ANGULAR TEST

مرحله اول :

https://angular.io/guide/testing-code-coverage

https://angular.io/guide/testing-services#angular-testbed

مرحله دوم :

متد های **jasmin** :

https://jasmine.github.io/api/edge/global.html

https://jasmine.github.io/api/edge/matchers.html

مرحله سوم:

متد های فایل **DataGenerator.ts** :

از متد های این فایل برای تولید دیتای رندوم با هر **typeof** استفاده می شود.

---

randomNumber()

تولید اعداد رندوم

---

randomString()

تولید رشته از حروف به صورت رندوم

---

randomBoolean()

تولید مقدار بولین به صورت رندوم

---

randomDate()

تولید تاریخ رندوم

---

randomArray(array)

تولید آرایه ای از مقادیر رندوم

```
randomArray(['string', 'number', 'string']);
```

```
▶ (3) ['zkgbwrafihwelnvbilxs', 26, 'vwbtprsjoprfxndyidqg']
```

```
randomArray(['string', 'number', 'string', {hamze : 'string'}]);
```

```
▼ (4) ['xtmvpbgtwtpabuormcae', 48, 'upwwtaiuazrigbipjguu', {…}]
    0: "xtmvpbgtwtpabuormcae"
    1: 48
    2: "upwwtaiuazrigbipjguu"
  ▶ 3: {hamze: 'xveytvepurqomiqtqach'}
    length: 4
  ▶ [[Prototype]]: Array(0)
```

```
randomArray(['string', 'number', 'string', [{hamze : 'string'}, 'date']]);
```

```
▼ (4) ['gclcqawkzkceyrtehitj', 44, 'hebfwzixftiqylhrcpad', Array(2)]
    0: "gclcqawkzkceyrtehitj"
    1: 44
    2: "hebfwzixftiqylhrcpad"
  ▼ 3: Array(2)
    ▶ 0: {hamze: 'hhtihrjthcjnpjrcmdby'}
      1: "2022-11-18T11:51:16.141Z"
      length: 2
    ▶ [[Prototype]]: Array(0)
    length: 4
  ▶ [[Prototype]]: Array(0)
```

---

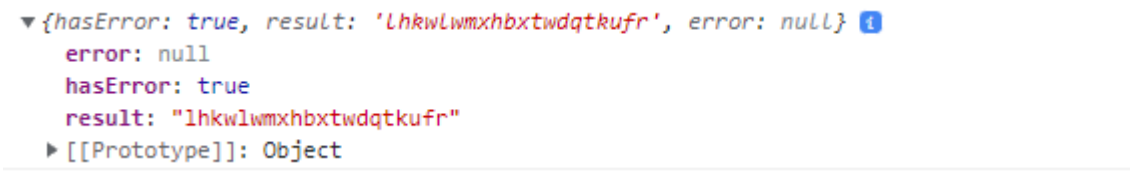
mockResult(object, count =1, properties = {})

تولید مقادیر رندوم برای پراپرتی های یک آبجکت (پارامتر اول).

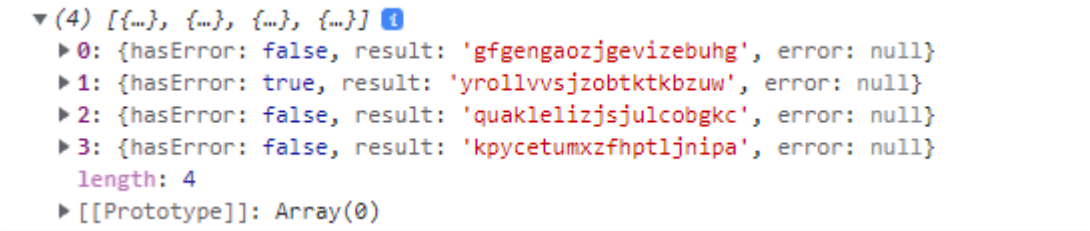
پارامتر دوم به صورت پیش فرض 1 است یعنی خروجی این تابع یک آبجکت با دیتای رندوم می باشد در صورتی که عدد پارامتر دوم بزرگتر از 1 باشد خروجی به آرایه ای از آبجکت ها تبدیل می شود.

