Simple Angular testing Demo

Step 1 : create a project

App.component.ts

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

@Component({

  selector: 'app-root',

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

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  value = 0;

  message!: string;

  increment() {

    if (this.value < 15) {

      this.value += 1;

      this.message = '';

    }

     else {

      this.message = 'Maximum reached!';

    }

  }

  decrement() {

    if (this.value > 0) {

      this.value -= 1;

      this.message = '';

    }

    else {

      this.message = 'Minimum reached!';

    }

  }

  }

App.component.html

<h1>{{ value }}</h1>

<hr>

<button (click)="increment()" class="increment">Increment</button>

<button (click)="decrement()" class="decrement">Decrement</button>

<p class="message">  {{ message }} </p>

App.component.spec.ts

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

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

import { By } from '@angular/platform-browser';

import { RouterTestingModule } from '@angular/router/testing';

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

describe('AppComponent', () => {

  let fixture: ComponentFixture<AppComponent>;

  let debugElement: DebugElement;

  beforeEach(waitForAsync(() => {

    TestBed.configureTestingModule({

      declarations: [

        AppComponent

      ],

    }).compileComponents();

    fixture = TestBed.createComponent(AppComponent);

    debugElement = fixture.debugElement;

  }));

  it('should increment and decrement value', () => {

    fixture.componentInstance.increment();

    expect(fixture.componentInstance.value).toEqual(1);

    fixture.componentInstance.decrement();

    expect(fixture.componentInstance.value).toEqual(0);

  });

  it('should increment value in template', () => {

    debugElement

      .query(By.css('button.increment'))

      .triggerEventHandler('click', null);

    fixture.detectChanges();

    const value = debugElement.query(By.css('h1')).nativeElement.innerText;

    expect(value).toEqual('1');

  });

  it('should stop at 0 and show minimum message', () => {

    debugElement

      .query(By.css('button.decrement'))

      .triggerEventHandler('click', null);

    fixture.detectChanges();

    const message = debugElement.query(By.css('p.message')).nativeElement.innerText;

    expect(fixture.componentInstance.value).toEqual(0);

    expect(message).toContain('Minimum');

  });

  it('should stop at 15 and show maximum message', () => {

    fixture.componentInstance.value = 15;

    debugElement

      .query(By.css('button.increment'))

      .triggerEventHandler('click', null);

    fixture.detectChanges();

    const message = debugElement.query(By.css('p.message')).nativeElement.innerText;

    expect(fixture.componentInstance.value).toEqual(15);

    expect(message).toContain('Maximum');

  });

});

Command to run the test

Ng test 🡺 karma test runner will run by default and create the browser instance, and run the tests to provide the expected results.

Steps to Test a service implemented in Angular

Ng g class quoteModel

Quote-Model.ts

export class QuoteModel {

    constructor(public text: String, public timeCreated: String) {}

}

Quote-model.spec.ts

import { QuoteModel } from './quote-model';

describe('QuoteModel', () => {

  it('should create an instance', () => {

    expect(new QuoteModel('text','time')).toBeTruthy();

  });

});

Add service:

Ng g s quote

quoteService.ts

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

import { QuoteModel } from './quotes/models/quote-model';

@Injectable({

  providedIn: 'root'

})

export class QuoteService {

  public quoteList: QuoteModel[] = [];

  constructor() {

  }

addNewQuote(quote: String) {

    console.log("in service quoteText is:"+quote);

    const date = new Date();

     const day = date.getDay();

     const month=date.getMonth();

     const year = date.getFullYear();

    this.quoteList.push(

      new QuoteModel(quote, `${day}-${month}-${year}`)

    );

  }

  getQuote() {

    return this.quoteList;

  }

  removeQuote(index: number) {

    this.quoteList.splice(index, 1);

  }

}

Quote-service.spec.ts

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

import { QuoteService } from './quote.service';

describe("QuoteService", () => {

  let service: QuoteService;

  beforeEach(() => {

    service = new QuoteService();

  });

  it("should create a post in an array", () => {

    const qouteText = "This is my first post";

    service.addNewQuote(qouteText);

    expect(service.quoteList.length).toBeGreaterThanOrEqual(1);

  });

  it("should remove a created post from the array of posts", () => {

    service.addNewQuote("This is my first post");

    service.removeQuote(0);

    expect(service.quoteList.length).toBeLessThan(1);

  });

});

Ng g c quote

Quote.component.ts

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

import { QuoteService } from '../quote.service';

import { QuoteModel } from '../quotes/models/quote-model';

@Component({

  selector: 'app-quote',

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

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

})

export class QuoteComponent implements OnInit {

  public quoteList: QuoteModel[]=[];

  public quoteText:any = null;

  constructor(private service: QuoteService) { }

  ngOnInit() {

    this.quoteList = this.service.getQuote();

