**Angular-T09-HOL001\_TestCase [Test @Input and @Output Angular)**

In this hands-on, we will test @Input and @Output decorator through which we can pass data among parent-child components. Also you will know how to use the spyOn function to effectively “spy” on the emit method of EventEmitter object.

**Tasks:**

1. Under AppComponent class, create a property called message with the value “This is a message from parent component”
2. Create a component called Message.
3. Pass the message info from AppComponent to MessageComponent using @Input() decorator. Display the message of AppComponent both under app.component.html (in an h1 tag) and message.component.html (in a h2 tag).
4. Under message.component.html take an input field and a button.
5. Use @Output() decorator and create a EventEmitter object. On click of the button pass the info of textbox to AppComponent.
6. Create a method under AppComponent called childMessage(newMessage:string) to receive the passed data from MessageComponent.
7. Display the MessageComponent data under app.component.html under an h3 tag.

**Sample app.component.ts code:**

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

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  message = 'This is a message from parent component';

  childMNessage:string;

  childMessage(newmessage) {

    this.childMNessage = newmessage;

  }

}

**Sample app.component.html code:**

<h1>

  Parent Compoent Message: {{message}}

</h1>

<br/>

<h3>Child Component Called Under Prent: {{childMNessage}}</h3>

<br/><br/>

<app-message [message]="message" (messageEmitter)="childMessage($event)"></app-message>

**Sample message.component.ts code:**

import { Output } from '@angular/core';

import { ViewChild } from '@angular/core';

import { ElementRef } from '@angular/core';

import { EventEmitter } from '@angular/core';

import { Input } from '@angular/core';

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

@Component({

  selector: 'app-message',

  templateUrl: './message.component.html',

  styleUrls: ['./message.component.css']

})

export class MessageComponent {

  @Input() message: string;

  @Output() messageEmitter=new EventEmitter<string>();

  handleButtonClick(newMessage:string) {

    if(newMessage) {

      this.messageEmitter.emit(newMessage);

    }

  }

}

**Sample message.component.html code:**

<br/>

<h2>Parent Component Message Call: {{ message }}</h2>

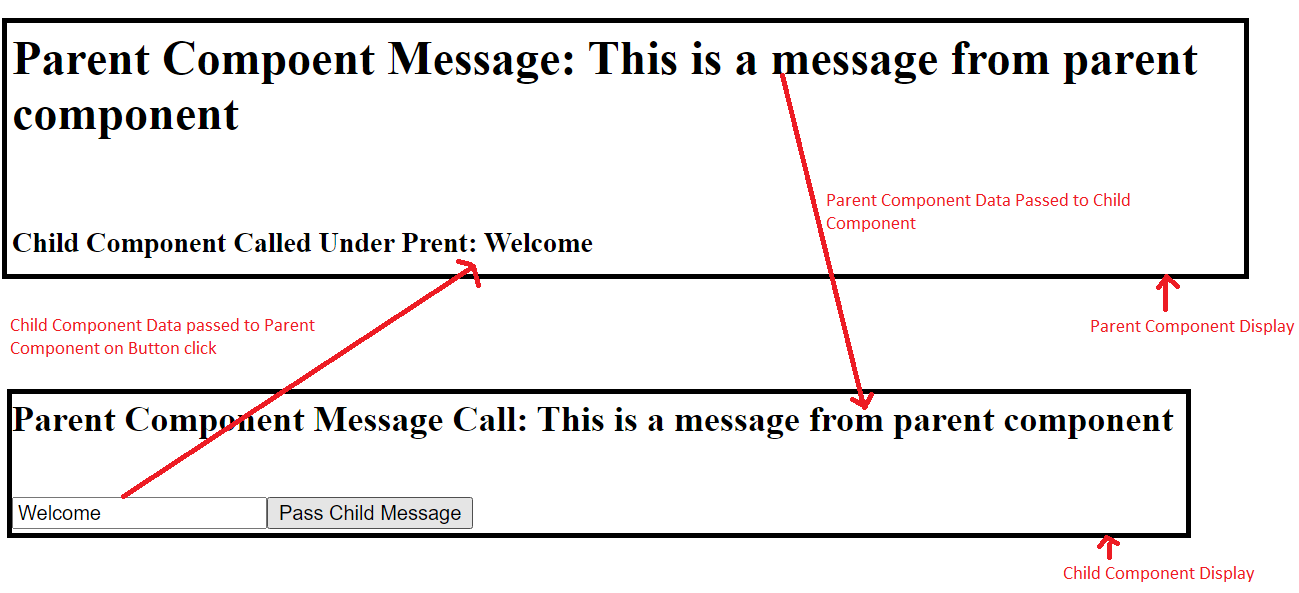
<br/>

<input #messageField type="text" />

<button (click)="handleButtonClick(messageField.value)">Pass Child Message</button>

***Note: The sample codes are given just for your reference. You can use your own variable names or method names or property values or your own codes for passing data between parent and child components using @Input and @Output decorator.***

**Sample Output:**

****

1. Check if the describe () block for AppComponent is present under app.component.spec.ts.
2. Check whether a beforeEach() block that instantiates a new instance of AppComponent for each spec is present. Also under beforeEach() block add MessageComponent.
3. Check if there is a test case to check whether the AppComponent is created is present.
4. Create a test case to check if the parent message with value “This is a message from parent component” is present.
5. Create a test case to check if the parent message is displayed in an h1 tag.
6. Check if the describe () block for MessageComponent is present under message.component.spec.ts.
7. Check whether a beforeEach() block that instantiates a new instance of MessageComponent for each spec is present.
8. Check if there is a test case to check whether the MessageComponent is created is present.
9. Create a test case to check if the MessageComponent correctly renders the passed @Input value.
10. Create a test case to check if the MessageComponent is correctly passing the value to AppComponent using @Output decorator.