پارامتر سوم پراپرتی های موجود در پارامتر اول می باشد که می خواهیم برای آنها به صورت دستی یک مقدار تعیین کنیم نه به صورت رندوم.

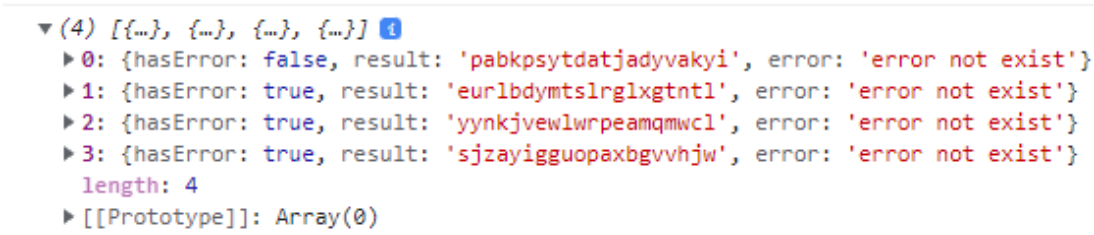
```
mockResult({hasError : 'boolean', result : 'string', error : null});
```




```
mockResult({hasError : 'boolean', result : 'string', error : null}, 4);
```




```
mockResult({hasError : 'boolean', result : 'string', error : null}, 4, {error : 'error not exist'});
```



---

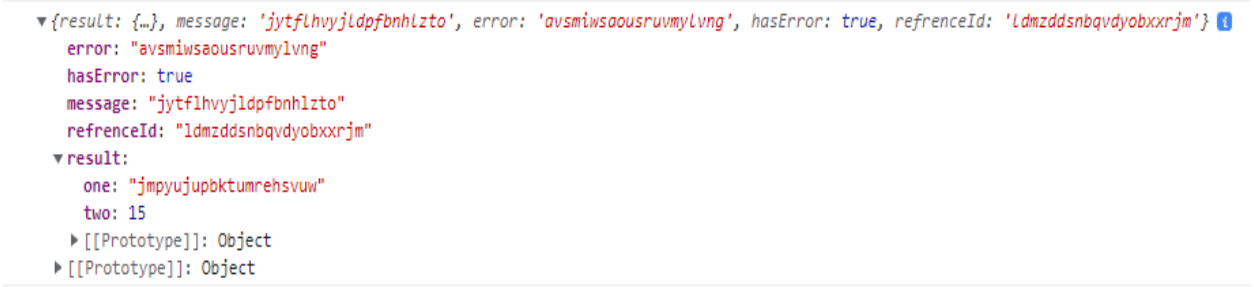
mockTServiceResult(object, properties = {})

تولید مقادیر رندوم برای پراپرتی های یک آبجکت (پارامتر اول).

پارامتر دوم پراپرتی های موجود در پارامتر اول می باشد که می خواهیم برای آنها به صورت دستی یک مقدار تعیین کنیم نه به صورت رندوم.

مدل خروجی : TServiceResult<object>

```
mockTServiceResult({one : 'string', two : 'number'});
```




```
mockTServiceResult({one : 'string', two : 'number'}, {one : 'random one', hasError : false});
```

---

mockPageListResult(object, count =1, properties = {})

