Automatsko testiranje

Unit i E2E testovi za angularJS aplikaciju

Testiranje aplikacije

- Zašto praviti automatske testove?
  - Sve veća složenost i veličina aplikacija
  - Ciklus izdavanja nove verzije aplikacije sve manji (nekada se merilo u mesecima, danas u danima)
  - Puno aspekata treba testirati
    - ponašanje aplikacije na različitim web čitačima
    - ponašanje aplikacije kada je koristi više klijenata
    - izgled na ekranima različite veličine i rezolucije
- *“Automate everything that can be automated”*

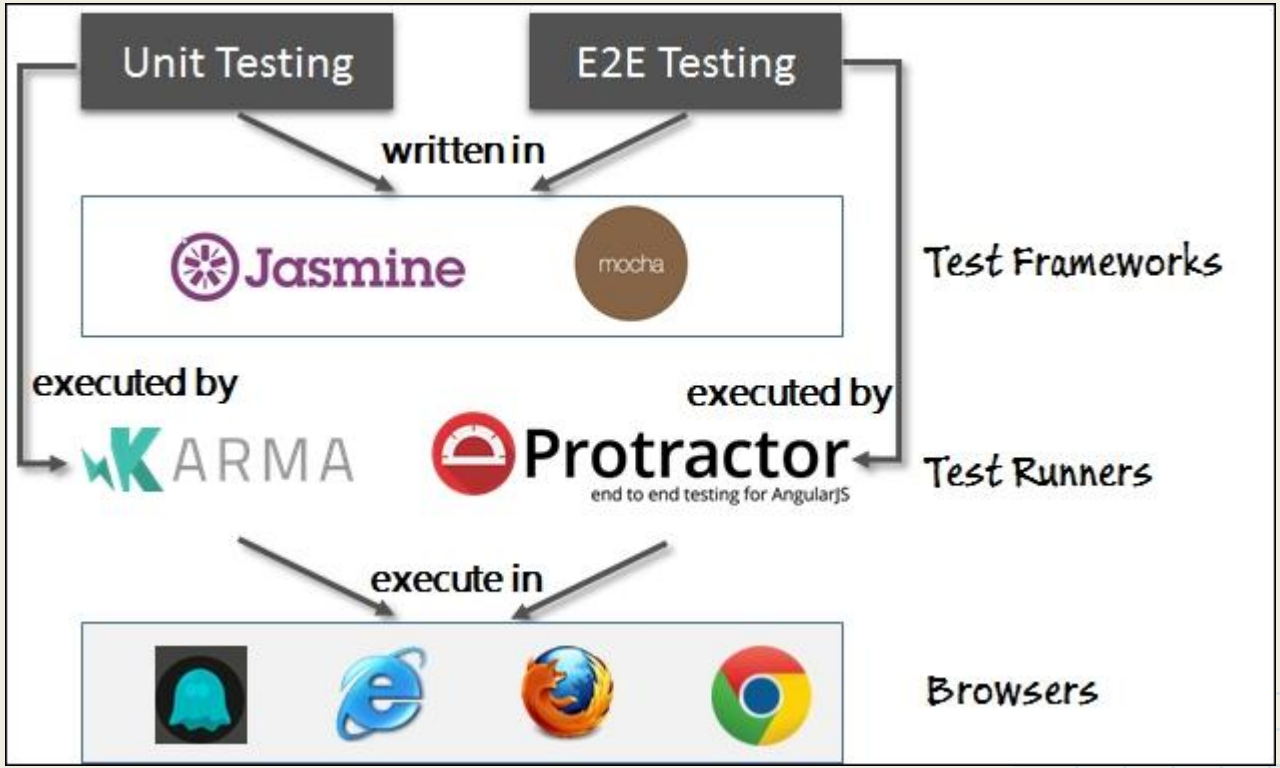
Tipovi testova

- Unit testovi
  - Testiranje komponente u izolaciji kako bi se ustanovilo korektnost ponašanja komponente
  - Većina zavisnosti koje komponenta koristi u testu trebaju da budu zamenjene “mock” implementacijama (veštačka implementacija koja simulira stvarnu) kako bi se osiguralo da ne dolazi do otkaza testa zbog otkaza zavisne komponente
- End-to-end test
  - Izvršavanje aplikacija kao pravi korisnik kako bi se ustanovila korektnost ponašanja cele aplikacije
  - Testiranje se vrši nad sistemom koji radi u pravom web čitaču i proverava se stanje korisničkog interfejsa i sadržaj koji je prikazan

