})
export class ChildComponent {
  message = 'Hello from Child!';

  changeMessage(newMessage: string) {
    this.message = newMessage; // Thay đổi nội dung message
  }
}
```

4.2. Tạo component cha **parent.component.ts**

typescript

```
import { Component, ViewChild } from '@angular/core';
import { ChildComponent } from './child.component';

@Component({
  selector: 'app-parent',
  template: `
    <app-child></app-child>
    <button (click)="changeChildMessage()">Change Child Message</button>
  `,
})
export class ParentComponent {
  @ViewChild(ChildComponent) childComponent!: ChildComponent;

  changeChildMessage() {
    this.childComponent.changeMessage('New message from Parent!'); // Gọi
    phương thức của component con
  }
}
```

✓ Giải thích:

- **@ViewChild(ChildComponent) childComponent;** giúp cha truy cập vào component con.
- **changeChildMessage()** gọi phương thức **changeMessage()** của component con để cập nhật nội dung.

5. Giao tiếp giữa các component không liên quan bằng Service

Khi hai component không có quan hệ cha-con, ta dùng một **Service** kết hợp với **Subject** hoặc **BehaviorSubject**.

Ví dụ:

5.1. Tạo service **message.service.ts**

typescript

```
import { Injectable } from '@angular/core';
import { BehaviorSubject } from 'rxjs';

@Injectable({
  providedIn: 'root',
})
export class MessageService {
  private messageSource = new BehaviorSubject<string>('Default Message');
  currentMessage = this.messageSource.asObservable();

  changeMessage(message: string) {
    this.messageSource.next(message); // Phát dữ liệu mới
  }
}
```

5.2. Sử dụng service trong Component A (**component-a.ts**)

typescript

```
import { Component } from '@angular/core';
import { MessageService } from '../message.service';

@Component({
  selector: 'app-component-a',
  template: `<button (click)="sendMessage()">Send Message</button>`,
})
export class ComponentA {
  constructor(private messageService: MessageService) {}

  sendMessage() {
    this.messageService.changeMessage('Hello from Component A!');
  }
}
```

5.3. Sử dụng service trong Component B (**component-b.ts**)

typescript

```
import { Component } from '@angular/core';
import { MessageService } from './message.service';

@Component({
  selector: 'app-component-b',
  template: `<p>Message: {{ message }}</p>`,
})
export class ComponentB {
  message: string = '';

  constructor(private messageService: MessageService) {
    this.messageService.currentMessage.subscribe(msg => {
      this.message = msg;
    });
  }
}
```

✓ Giải thích:

- **MessageService** chứa một **BehaviorSubject** để lưu trữ và cập nhật dữ liệu.
- Component A thay đổi message bằng **changeMessage()**.
- Component B nhận dữ liệu mới bằng **subscribe()**.

💡 Tổng kết:

- **@Input()**: Truyền dữ liệu từ Cha → Con.
- **@Output() + EventEmitter**: Truyền dữ liệu từ Con → Cha.
- **@ViewChild()**: Cha truy cập trực tiếp vào con.
- **Service + Subject**: Dùng để chia sẻ dữ liệu giữa các component không liên quan.

Bạn có muốn thực hành bằng cách tự viết một ví dụ không? 🚀