Angular 单元测试编写入门指南 – 孤立的单元测试

孤立的单元测试意味着它很孤独，但同时又是轻量级的。属于小清新之类的。这里讲的是Angular，所以默认语言是TypeScript.

- 推荐的使用范围： 可用于测试服务、带依赖的服务、管道、组件api边界测试（静态测试，输入可以是动作，输出可以是结果）。

  > 建议： 独立测试对象尽量调用简单。被测对象要有全局变量或返回值，以便进行expect断言。

- 编写方法： 编写孤立的单元测试，就是将对象看成类，通过new的方式创建类。这个你一定很熟悉。

  > 在被测对象中使用了多个注入类的(常见的，router，activatedroute，http，location等)，需要用复制、stub等另外定义，引入后才可以避免编译问题。需要根据编译错误提示来见招拆招。

孤立测试案例

【被测组件介绍】

先说明下，组件使用到的类和函数

ut-isolatedTest1.component.ts的export class ut-isolatedTest1构造函数为

```
constructor(public router: Router, public http: Http,
            public activatedRoute:ActivatedRoute,
            private devcfgservice: DeviceConfigService, private location: Location,private el: ElementRef)`
```

并且在ut-isolatedTest1中使用了以下的类及函数

1.activatedRoute -- activatedRoute.params.subscribe()

```
constructor(public router: Router,  public http: Http,
    public activatedRoute:ActivatedRoute,
    private devcfgservice: DeviceConfigService,
    private location: Location,private el: ElementRef)
    {
        this.activatedRoute.params.subscribe(params => {this.projectidinput = params['projectid'];  this.displaybutton = 0;
    });
    this.mote = initMote
    this.pictureuploadstate = 0;
    }
```

2.http -- http.get()

```
/*用到http.get函数*/
getModelinfo() {
    let that=this;
    that.http.get('/iot/service/motemodelsbasicinfo?owner='+window.sessionStorage.username)
           .map((res:Response) => res.json())
              .subscribe((res: Response) => {
              that.moduleobjectrep = res.json();
              that.modelinfoList = that.moduleobjectrep.motemodelsbasicinfo;
        });
    }
```

3.自定义服务 -- devcfgservice.create()

```
略
```

####【测试代码编写】####

被测组件有几类注入服务以及自定义的服务，这些服务有的有构造函数，有的没有构造函数。

独立的单元测试要通过new的方式创建被测对象的组件类实例，因此，在没有构造函数的类的stub中需要增加构造函数。

1 根据类的入参，定义相关文件和类

1.1 route桩文件--routestub.ts

```
import { Router } from '@angular/router';
export class RouterStub extends Router{
constructor()
{
    this.navigate('default');
}
navigate(url: any) { return url; };}
```

1.2 http桩文件--httpstub.ts（仅供参考，可以有多种写法）

```
import {Http} from "@angular/http";
export class HTTPStub extends Http{
constructor()
{
    this.getBaseHref('ithings');
}
    getBaseHref(url: any) { return url; };
}
```

1.3 activatedroutes桩文件--activatedroutestub.ts

```
/*（按照原声明重载）*/
import {Params} from "@angular/router";
import {ObservableStub} from "./ObservableStub";

export class ActivatedRouteStub {
/**
 * The matrix parameters scoped to this route. The observable will emit a new value when
 * the set of the parameters changes.
 */
params:ObservableStub<Params>;
constructor()
{
    this.params= new ObservableStub();
}
}
```

1.4 Observable桩文件--可观察对象ObservableStub.ts

因为被测对象使用了activatedRoute的params，所以ActivatedRouteStub调用的ObservableStub也是重载来的。

```
import {Subscribable} from "rxjs/Observable";
import {Subscription} from "rxjs/Rx";
export class ObservableStub<T> implements Subscribable<T> {
    constructor()
    {};
    subscribe(): Subscription
    {};
}
```

1.5 location桩文件--locationstub.ts(export LocationStub)

```
 /*类似写法，略*/
```

> - 这几个构造函数与库提供服务的构造函数不同，且用到了服务的函数或变量，所以需要写复制品，并且import到测试组件spec.ts中。

> - DeviceConfigService 自定义的构造函数没有参数，且比较简单，只需要引入即可，不需要写复制品。
> - route http类都有构造函数，因此用new，而ActivatedRoute没有构造函数，所以需要重载成有构造函数的类。

2 在ut-isolatedTest1.component .spec.ts的beforeEach()中，通过new的方式创建被测对象的组件类实例

```
beforeEach(() => adddevicecomp = new Ut-isolatedTest1Component(new RouterStub(),new HTTPStub(),new ActivatedRouteStub()),new Devic
```

3 测试用例it编写 --直接使用对象实例

```
/*以下代码为参考代码，没有对象代码是运行不了的，你只要关注写法*/
it('#adddevice-0001 孤立测试 submitAddDevcie() accessmode should  to "255"', () => {
    DevaccessComponentInstance.mote.accessprotocol = 1;
    DevaccessComponentInstance.mote = initMote;
    DevaccessComponentInstance.submitAddDevcie();
    expect(DevaccessComponentInstance.mote.appeui).toBe('255');
});
```

> - 孤立测试只关心被测对象逻辑，没有DOM操作，不需要进行测试准备。

编完，运行angular框架带的相关命令即可测试。如angular-seed 的npm test就可以测试。出现以下summary即表明测试成功。

框架如何配置以进行测试，另文描述。

一些可能错误的FAQ

---

1. 复制服务后，测试结果ok但出现 ALERT：false什么意思？

   如果被测组件中有alert语句，运行到时会由此输出，不影响测试结果。

2. 如果没有测试用例，运行结果也会提示错误

   npm ERR! Test failed. See above for more details.

3. spec测试文件new stub类的时候，如何保证stub与被测对象类型一致

   建议采用XXXStub extends XXX的写法

恭喜你，可以开始编写孤立代码测试，也意味着你进入了一次次掉坑、填坑的痛并快乐的旅途中。

后记：入门第二篇，后续还有七八九篇。