***Sample code is given for your reference. Please feel free to modify the test cases according to your skills.***

**App.comonent.spec.ts code:**

import { TestBed, async } from '@angular/core/testing';

import { AppComponent } from './app.component';

import { MessageComponent } from './../app/message/message.component';

describe('AppComponent', () => {

  beforeEach(async(() => {

    TestBed.configureTestingModule({

      declarations: [

        AppComponent,

        MessageComponent

      ],

    }).compileComponents();

  }));

  it('should create the AppComponent', () => {

    const fixture = TestBed.createComponent(AppComponent);

    const app = fixture.debugElement.componentInstance;

    expect(app).toBeTruthy();

  });

  it(`should have a parent own message 'This is a message from parent component'`, () => {

    const fixture = TestBed.createComponent(AppComponent);

    const app = fixture.debugElement.componentInstance;

    expect(app.message).toEqual('This is a message from parent component');

  });

  it('should render parent own message in a h1 tag', () => {

    const fixture = TestBed.createComponent(AppComponent);

    fixture.detectChanges();

    const compiled = fixture.debugElement.nativeElement;

    expect(compiled.querySelector('h1').textContent).toContain('Parent Compoent Message: This is a message from parent component');

  });

});

**Message.component.spec.ts code:**import { async, ComponentFixture, TestBed } from '@angular/core/testing';

import { MessageComponent } from './message.component';

describe('MessageComponent', () => {

  let component: MessageComponent;

  let fixture: ComponentFixture<MessageComponent>;

  beforeEach(async(() => {

    TestBed.configureTestingModule({

      declarations: [ MessageComponent ]

    })

    .compileComponents();

  }));

  beforeEach(() => {

    fixture = TestBed.createComponent(MessageComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create MessageComponent', () => {

    expect(component).toBeTruthy();

  });

  it('should correctly render the passed @Input value', () => {

    component.message = 'Calling Parent Message in Child Message Component';

    fixture.detectChanges();

    const compiled = fixture.debugElement.nativeElement;

    expect(compiled.querySelector('h2').textContent).toBe('Parent Component Message Call: Calling Parent Message in Child Message Component');

  });

  it('should correctly @Output value of text input in component', () => {

    spyOn(component.messageEmitter, 'emit');

    const button = fixture.nativeElement.querySelector('button');

    fixture.nativeElement.querySelector('input').value = 'Welcome User';

    const inputText = fixture.nativeElement.querySelector('input').value;

    button.click();

    fixture.detectChanges();

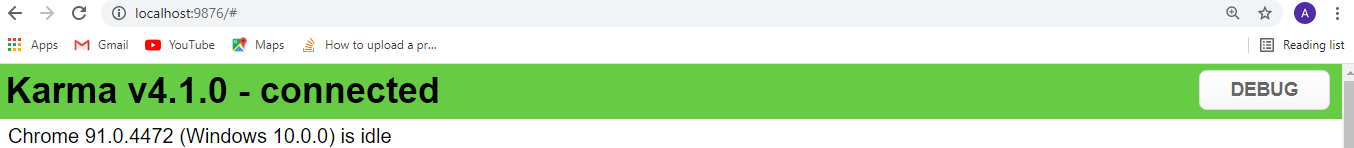
    expect(component.messageEmitter.emit).toHaveBeenCalledWith(inputText);

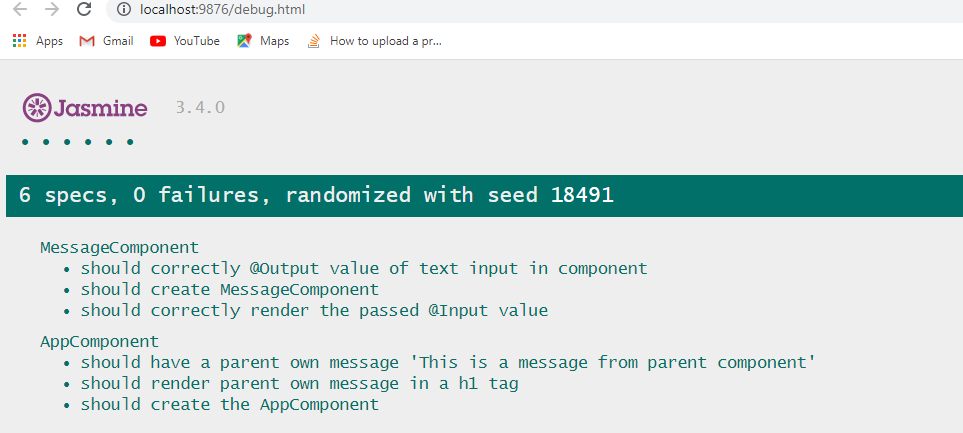
  });

});

1. Run the command ng test in cli and click the debug button of Karma window in browser to see the jasmine test results.

**Sample output is given for your reference:**

****

****