Testiranje aplikacije

- U kom trenutku pisati automatske testove?
  - E2E testovi se pišu kad se razvoj aplikacije završi
  - Unit testovi
    - TDD (test-driven development) - testovi se pišu pre nego što je komponenta implementirana (mana: deo promena zahteva se dešava tokom implementacije, te se testovi moraju ažurirati)
    - Testovi se pišu kada je komponenta implementirana da potvrde ponašanje (mana: testira se implementacija, ne specifikacija)
    - Testovi se pišu dok se komponenta razvija

Tehnologije za automatsko testiranje



Unit testiranje

- Preporuke za pisanje unit testova:
  - Jedan test treba da testira jedno ponašanje (ako test otkaže treba da bude jasno gde je problem)
  - Zavisnosti trebaju da budu mokovane (pored što se time izoluje komponenta ubrza se test jer se ne moraju praviti stvarni pristupi bazi, udaljenim servisima i slično)
  - Test ne treba trajno da menja stanje komponente koju testira
  - Redosled izvršavanja testova treba da bude nebitan
  - Unit test treba da bude brz
  - Treba težiti da se testovima pokriju sve putanje u kodu (grananja, izuzeci)
  - Treba testirati pozitivne i negativne slučaje (testirati i kod koji treba da pukne)

Priprema projekta za unit testove

- U korenskom direktorijumu projekta (gde se nalazi `package.json`) izvršiti

```
npm install karma --save-dev
npm install -g karma-cli  //ovu komandu izvršiti jednom po računaru
```

- Formirati `tests` direktorijum i u njemu izvršiti `karma init`
- Prolaskom kroz wizard dobijamo `karma.conf.js` datoteku koja sadrži konfiguraciju infrastrukture za testiranje
- Ukoliko ne koristimo wizard da formiramo `karma.conf.js` potrebno je ručno instalirati karma dodatak za pokretanje testova u određenom browseru

```
npm install karma-chrome-launcher --save-dev
```

- Na kraju u korenskom direktorijumu treba izvršiti

```
npm install karma-jasmine@2_0 --save-dev
```

- Pokretanje karma testova `karma start tests/karma.conf.js` primer 1

karma.conf.js

- basePath - korenski direktorijum
- frameworks - šta koristimo za pisanje testova (jasmine, mocha,...)
- files - niz datoteka koje moramo učitati da bi unit testovi radili (bitan je redosled učitavanja datoteka!)
- exclude - niz datoteka koje ne želimo da učitamo od onih navedenih u files
- preprocessors - obrađuju datoteka pre nego što ih koristimo za testiranje
- browsers - u kojim web čitačima pokretati testove
- autoWatch - kada izmenimo datoteku iz files automatski pokreće testove iznova

Jasmine - osnove

- Jasmine testovi se pišu putem javascript programskog jezika što znači da imamo pristup svim elementima tog jezika, poput promenljiva, petlji i slično
- Test suite - predstavlja grupu logičko povezanih testova i definiše se sa globalnom funkcijom `describe` koja ima 2 parametra, ime ili opis grupe (koje komponente testiramo) i funkciju u kojoj se navode testovi
- Spec - predstavlja konkretan test koji definišemo putem globalne `it` funkcije koja prima ime ili opis testa i funkciju u kojoj se nalazi jedan ili više `expect` izraza
- Expectation - grade se putem `expect` funkcije koja ima 1 parametar koji predstavlja vrednost, i koja se ulančava sa `matcher` funkcijama
- Matcher - svaka matcher funkcija implementira boolean poređenje između vrednosti koja je prosleđena (putem expect funkcije) i vrednosti koja se očekuje
- Test je uspešan samo ako su svi unutrašnji `expect` izrazi uspešni primer 2