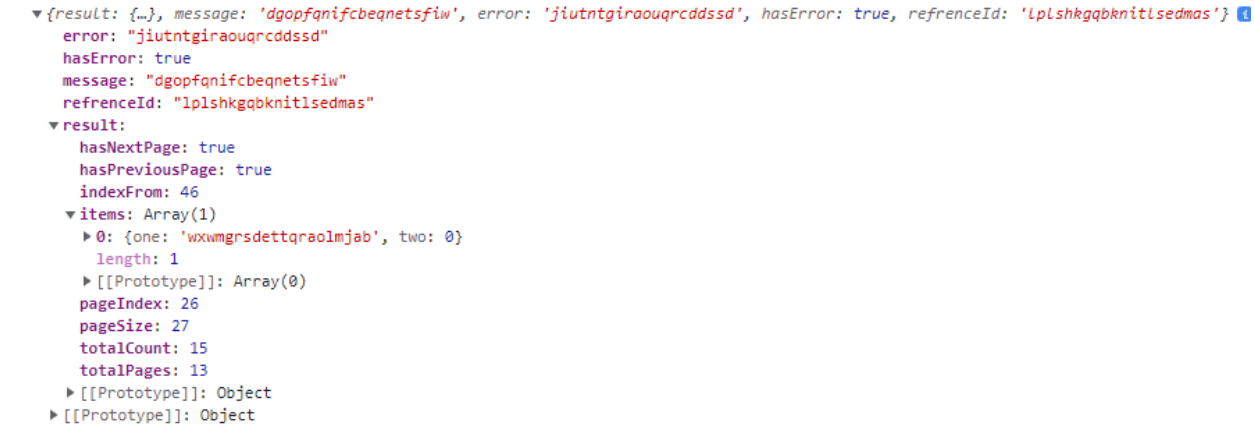
تولید مقادیر رندوم برای پراپرتی های یک آبجکت (پارامتر اول).

پارامتر دوم به صورت پیش فرض 1 است یعنی آرایه خروجی تک عضوی می باشد.

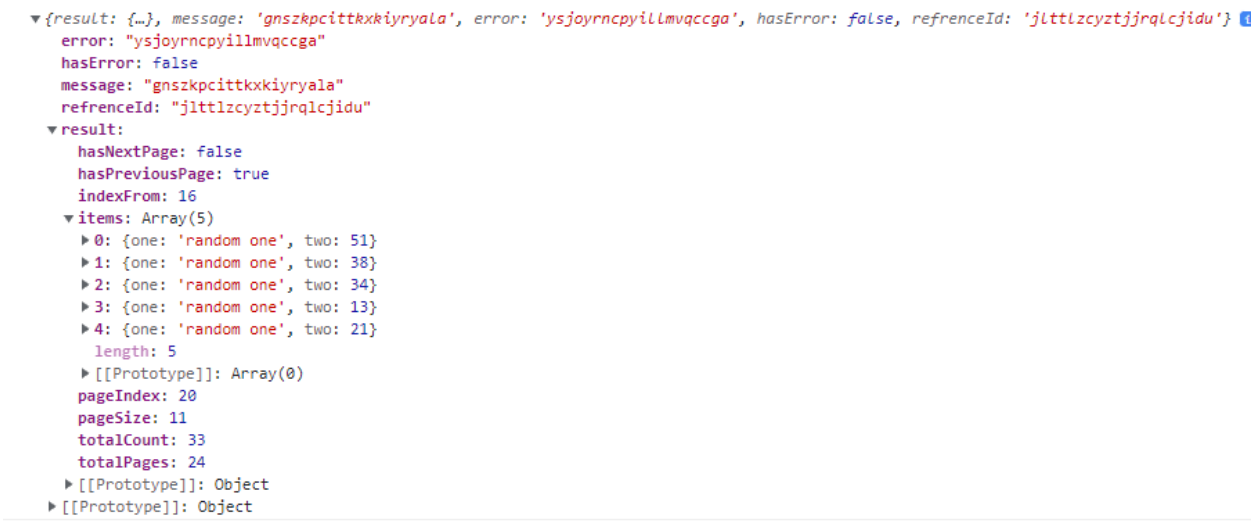
پارامتر سوم پراپرتی های موجود در پارامتر اول می باشد که می خواهیم برای آنها به صورت دستی یک مقدار تعیین کنیم نه به صورت رندوم.

