**Set up**

First, create a folder called BDD under C:\Users\{your\_username}\sandbox\

Open a DOS command prompt window.

cd C:\Users\{your\_username}\sandbox\BDD

git clone https://github.com/sgavathe/ng-pokedex.git

cd ng-pokedex

npm install

npm install -g @angular/cli@latest

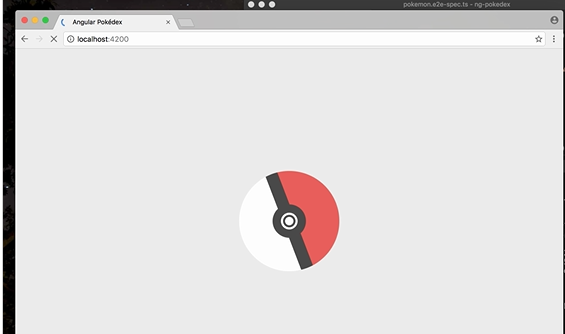
set PATH=%PATH%;%APPDATA%\npm

**Development server**

Run ng serve for a dev server. Navigate to http://localhost:4200/. The app will automatically reload if you change any of the source files.

**ng serve**

Launch Chrome. Navigate to <http://localhost:4200/>. You should see:

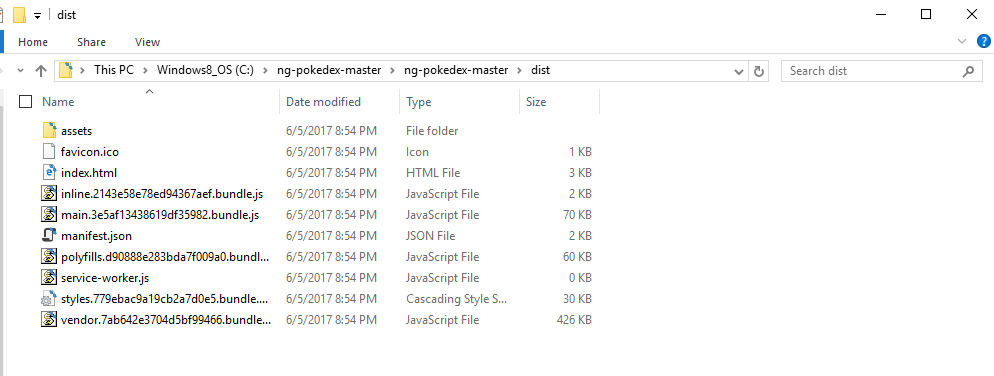


*Open Development environment complete!*

**Build**

Run ng build to build the project. The build artifacts will be stored in the **dist/** directory. Use the -prod flag for a production build.

ng build –prod



*Note: Bundled files under dist directory are the ones that you will need to deploy into production.*

***src/****stands for source, and is the raw code before minification or concatenation or some other compilation - used to read/edit the code.*

***dist/****stands for distribution, and is the minified/concatenated version - actually used on production sites.*

*Build Process complete!*

**Running unit tests**

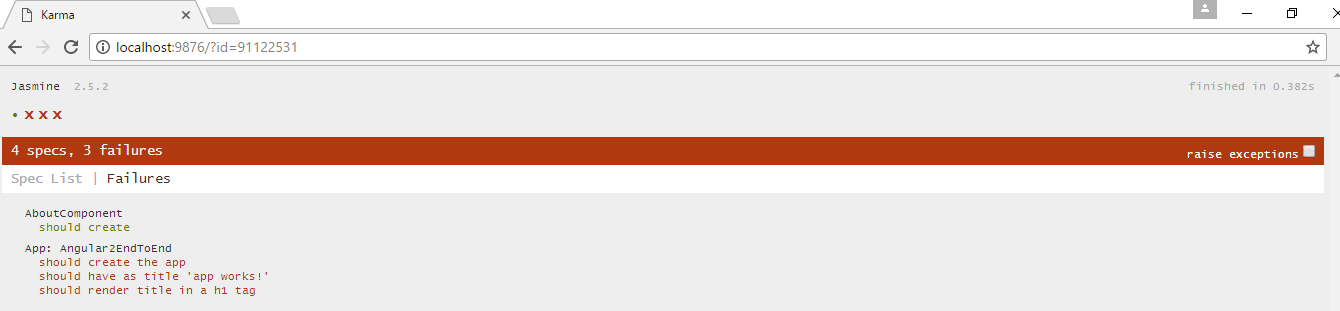
**--add step to enable html report for karma**

Run ng test to execute the unit tests via [Karma](https://karma-runner.github.io/).

ng test

You will see browser with karma debug screen running as below.

