



Tháng 12 khai giảng
Lập Trình **RUBY**

Full time: **540 Giờ**

Part time: **240 Giờ**

Create your future with code



Uy Tran @uytran

Follow

★ 342 👤 6 📝 12

Published Jan 31st, 10:07 PM - 2 min read

TABLE OF CONTENTS

👁 2.1K 💬 0 🔗 0

View child thông qua @Input

EventEmitter

Data service BehaviourSubject

SUGGESTED ORGANIZATIONS



Sun* Blockchain Team

📝 59 👤 8 👥 25



Avengers Group

📝 33 👤 31 👥 51



Sun* Cyber Security Team

📝 26 👤 4 👥 15



+1



3 cách để giao tiếp giữa các component với nhau trong Angular 2+

share data

angular

viewchild

Behaviour Subject

EventEmitter

Trong một ứng dụng Angular, thành phần được hướng theo component.

Có nhiều cách để chia sẻ dữ liệu, giao tiếp giữa các component trong ứng dụng, bài viết

1. View Child thông qua @ViewChild

2. Event Emitter

3. Data service BehaviourSubject

Bắt đầu thôi!

View child thông qua @Input

Trong ba cách thì sử dụng @Input là cách đơn giản nhất về cách khai báo cũng như sử dụng.

Để chia sẻ biến hoặc giá trị nào đó từ component cha với component con bên trong, chúng ta có thể sử dụng @Input:

Ở children component: Khởi tạo:

```
// children.component.ts
import { Component, OnInit, Input } from '@angular/core';

@Component({
  selector: 'app-children',
  template: `
    <p>
```

```
<span>Message from parent component: {{childMessage}}</span>
',
  styleUrls: ['./children.component.css']
})
export class ChildrenComponent implements OnInit {
  @Input() childMessage: string;

  constructor() { }

  ngOnInit() {
  }
}
```

Thông qua biến `@Input childMessage`, parent component có thể truyền attribute vào children component.

Ở parent component chỉ việc khai báo:

```
// parent.component.ts
parentMessage: string = "Message from parent";
```

Ō template.html:

Vậy là `ParentComponent` đã có thể truyền attribute tới `ChildrenComponent` thông qua `@Input`.

EventEmitter

Cách thứ 2 là thông qua biến @Output sử dụng EventEmitter:

Event emitter được thiết kế để báo cho component cha khi component con có sự thay đổi

Thông qua biến `@Output`, EventEmitter sẽ bắn một value nào đó ra ngoài, và component cha sẽ bắt được value này.

- Ở component con:

```
<button class="btn btn-primary" (click)="voted()">Click to vote</button>
```

```
// ...
@Output() voteSize = new EventEmitter();
counter: number = 0;

// ...
```

```
this.voteSize.emit(this.counter);
// Hàm vote sẽ tăng counter lên 1, đồng thời thông qua EventEmitter bắn value count
}
// ...
```

- Ở component cha:

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

@Component({
  selector: 'app-parent',
  template: `
    <app-children (voteSize)="voteCount($event)"></app-children>
    <h1>Total vote from children: {{vote}}</h1>
  `,
  styleUrls: ['./parent.component.css']
})

export class ParentComponent implements OnInit {

  vote: number = 0;

  constructor() { }

  ngOnInit() {
  }
}
```

+1



Data service BehaviourSubject

Data service sử dụng BehaviourSubject (rxjs).



Rxjs hỗ trợ observes (consuming interface) và observables (push interface) (có bài viết đọc khá dễ hiểu, các bạn có thể tham khảo thêm [ở đây](#)).

Ở ví dụ sau mình sử dụng cả 2 interface trên của BehaviourSubject, thông qua hàm `asObservable()`

```
ng generate service services/data
```

File data.service.ts:

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

import { BehaviorSubject } from 'rxjs';

@Injectable({
  providedIn: 'root'
```

VIBLO

Posts

Questions

Discussions

Search Viblo



→ Sign In/Sign up



+1



```
currentMessage = this.messageSource.asObservable();
// có thể subscribe theo dõi thay đổi value của biến này thay cho messageSource

constructor() { }

// method này để change source message
changeMessage(message) {
  this.messageSource.next(message);
}
}
```

Bây giờ để chia sẻ data giữa các component, chúng ta cần import DataService:

```
import { DataService } from '../services/data.service';
```

Gọi hàm changeMessage để push data:

```
createMessage(message) {
  this.data.changeMessage(message);
}
```

Và subscribe vào currentMessage để get data:

VIBLO

Posts

Questions

Discussions

Search Viblo



Sign In/Sign up



+1



Các bạn có thể clone source code demo các example này tại đây: <https://github.com/at-uytran/share-data-example>

Như vậy mình vừa trình bày về 3 cách để giao tiếp giữa các component với nhau trong angular.



Hy vọng bài viết có thể giúp ích cho những ai còn thắc mắc hay đang tìm hiểu về cách giao tiếp giữa các component trong angular.

Cảm ơn vì đã ghé đọc bài viết, nếu các bạn có thắc mắc hay góp ý gì hãy để lại comment nhé.



Related

[VueJS] Giao tiếp giữa các component

Le Xuan Duy

8 min read

👍 12505 👎 24 🔄 15 📈 41

Tương tác giữa các component trong angular...

Phan Tân

2 min read

👍 2769 👎 4 🔄 1 📈 0

Angular 2 căn bản - Phần 3: Component thứ hai và data...

Vu Thanh Tung

4 min read

👍 1056 👎 1 🔄 0 📈 0

[Angular] Những kiến thức cơ bản để tạo nên một web app

Le Van Chien

7 min read

👍 1226 👎 1 🔄 0 📈 1

VIBLO

Posts

Questions

Discussions

Search Viblo



Sign In/Sign up

More from Uy Tran

Scaling & load balancing ứng dụng của bạn với Docker...

Virtual DOM and DOM - So sánh cơ chế thao tác DOM...

Deploy Angular và Rails api lên AWS EC2 sử dụng Dock...

Dockerize rails app using Docker and Docker compose

VIBLO

Posts

Questions

Discussions

Search Viblo



Sign In/Sign up

Comments



Login to comment

RESOURCES

Posts
Questions
Videos
Discussions
Tools
System Status

Organizations
Tags
Authors
Recommend System
Machine Learning

SERVICES

 Viblo Code
 Viblo CV

MOBILE APP



LINKS