مدل خروجی : TServiceResult<PageList<object>>

```
mockPageListResult({one : 'string', two : 'number'});
```



```
mockPageListResult({one : 'string', two : 'number'}, 5, {one : 'random one', hasError : false});
```



مرحله چهارم :

متد های فایل **ElementsAccessor.ts**

از متد های این فایل برای تست کردن DOM استفاده می شود.

---

sumitForm(fixture, selector)

ارسال فرم.

پارامتر دوم سلکتور المنت مورد نظر می باشد (نام تگ ، id ، نام کلاس)

```
submitForm(fixture, 'form');
```

---

eventElement(fixture, selector, event, parameters = null)

صدا زدن event روی المنت مورد نظر.

پارامتر سوم نام event مورد نظر می باشد.

اگر تابع روی این event پارامتر داشته باشد پارامتر چهارم را مقداردهی می کنیم.

```
eventElement(fixture, '#input-search-book', 'focusin');
```

```
eventElement(fixture, '#start-date > input', 'dateChange', { value: date.toString() });
```

---

getElement(fixture, selector)

دسترسی به اولین المنت با سلکتور مورد نظر در html

```
expect(getElement(fixture, '.navbar')).toHaveClass('hide-navbar');
```

```
expect(getElement(fixture, 'section.activity-container')).toBeTruthy();
```

---

getAllElements(fixture, selector)

دسترسی به تمامی المنت ها با یک سلکتور خاص در html (تمامی المنت ها را به صورت یک آرایه به ما بر میگرداند).

```
let elemetns = getAllElements(fixture, '.card-book');
```

```
getAllElements(fixture, '#filter-visible > span')[2].nativeElement.click();
```

---

clickedElement(fixture, selector)

کلیک کردن اولین المنت با سلکتور مورد نظر.

```
clickedElement(fixture, '#exit-btn');
```

---

getAllClasses(fixture, selector)

دریافت تمامی کلاس های اولین المنت با یک سلکتور خاص به صورت یک رشته.

```
expect(getAllClasses(fixture, 'section.section')).toContain('p-2');
```

```
expect(getAllClasses(fixture, '#popular-books')).not.toContain('align-items-between d-flex flex-wrap');
```

classElement(element, className)

اگر پارامتر دوم در یک کلاس از کلاس های پارامتر اول باشد true را برمیگرداند در غیر اینصورت false.

پارامتر مورد نظر المنت مورد نظر می باشد که میتوان از getElement یا getAllElements استفاده کرد.

پارامتر دوم نام کلاس مورد نظر برای جستجو می باشد.

```
let elements = getAllElements(fixture, '.book-list');
expect(classElement(elements[0], 'border-top')).toBeFalse();
expect(classElement(elements[1], 'border-top')).toBeTrue();
```

contentElement(fixture, selector)

محتوای المنت موردنظر با یک سلکتور خاص را برمیگرداند. همراه با این تابع بهتر است از تابع trim() هم استفاده شود.

```
expect(contentElement(fixture, '#search-step-title').trim()).toEqual('جستجوی کتاب');
```

```
expect(contentElement(fixture, '.created-at')).toContain(jalaliPipe.transform(component.myListPackages[0].createdAt));
```

getAttributeContent(fixture, selector, attributes : string | string[])

دریافت مقادیر اتریبیوت های یک المنت با سلکتور خاص.

پارامتر سوم نام یک اتریبیوت (به صورت string) یا اتریبیوت ها (به صورت آرایه) می باشد. اگر به صورت آرایه باشد خروجی تابع به صورت یک آبجکت از نام اتریبیوت ها و مقادیر آنها می باشد.

```
expect(getAttributeContent(fixture, '.header-popular-books > a', 'href')).toEqual(htmlUrls.link);
```

```
let attributes = getAttributeContent(fixture, '.logo-box-desktop > img', ['src', 'width', 'height', 'alt']);
expect(attributes.src).toEqual('../../../assets/Images/Logo.svg');
expect(attributes.width).toEqual('162px');
expect(attributes.height).toEqual('43px');
expect(attributes.alt).toEqual('بهخوان');
```

getStyleElement(fixture, selctor)

دریافت استایل های یک المنت با سلکتور خاص.