[http://localhost:9876/?id={some](http://localhost:9876/?id=%7bsome) id}

**

**You will notice that 3 out of 4 tests are failing. We will do below to fix those errors.**

Open Project using the Visual Studio Code, if not already opened.

Open file [src](https://github.com/sgavathe/ng-pokedex/tree/master/e2e)/app/app.component.spec.ts and copy below.

/\* tslint:disable:no-unused-variable \*/

/\* tslint:disable:no-unused-variable \*/

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

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

import { CUSTOM\_ELEMENTS\_SCHEMA, NO\_ERRORS\_SCHEMA, SchemaMetadata, SecurityContext} from '@angular/core';

//import { PokemonService } from './services/pokemon.service';

describe('App: Pokeman', () => {

let component: AppComponent;

beforeEach(() => {

TestBed.configureTestingModule({

declarations: [

AppComponent

],

schemas: [NO\_ERRORS\_SCHEMA]

});

});

it('should create the app', async(() => {

let fixture = TestBed.createComponent(AppComponent);

let app = fixture.debugElement.componentInstance;

expect(app).toBeTruthy();

}));

it(`should have as title 'app works!'`, async(() => {

let fixture = TestBed.createComponent(AppComponent);

let app = fixture.debugElement.componentInstance;

expect(app.title).toEqual('app works!');

}));

it('should render title in a h1 tag', async(() => {

let fixture = TestBed.createComponent(AppComponent);

fixture.detectChanges();

let compiled = fixture.debugElement.nativeElement;

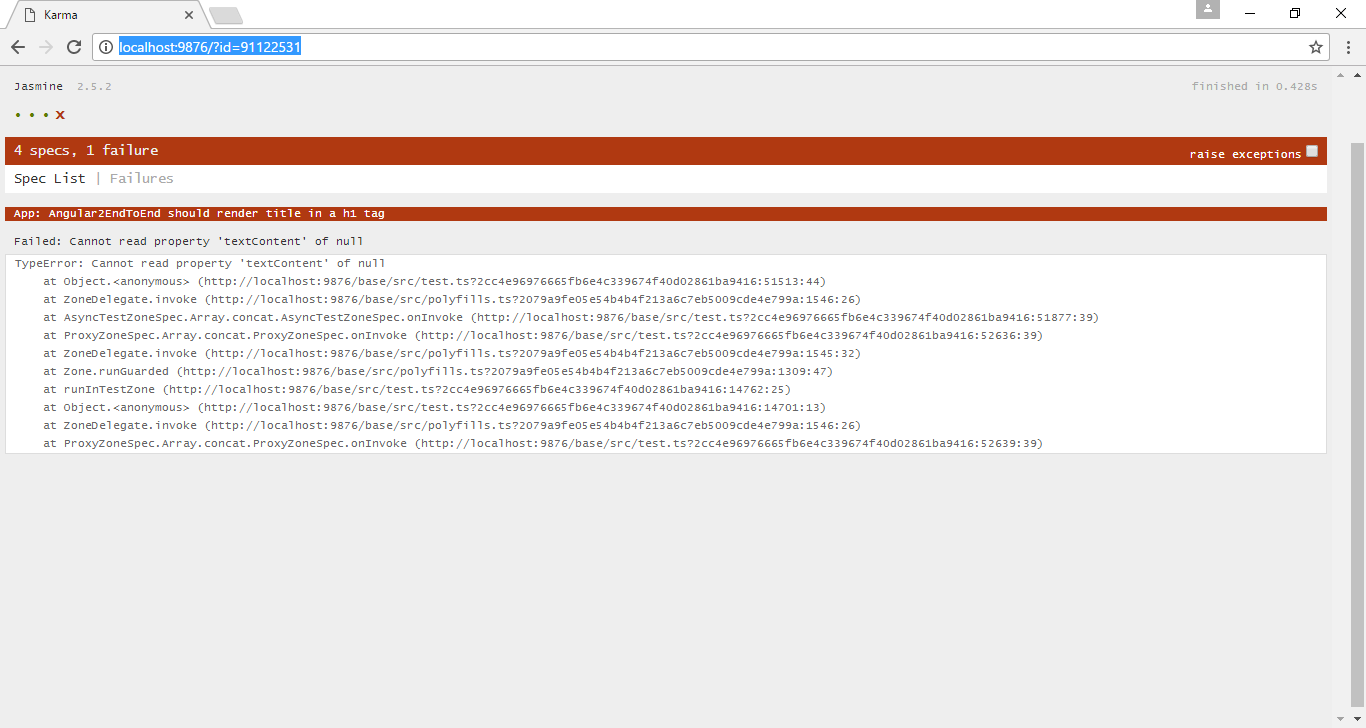
expect(compiled.querySelector('h1').textContent).toContain('app works!');

