Angular Component Communication & Sharing Data

Leave a Comment / 4 minutes of reading / May 16, 2021

← Using Pipes in Components/Services

Passing Data from Parent to Child Component →

In this article, we will show you a few ways in which Angular Components can communicate or interact with each other. The Component is the main building block of an Angular App. A typical Angular application consists of a lot of components. Each component handles a small part of the UI. These components must interact or communicate together to produce the complete user interface of the application

Table of Contents

Component Communication

There are few ways in which components can communicate or share data between them. And methods depend on whether the components have a Parent-child relationship between them are not.

Here are the three Possible scenarios

1. Parent to Child Communication
2. Child to Parent Communication
3. Interaction when there is no parent-child relation

Parent to Child Communication

If the Components have a parent-child relationship then, then the parent component can pass the data to the child using the @input Property.

Using @Input Decorator to Pass Data

Create a property (someProperty) in the Child Component and decorate it with @Input() . This will mark the property as input property

```
export class ChildComponent {
   @Input() someProperty: number;
}
```

And in the Parent Component Instantiate the Child Component. Pass the value to the someProperty using the Property Bind Syntax

```
<child-component [someProperty]=value></child-component>`
```

```
3
```

In this way, Child Component will receive the data from the parent

You can refer to the tutorial pass data from parent to child in Angular.

Listen for Input Changes

The Child Component can get the values from the `someProperty`. But it also important for the child component to get notification when the values changes.

There are two ways in which we can achieve that.

1. Using `OnChanges` life Cycle hook or
2. Using a Property Setter on Input Property

Refer to the following tutorial

1. Pass data from parent to child in Angular.
2. OnChanges Life Cycle hook
3. Angular @input, @output & EventEmitter

Child to Parent Communication

The Child to Parent communication can happen in three ways.

1. Listens to Events from Child
2. Uses Local Variable to access the child in the Template
3. Uses a @ViewChild to get a reference to the child component

Listens to Child Event

This is done by the child component by exposing an EventEmitter Property. We also decorate this Property with @Output decorator. When Child Component needs to communicate with the parent it raises the emit event of the EventEmitter Property. The Parent Component listens to that event and reacts to it.

For Example, refer to the tutorial Parent Listens to Child Event

Uses Local Variable to access the child

Using Local Variable is to refer to the child component is another technique.

For Example, Create a reference variable `#child` to the Child Component.

```
<child-component #child></child-component>
```

You can use the `child` (note without #) to access a property of the Child Component. The Code below displays `count` of the Child Component and displays it on screen

```
<p> current count is {{child.count}} </p>
```

For Example, refer to the tutorial local variable to access the Child in Template

Uses a @ViewChild to get the reference to the child component

```
<child-component></child-component>
```

Another way to get the reference of the child component is using the ViewChild query in the component class

```
@ViewChild(ChildComponent) child: ChildComponent;
```

You can call any method in the Child component.

```
increment() {
  this.child.increment();
}
```

For Complete explanation please refer to Pass data from child to Parent Component & ViewChild, ViewChildren & QueryList

Communication when there is no relation

If the Components do not share the Parent-child relationship, then the only way they can share data is by using the services and observable.

The advantageous of using service is that

1. You can share data between multiple components.
2. Using observable, you can notify each component, when the data changes.

Create a Service and create an Angular Observable in that service using either BehaviorSubject or Subject.

```
```

```
export class TodoService {

  private _todo = new BehaviorSubject<Todo[]>([]);
  readonly todos$ = this._todo.asObservable();
  ...
}

```

The `_todo` observable will emit data, whenever it is available or changes using the `next` method of the Subject.

```

this._todo.next(Object.assign([], this.todos));

```

In the component class, you can listen to the changes just by subscribing to the observable

```

  this.todoService.todos$.subscribe(val=> {
    this.data=val;
    //do whatever you want to with it.
  })

```

For the complete explanation of the code, you can refer to Angular Subject Example for sharing Data Between Components

Reference

1. https://angular.io/guide/component-interaction
2. Examples

Read More