ANGULAR TESTING IN VIGO NG

INHALT

- Tools
- Angular Testing
- Types of Tests
- VIGO NG Testing

TOOLS

KARMA

```
09 04 2019 21:03:33.731:INFO [karma]:
09 04 2019 21:03:33.732:INFO [launcher]:
09 04 2019 21:03:33.741:INFO [launcher]:
09 04 2019 21:03:59.049:WARN [karma]:
09 04 2019 21:03:59.500:INFO [HeadlessChrome 73.0.3683 (Linux 0.0.0)]:

```

KARMA

```
09 04 2019 21:03:33.731:INFO [karma]: Karma v1.7.1 server started at http://0.0.0.0:9876/
09 04 2019 21:03:33.732:INFO [launcher]:
09 04 2019 21:03:33.741:INFO [launcher]:
09 04 2019 21:03:59.049:WARN [karma]:
09 04 2019 21:03:59.500:INFO [HeadlessChrome 73.0.3683 (Linux 0.0.0)]:

```

KARMA

```
09 04 2019 21:03:33.731:INFO [karma]: Karma v1.7.1 server started at http://0.0.0.0:9876/
09 04 2019 21:03:33.732:INFO [launcher]: Launching browser ChromeCustom with unlimited concurrency
09 04 2019 21:03:33.741:INFO [launcher]:
09 04 2019 21:03:59.049:WARN [karma]:
09 04 2019 21:03:59.500:INFO [HeadlessChrome 73.0.3683 (Linux 0.0.0)]:

```

KARMA

```
09 04 2019 21:03:33.731:INFO [karma]: Karma v1.7.1 server started at http://0.0.0.0:9876/
09 04 2019 21:03:33.732:INFO [launcher]: Launching browser ChromeCustom with unlimited concurrency
09 04 2019 21:03:33.741:INFO [launcher]: Starting browser Chrome
09 04 2019 21:03:59.049:WARN [karma]:
09 04 2019 21:03:59.500:INFO [HeadlessChrome 73.0.3683 (Linux 0.0.0)]:

```

KARMA

```
09 04 2019 21:03:33.731:INFO [karma]: Karma v1.7.1 server started at http://0.0.0.0:9876/
09 04 2019 21:03:33.732:INFO [launcher]: Launching browser ChromeCustom with unlimited concurrency
09 04 2019 21:03:33.741:INFO [launcher]: Starting browser Chrome
09 04 2019 21:03:59.049:WARN [karma]: No captured browser, open http://localhost:9876/
09 04 2019 21:03:59.500:INFO [HeadlessChrome 73.0.3683 (Linux 0.0.0)]:

```

KARMA

```
09 04 2019 21:03:33.731:INFO [karma]: Karma v1.7.1 server started at http://0.0.0.0:9876/
09 04 2019 21:03:33.732:INFO [launcher]: Launching browser ChromeCustom with unlimited concurrency
09 04 2019 21:03:33.741:INFO [launcher]: Starting browser Chrome
09 04 2019 21:03:59.049:WARN [karma]: No captured browser, open http://localhost:9876/
09 04 2019 21:03:59.500:INFO [HeadlessChrome 73.0.3683 (Linux 0.0.0)]: Connected on socket ... with id 0

```

KARMA

```
09 04 2019 21:03:33.731:INFO [karma]: Karma v1.7.1 server started at http://0.0.0.0:9876/
09 04 2019 21:03:33.732:INFO [launcher]: Launching browser ChromeCustom with unlimited concurrency
09 04 2019 21:03:33.741:INFO [launcher]: Starting browser Chrome
09 04 2019 21:03:59.049:WARN [karma]: No captured browser, open http://localhost:9876/
09 04 2019 21:03:59.500:INFO [HeadlessChrome 73.0.3683 (Linux 0.0.0)]: Connected on socket ... with id 0
HeadlessChrome 73.0.3683 (Linux 0.0.0): Executed 81 of 424 SUCCESS (0 secs / 18.735 secs)
```

JASMINE

```
describe("a suite", () => {
  it("should add 2 and 2 to 4", () => {
    expect(2 + 2).toBe(4);
  });

  it("should concatenate strings", () => {
    const a = 'someName';
    expect(`this is my name: ${a}`).toBe('this is my name: someName');
  });
});
```

JASMINE MATCHERS

```
// not.(other-matcher)
expect(something).not.toBe(true);

// equals
expect(something).toBe(true);

// equals float
expect(something).toBeCloseTo(expected, precision);
expect(number).toBeCloseTo(42.2, 3);

//contains
expect(array).toContain(anElement);

//error
expect(aFunction()).toThrow(anError);
```

TS-MOCKITO

```
// from ts-mockito docs:
// Creating mock
let mockedFoo:Foo = mock(Foo);

// stub method before execution
when(mockedFoo.getBar(3)).thenReturn('three');

// Getting instance
let foo:Foo = instance(mockedFoo);

// prints three
console.log(foo.getBar(3));

// verify called
verify(foo.getBar(3)).called();
```