}));

});

*Go back to browser where jasmine screen is being executing,*

**You will notice that 3 out of 4 tests are passed as below.**

****

*Let’s add more code to test services. We will create new TS Spec file for our service tests.*

**cd C:\Users\{your\_username}\sandbox\BDD\ ng-pokedex \src\app\common\core\services**

**echo.> app.common.services.spec.ts**

Open file [src/app/common/core/services](https://github.com/sgavathe/ng-pokedex/tree/master/e2e)/app.common.services.spec.ts (VSC should still be open).

//This is an "isolated unit test", as it does require angular dependencies

//or interact with service

import { PokemonService } from './pokemon.service';

import { Pokemon } from './../../interfaces/pokemon';

import { Observable } from 'rxjs/Rx';

describe('LookupService', () => {

let service: PokemonService;

let pokemon: Observable<Pokemon[]>;

beforeEach(() => { service = new PokemonService});

it('service should be instantiated', () => {

expect(service).toBeTruthy();

});

it('should contain the 151 elements', () => {

service.getPokemanList().subscribe((blogs: Pokemon[]) => {

expect(blogs.length).toBe(151); ////observable array test

});

});

it('should 148th element be dragonair', () => {

service.getPokemanList().subscribe((blogs: Pokemon[]) => {

expect(blogs[147].name).toBe("dragonair"); //observable array test

});

});

});

*Add new method to test service data.*

Open file [src/app/common/core/services](https://github.com/sgavathe/ng-pokedex/tree/master/e2e)/pokemon.service.ts (VSC should still be open).

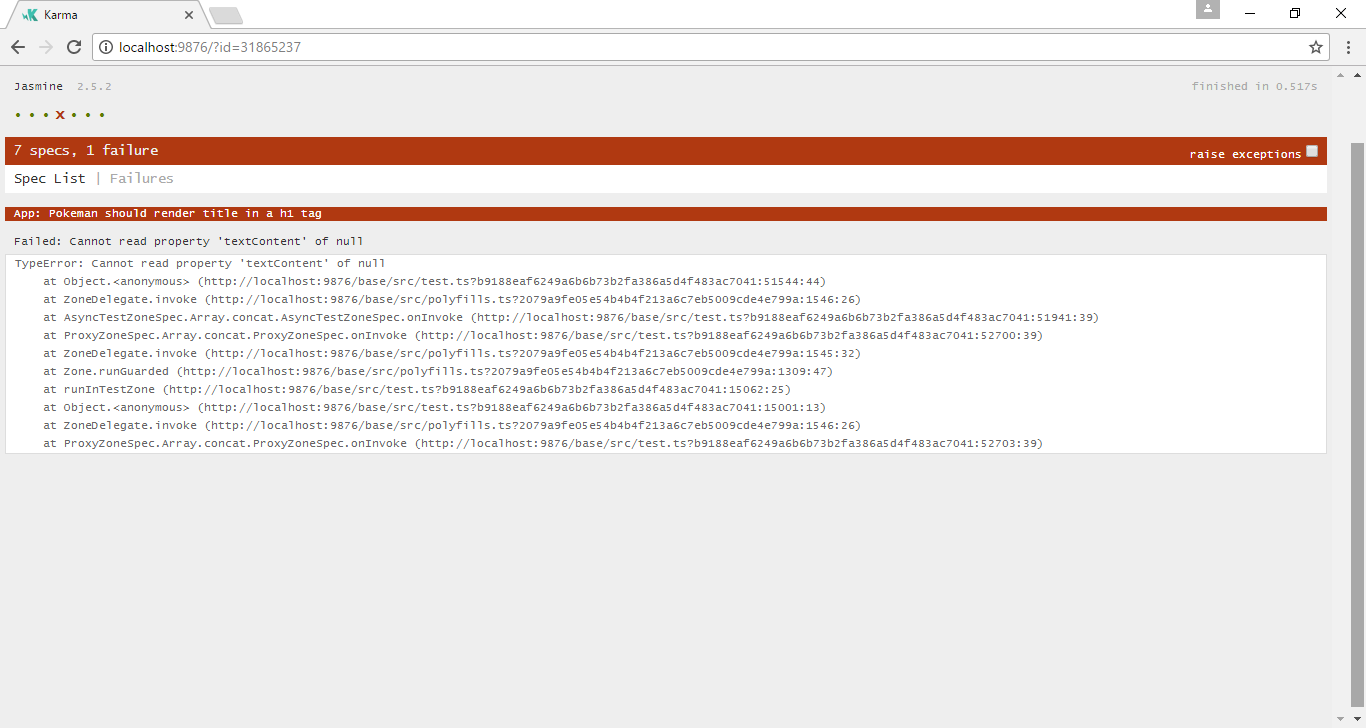
public getPokemanList() {

return this.pokemon;

