TDD in Web Applications

examples with framework Angular

Luzern, 31. März 2021

Marco Birrer

ALCOR Academy Training

CSS Versicherung

Angular Services

- easy to test
- can have state or not
- should contain all business logic
- tests are very useful here and possible test driven

Angular Services example: fail, pass, refactor

```
describe(should return no member inital ', () => {
      const service = new FunnyAlcorGroupService();

      const membersCount = service.getMembersCount();

      expect(membersCount).toBe(0);
});
```

```
@Injectable()
export class FunnyAlcorGroupService {
      public getMembersCount(): number {
            return -1;
      }
```

```
describe(should return no member inital ', () => {
      const service = new FunnyAlcorGroupService();

      const membersCount = service.getMembersCount();

      expect(membersCount).toBe(0);
});
```

```
@Injectable()
export class FunnyAlcorGroupService {
      public getMembersCount(): number {
            return 0;
      }
```

```
describe(should return one member after adding one ', () => {
      const service = new FunnyAlcorGroupService();
      service.addMember(new AlcorMember('Alessandro'));

      const membersCount = service.getMembersCount();

      expect(service.getMembersCount()).toBe(1);
});
```

```
@Injectable()
export class FunnyAlcorGroupService {
      …….
      public addMember(member: AlcorMember): void {
      }
```

Angular Services HTTP calls

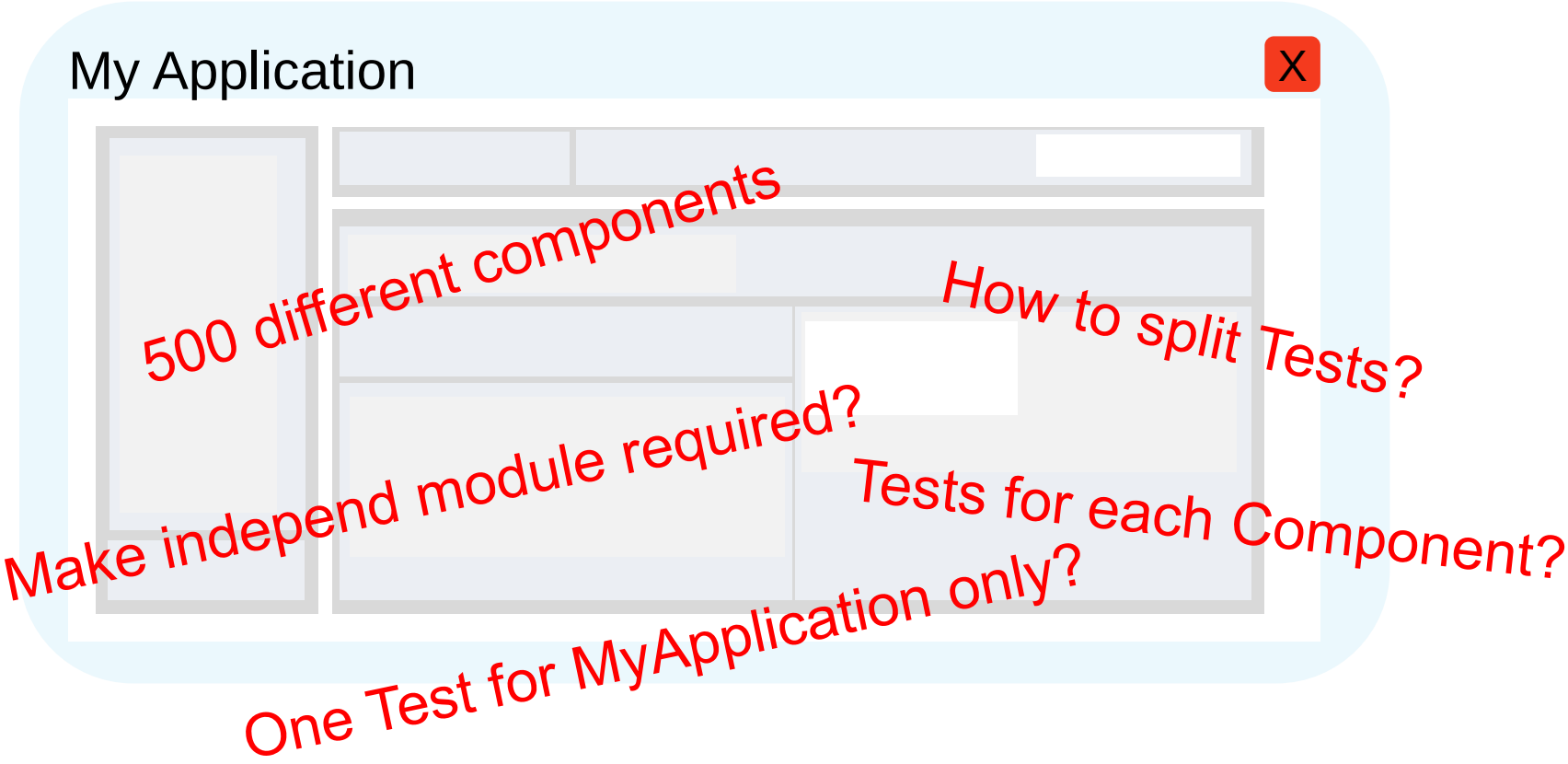
- testing http service important too
- test call was made
- test error handled

```
const funnyAlcorMembers = [ { name: ‘tamas’ } as AlcorMember,  { name: ‘kay’ } as AlcorMember]

describe(FunnyAlcorGroupService ', () => {
    const testee : FunnyAlcorGroupService;
    const http: HttpTestingControler // from angular testing package

    beforeEach(() => {
        TestBed.configureTestingModule({
            imports: [HttpClientTestingModule],
            providers: [FunnyAlcorGroupService]
        });
        testee = TestBed.get(FunnyAlcorGroupService);
        http = TestBed.get(HttpTestingController);
    });

    // some tests testing calls might use
    const request = http.expectOne(‘/alcor/members/’)
    expect(request.request.method).toEqual(‘GET’);
    http.flush(funnyAlcorMembers);
    http.flush(‘ERROR’, { status: 500, statusText: ‘Internal Server Error’ } // case error
    http.verify();
});
```

Angular Components

- Should not cotain business logic
- Need little mor Setup (module where component is placed)
- Services can be mocked (spy)

Angular Components Nested GUI



Angular Component: mock nested components

```
<div>
 ……
 <div *ngFor="let member of (alcorMembers$ | async)">
     <alcor-member
         [member]="member" (stopbedriver)="stopBeDriver($event)">
    </alcor-member>
 </div>
 ……..
```

we want mock away <alcor-member>

```
@Directive({
  selector: 'alcor-member'
})
class MockTaskDirective {
    @Input('member')  public member: AlcorMember;
    @Output('stopbedriver') public stopBeDriverEmitter = new EventEmitter<void>();
}

beforeEach(() => {
    TestBed.configureTestingModule({
        declarations: [TaskListComponent, MockTaskDirective], or
        schemas: [CUSTOM_ELEMENTS_SCHEMA],
        providers: [
            { provide: TasksService, useClass: MockService }
        ]
    });

    fixture = TestBed.createComponent(RootComponent);
    cmp = fixture.componentInstance;
});
```

Create a mock component with same **selector** and need @Input / @Output OR just ignore any error if tag not found with schemas.

My Question: Why do you not practice TDD in UI?

- Is it worth write TDD for styling?
- Does UI change to much after development by Customer review?
- Should UI have much business logic at all?
- Is Performance to bad for TDD excample Cypress?
- We always need mock whole GUI backend without mocking part?

Thank for your attention