ANGULAR TESTING

TESTBED & TESTING MODULE

```
let component: BannerComponent;
let fixture:   ComponentFixture<BannerComponent>;
let h1:        HTMLElement;

beforeEach(() => {
  TestBed.configureTestingModule({
    declarations: [BannerComponent],
    providers: [],
    imports: []
  });

  fixture = TestBed.createComponent(BannerComponent);
  component = fixture.componentInstance;
  h1 = fixture.nativeElement.querySelector('h1');
});+
```

HTTP CLIENT TESTING MODULE

```
TestBed.configureTestingModule({
    //...
    imports: [HttpClientTestingModule]
});
```

```
const req = controller.expectOne(`myBasePath/gfs/${beilage.gfId}/beilagen`);

expect(req.request.method).toBe('POST');
const body: FormData = req.request.body;
expect(body.get('art')).toBe(beilage.art);
//...
req.flush('A', {headers: new HttpHeaders()});
```

TESTBED PROS/CONS

Positives	Negatives
• Dependency injection • Importing modules • Mocking with injection • Less code in tests • Tests module configuration	• Performance

TYPES OF TESTS

SHALLOW TESTS

- Shallow-render
- Template testing
- Component bootstrap testing
- NgOnInit template set-up testing

ISOLATED TESTS

- Controller testing
- Isolated from other units
- Mock every dependency (except ZEntity related things)
- No Angular bootstrap

SERVICE TESTS

- Isolated from other units
- Mock every dependency (except ZEntity related things)
- TestBed vs non-TestBed tests

INTEGRATION/E2E TESTS

- Protractor
- Page-object pattern
- Multiple components and services

VIGO NG TESTING

MOTIVATION

- Cumbersome TestBed
- Big Tests
- Many dependencies
- Similar setups

SETUP

```
// these replace the TestBed.configureTestingModule(...) in be
setupServiceTestBuilder(Service);
setupShallowTestBuilder(Component);
setupIsolatedTestBuilder(Controller);

//configurable
setupShallowTestBuilder(Component, builder => builder
    .withImports(SomeModule)
    .withProviders(MyManager, GeschaeftsfallManager)
    .withForms());
```

TESTCONTEXT

```
// these replace the TestBed.configureTestingModule(...) in bef
forServiceTestContext(context => ...);
forShallowTestContext(context => ...);
forIsolatedTestContext(context => ...);

// zentity can be configured above aswell but this also exists
// in here create ZEntities and save them to the "Store" (map i
setupZentityMock(mockService => ...)
```

TESTING COMPONENTS

```
describe('BeilagenHinzufuegenDialogComponent (Shallow Test)', () => {
  let testContext: ShallowTestContext<BeilagenHinzufuegenDialogComponent>
  let beilagenService: BeilagenService;

  beforeEach(() => {
    beilagenService = mock(BeilagenService);
    when(beilagenService.getBeilagenArtEnum()).thenReturn(of([]));
  });

  setupShallowTestBuilder(BeilagenHinzufuegenDialogComponent, builder =>
    builder
      .withProviders({provide: BeilagenService, useValue: instance(beilagenService)}, GfServiceResponseManager)
      .withDialog()
  );

  forShallowTestContext<BeilagenHinzufuegenDialogComponent>(context => (testContext = context));

  setupZentityMock(mockService => {
    const abc = new BasisGeschaeftsfallUI();
    abc.bestandsystemId = '123';
    mockService.setAtStore(abc);
  });

  it('should create', () => {
    expect(testContext.component).toBeTruthy();
  });
});

describe('BeilagenHinzufuegenDialogComponent (Isolated Test)', () => {
  let comp: BeilagenHinzufuegenDialogComponent;
  let beilagenService: BeilagenService;
```

TESTING SERVICES

```
describe('HttpMessagesService', () => {
  let service: HttpMessagesService;
  let messagingServiceMock: MessagingService;

  beforeEach(() => {
    messagingServiceMock = mock(MessagingService);
  });

  setupServiceTestBuilder(HttpMessagesService, builder =>
    builder.withProviders(
      {provide: MessagingService, useValue: instance(messagingServiceMock)},
    )
  );

  forTestContext(context => {
    service = context.getInstance(HttpMessagesService);
  });

  it('should create', () => {
    expect(service).toBeTruthy();
  });

  it('should check if meldung is fachlich', () => {
    const fachlicheMeldung = (service as any).checkIfMsgIsFachlich({
      id: 'gg',
      text: 'bla',
      schweregrad: 'ERROR',
      marker: ['FACHLICHE_MELDUNG', 'TECHNISCHE_MELDUNG']
    });
    expect(fachlicheMeldung).toBe(true);

```