خروجی این تابع به صورت یک آبجکت از نام استایل ها و مقادیر آنها می باشد.

```
let styles = getStyleElement(fixture, '.not-publisher-link > span:first-child');
expect(styles.width).toEqual('18px');
expect(styles.height).toEqual('18px');
```

```
expect(getStyleElement(fixture, '.search-icon').display).toEqual('none');
```

---

changeInputValue(fixture, selector, value)

تغییر مقدار input در html.

```
changeInputValue(fixture, '#search-input', randomString());
```

```
changeInputValue(fixture, '#search-input', '');
```

---

getValueInput(fixture, selector)

دریافت مقدار input در html.

```
expect(getValueInput(fixture, 'input')).toEqual('');
```

```
expect(component.contactUsModel.description).toEqual(getValueInput(fixture, 'textarea'));
```

---

changeFormValue(fixture, formName, values : {})

تغییر مقادیر فرم.

```
changeFormValue(fixture, component.contactUsForm, {message : 'message'});
```

```
changeFormValue(fixture, component.signIn_form, { userName: 'sdflksdj', userPassword: '۰۱۲۲۳۴۵۶۷' });
```

مرحله پنجم :

متدهای فایل **mockDependencies.ts:**

از متد های این فایل برای ماک کردن dependency ها استفاده می شود.

---

setSpy(config)

روش اول : ماک کردن dependency های کامپوننت.

نحوه تعریف کردن در داخل describe() :

```
spy = setSpy({
  MatDialog: { methods: ['open'] },
  MatBottomSheet: { methods: ['open'] },
  ActivatedRoute: { methods: [], properties: { queryParams: of(undefined) } },
  Router: {methods: ['navigate']}
});
```

هر dependency که ماک می شود باید به صورت زیر provide شود :

```
await TestBed.configureTestingModule({
  declarations: [WalletComponent],
  imports: [WalletModule, HttpClientTestingModule, RouterTestingModule],
  providers: [
    {
      provide: MatDialog, useValue: spy['MatDialog']
    },
    {
      provide: MatBottomSheet, useValue: spy['MatBottomSheet']
    },
    {
      provide: ActivatedRoute, useValue: spy['ActivatedRoute']
    },
    {
      provide: Router, useValue: spy['Router']
    }
  ]

})
  .compileComponents();
```

نحوه استفاده از dependency ماک شده :

```
expect(spy['MatDialog'].open).toHaveBeenCalledOnceWith(ActivePackagesDialogComponent,{
  width: '474px',
  height: '316px',
});
```

---

mockSpyOn(object, methodNames : string | string[])

روش دوم : ماک کردن dependency های کامپوننت.

پارامتر اول نام dependency می باشد .

پارامتر دوم متد هایی از آن dependency می باشد که می خواهیم ماک کنیم.

```
router = mockSpyOn(Router, 'navigate');
```

```
repo = mockSpyOn(LocalRepository, ['remove', 'getItem', 'setItem']);
```

---

mockSpyByReturn(object, methodName, returnValue)

روش سوم : ماک کردن dependency های کامپوننت (از این مورد بیشتر برای ماک کردن سرویس ها درون کامپوننت ها استفاده می شود).

پارامتر اول نام سرویس یا dependency می باشد.

پارامتر دوم نام متد می باشد.

پارامتر سوم یک ریسپانس برای خروجی تابع (پارامتر دوم) می باشد.

```
response = mockPageListResult(currentPackageModel, 5, { hasError: false });
service = mockSpyByReturn(MyPackagesService, 'getMyPackages', of(response));
```

---

mockDialog(returnValue = true)

برای ماک کردن دیالوگ هایی که بعد از باز کردن آنها afterClosed به کار رفته است.

پارامتر اول به عنوان خروجی afterClosed استفاده می شود.