  }

  createNewQuote() {

    this.service.addNewQuote(this.quoteText);

    this.quoteText =null;

  }

  removeQuote(index: number) {

    this.service.removeQuote(index);

  }

}

Quote.component.spec.ts

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

import { By } from "@angular/platform-browser";

import { QuoteComponent } from "./quote.component";

import { QuoteService} from '../quote.service';

import { QuoteModel } from "../quotes/models/quote-model";

import { FormsModule } from "@angular/forms";

describe("QuotesComponent", () => {

  let component: QuoteComponent;

  let fixture: ComponentFixture<QuoteComponent>;

  beforeEach(() => {

    TestBed.configureTestingModule({

      imports: [FormsModule],

      declarations: [QuoteComponent]

    });

  });

  beforeEach(() => {

    fixture = TestBed.createComponent(QuoteComponent);

    component = fixture.debugElement.componentInstance;

  });

  it("should create Quote component", () => {

    expect(component).toBeTruthy();

  });

  it("should use the quoteList from the service", () => {

    const quoteService = fixture.debugElement.injector.get(QuoteService);

    fixture.detectChanges();

    expect(quoteService.getQuote()).toEqual(component.quoteList);

  });

  it("should create a new post", () => {

    component.quoteText = "I love this test";

    fixture.detectChanges();

    const compiled = fixture.debugElement.nativeElement;

    expect(compiled.innerHTML).toContain("I love this test");

  });

  it("should disable the button when textArea is empty", () => {

    fixture.detectChanges();

    const button = fixture.debugElement.query(By.css("button"));

    expect(button.nativeElement.disabled).toBeTruthy();

  });

  it("should enable button when textArea is not empty", () => {

    component.quoteText = "I love this test";

    fixture.detectChanges();

    const button = fixture.debugElement.query(By.css("button"));

    expect(button.nativeElement.disabled).toBeFalsy();

  });

  it("should remove post upon card click", () => {

    component.quoteText = "This is a fresh post";

    fixture.detectChanges();

    fixture.debugElement

      .query(By.css(".row"))

      .query(By.css(".card"))

      .triggerEventHandler("click", null);

    const compiled = fixture.debugElement.nativeElement;

    expect(compiled.innerHTML).toContain("This is a fresh post");

  });

  // it("should fetch data asynchronously", async () => {

  //   const fakedFetchedList = [

  //     new QuoteModel("I love unit testing", "Mon 4, 2018")

  //   ];

  //   const quoteService = fixture.debugElement.injector.get(QuoteService);

  //   let spy = spyOn(quoteService, "fetchQuotesFromServer").and.returnValue(

  //     Promise.resolve(fakedFetchedList)

  //   );

  //   fixture.detectChanges();

  //   fixture.whenStable().then(() => {

  //     expect(component.fetchedList).toBe(fakedFetchedList);

  //   });

  // });

});

Quote.component.html

<p>quote works!</p>

<div class="container-fluid">

    <div class="row">

      <div class="col-8 col-sm-8 mb-3 offset-2">

        <div class="card">

          <div class="card-header">

            <h5>What Quote is on your mind ?</h5>

          </div>

          <div class="card-body">

            <div role="form">

              <div class="form-group col-8 offset-2">

                <textarea #quote class="form-control" rows="3" cols="8" [(ngModel)]="quoteText"

                 name="quoteText"></textarea>

              </div>

              <div class="form-group text-center">

                <button class="btn btn-primary" (click)="createNewQuote()"

                [disabled]="quoteText == null">Create a new

                  quote</button>

              </div>

            </div>

          </div>

        </div>

      </div>

    </div>

    <div class="row">

      <div class="card mb-3 col-5 list-card" id="quote-cards" style="max-width: 18rem;"

       \*ngFor="let quote of quoteList; let i = index"

        (click)="removeQuote(i)">

        <div class="card-body">

          <h6>{{ quote.text }}</h6>

        </div>

        <div class="card-footer text-muted">

          <small>Created on {{ quote.timeCreated }}</small>

        </div>

      </div>

    </div>

  </div>

App.component.html

<app-quote>

</app-quote>

<router-outlet></router-outlet>

Index.html

<!doctype html>

<html lang="en">

<head>

  <meta charset="utf-8">

  <title>Cgitestingdemo</title>

  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">

  <link rel="icon" type="image/x-icon" href="favicon.ico">

<!-- Latest compiled and minified CSS -->

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">

<!-- Optional theme -->

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css" integrity="sha384-rHyoN1iRsVXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRYooPp2bWYgmgJQIXwl/Sp" crossorigin="anonymous">

<!-- Latest compiled and minified JavaScript -->

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js" integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa" crossorigin="anonymous"></script>

</head>

<body>

  <app-root></app-root>

</body>

</html>