}

*Go back to browser where jasmine screen is being executing,*

**You will notice that 6 out of 7 tests are passed as below.**

**

*Unit Tests complete!*

***Now it’s time for the real and fun tests. The End 2 End Test Begins Here …….***

**Angular CLI and Protractor**

Open file [e2e](https://github.com/sgavathe/ng-pokedex/tree/master/e2e)/app.po.ts (VSC should still be open).

import { browser, element, by } from 'protractor';

export class NgPokedexPage {

navigateTo() {

// Navigate to the home page of the app

return browser.get('/');

}

getHeadingText() {

// Get the home page heading element reference

return element(by.css('app-root h1')).getText();

}

}

}

Open file e2e/app.e2e-spec.ts in your workspace and copy following

// app.e2e-spec.ts

import { NgPokedexPage } from './app.po';

describe('ng-pokedex App', function() {

let page: NgPokedexPage;

beforeEach(() => {

page = new NgPokedexPage();

});

it('should display heading saying NG-Pokédex', () => {

page.navigateTo();

expect(page.getHeadingText()).toEqual('NG-Pokédex');

});

});

Open file e2e/pokemon.po.ts in your workspace and copy following

import { browser, element, by, Key } from 'protractor';

export class PokemonPage {

navigateTo() {

return browser.get('/pokemon');

}

getPokemonCardElements() {

return element.all(by.css('.card--media'));

}

getFirstPokemonCardElement() {

return element(by.css('.card--media'));

}

getOpenModalElement() {

return element(by.tagName('app-pokemon-modal'));

}

getOpenModalHeadingElement() {

return element(by.css('app-pokemon-modal h1'));

}

selectNextKey() {

browser.actions().sendKeys(Key.ARROW\_RIGHT).perform();

}

selectPrevKey() {

browser.actions().sendKeys(Key.ARROW\_LEFT).perform();

}

selectEscapeKey() {

browser.actions().sendKeys(Key.ESCAPE).perform();

}

}

**E2E Test**

In our E2E scenario we want to test the user can navigate and view Pokémon on the page.

Open file e2e/ pokemon.e2e-spec.ts in your workspace and copy following

// pokemon.e2e-spec.ts

import { PokemonPage } from './pokemon.po';

import { browser } from 'protractor';

describe('ng-pokedex pokemon view', function () {

let page: PokemonPage;

beforeEach(() => {

page = new PokemonPage();

});

it('should display a list of pokemon', () => {

page.navigateTo();

expect(page.getPokemonCardElements().count()).toBe(151);

});

it('should open and view a particular pokemon', () => {

page.navigateTo();

page.getFirstPokemonCardElement().click();

expect(page.getOpenModalElement()).toBeTruthy();

expect(page.getOpenModalHeadingElement().getText()).toBe('Bulbasaur #1');

});

it('should open and allow arrow keys to navigate between pokemon', () => {

page.navigateTo();

page.getFirstPokemonCardElement().click();

page.selectNextKey();

expect(page.getOpenModalHeadingElement().getText()).toBe('Ivysaur #2');

page.selectPrevKey();

page.selectPrevKey();

expect(page.getOpenModalHeadingElement().getText()).toBe('Mew #151');

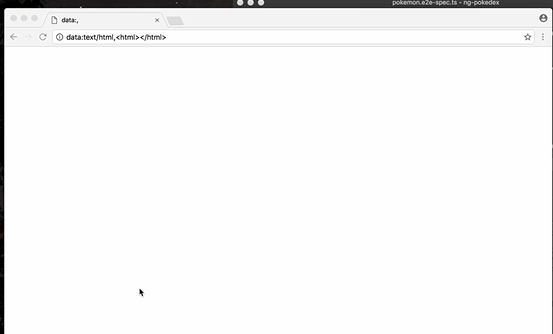
});

});

Run following command to see how end to end test runs on the browser.

**ng e2e**

It shall launch the browser automatically to execute the test cases we wrote. It will be the end users tests.



Run ng e2e to execute the end-to-end tests via [Protractor](http://www.protractortest.org/). Before running the tests make sure you are serving the app via ng serve.

*Final step and workshop complete!*