نحوه تعریف کردن :

```
let dialog = mockDialog();
```

نحوه استفاده از دیالوگ ماک شده :

```
expect(dialog.open).toHaveBeenCalledWith(ConfirmCancelDialogComponent, { data: { config, }, panelClass: 'mat-dialog-delete', });
```

---

mockBottomSheet(returnValue = true)

برای ماک کردن باتم شیت هایی که بعد از بازکردن آنها afterDismissed به کار رفته است.

پارامتر اول به عنوان خروجی afterDismissed استفاده می شود.

نحوه تعریف کردن :

```
let bottomSheet = mockBottomSheet();
```

نحوه استفاده از باتم شیت ماک شده :

```
expect(bottomSheet.open).toHaveBeenCalledWith(ConfirmCancelBottomSheetComponent as any, { data: { config, } });
```

---

injectableDependency(object)

برای نمونه ساختن از یک dependency استفاده می شود.

پارامتر اول شامل یک آبجکت می باشد که کلید های آن اسم دلخواه برای dependency می باشد و مقادیر آن dependency های است که میخواهیم از آنها نمونه بسازیم.

خروجی این تابع هم یک آبجکت با همان کلید های آبجکت ورودی می باشد.

نحوه تعریف کردن در داخل beforeach() :

```
inject = injectableDependecy({
  urlPipe: UrlPipe,
  searchAggrService: SearchAggregatorService,
  imgPipe: ImageSrcPipe,
  bookService: BooksService
});
```

نحوه استفاده کردن :

```
expect(attributes.src).toEqual(inject.imgPipe.transform(component.shopInfo.id, 4));
```

---

httpClientMock()

برای ماک کردن httpClient انگولار به کار می رود و فقط برای تست سرویس ها استفاده می شود.

نحوه تعریف کردن :

```
let httpClientSpy = httpClientMock();
```

نحوه provide کردن :

```
providers: [
  {
    provide: HttpClient,
    useValue:httpClientSpy
  },
]
```

نحوه استفاده کردن :

```
it("1# TEST_call_removePublisherFromBook_SHOULD_return_TserviceResult<string>_with_hasError_FALSE", () => {

  let response: TServiceResult<string> = mockResult(result, 1, { hasError: false, error: null });
  httpClientSpy.delete.and.returnValue(of(response));
        You, 1 second ago • Uncommitted changes
  let itemId = randomString();
  service.removePublisherFromBook(itemId).subscribe(res => {
    expect(res).toBeInstanceOf(Object);
    expect(res.hasError).toBeFalse();
    expect(res.error).toBeNull();
    expect(res.result).not.toBeNull();
  });

  expect(httpClientSpy.delete).toHaveBeenCalledWith(service['apiUrl'] + `/RemovePublisherFromBook/${itemId}`);
});
```

---

setNewActivatedRoute(spy, property)

ست کردن مقدار جدید برای پراپرتی های activatedRout.

نحوه ماک کردن activatedRoute :

```
let spy = setSpy({
  MatSnackBar: { methods: ['open'] },
  ActivatedRoute: { methods: [], properties: { queryParams: of(undefined) } },
});
```

نحوه مقداردهی پیش فرض در beforeach() :

```
  setNewActivatedRoute(spy['ActivatedRoute'], 'queryParams').and.returnValue(of(undefined));
```

نحوه ست کردن مقدار جدید :

```
let queryParams = { pgNum: 55, search: 'searchText' };
setNewActivatedRoute(spy['ActivatedRoute'], 'queryParams').and.returnValue(of(queryParams));
component.ngOnInit();
```

نحوه کامنت نویسی صحیح در بلاک (it) های تست :

تست ها حتما شماره گذاری شوند و توضیح دقیقی در مورد تست نوشته شود (به حروف بزرگ و کوچک دقت کنید)

```
it('13# TEST_search_book_step_SHOULD_create', () => {···
});

it('14# TEST_ngclass_search_book_step', () => {···
});

it('15# TEST_search_result_book_SHOULD_create', () => {···
});
```