Jasmine - osnove

- Ako grupišemo testove u grupu na smislen način moguće je objediniti kod za pripremu pre svakog testa (`beforeEach` funkcija) i za čišćenje komponente nakon svakog testa (`afterEach` funkcija)
- Moguće je ugnježdavati `describe` blokove tako da se dobija struktura tipa stabla (prvi `describe` blok je koren, dok su `it` blokovi listovi)
- Na ovaj način se hijerarhijski pozivaju `beforeEach` i `afterEach` funkcije ukoliko postoje (kreće se niz stablo od korena do lista i izvršavaju se svi `beforeEach` blokovi, zatim `it` blok, pa onda u obrnutom redosledu `afterEach` blokovi)
- Ukoliko se stavi karakter `x` ispred `describe` funkcije (`xdescribe`) taj blok se ignoriše, a ako se stavi ispred `it` (`xit`) taj test se ne izvršava već je `pending`
- Ukoliko se stavi karakter `d` ispred `describe` funkcije (`ddescribe`) taj blok se jedini pokreće, a ako se stavi `i` ispred `it` (`iit`) taj test se jedini izvršava

Testiranje angular komponenti

- $filter
  - funkcija koja prihvata jedan parametar, ime filtera, i vraća funkciju filtera
- $controller
  - funkcija koja prihvata dva parametra, ime kontrolera i listu zavisnosti, i vraća objekat kontrolera
- $provider
  - sadrži pet funkcija za definisanje servisa (constant, value, service, factory, provider) i radi po istom principu kao funkcije za stvaranje servisa
  - Koristi se za pravljenje mock servisa (najčešće u beforeEach bloku), i ukoliko se zada isto ime kao servis koji se mokuje prilikom injektovanja će se ubaciti mockovana implementacija na mesto spoljnog servisa

primer 3

ngMock

- Sadrži funkcije za podršku testiranja i podrazumevane mock implementacije za određene servise (`$interval`, `$timeout`, `$httpBackened`, itd.)
- `module` globalna funkcija
  - omogućava učitavanje modula i pravljenje mock komponenti
  - prihvata string, objekat ili funkciju kao argument
- `inject` globalna funkcija
  - omogućava injektovanje zavisnih komponenti u test
  - poziva se nakon svih `module` poziva
- `angular.mock.dump` globalna funkcija (`console.log(angular.mock.dump($scope))`)
  - omogućava formiranje preglednog ispisa (pogotovo za scope objekte)
- Za simuliranje prolaska vremena za `$interval` i `$timeout` mock servise poziva se komanda `$interval/$timeout.flush(<broj milisekundi>)` primer 4

Jasmine - špijuni

- Špijuni špijuniraju funkciju i prate kad se ona pozove
- `spyOn(<objekat>, <ime funkcije koja je u objektu>)`
- Kad se izvrši poziv, špijun može da ga propusti originalnoj funkciji, da vrati određenu povratnu vrednost, da ga prosledi drugoj funkciji ili da baci izuzetak
  - `spyOn(obj, "function").and.callThrough();`
  - `spyOn(obj, "function").and.returnValue(123);`
  - `spyOn(obj, "function").and.callFake(function2);`
  - `spyOn(obj, "function").and.throwError("error text");`
- Kad postavimo špijuna da posmatra funkciju izvršavamo test tako što pozivamo kod koji očekujemo da će pozvati funkciju koju špijuniramo, nakon čega vršimo:
  - `expect(obj.function).toHaveBeenCalled();`
  - `expect(obj.function).toHaveBeenCalledWith(456, "some text");`

primer 5

$httpBackend

- Dve grupe metoda koje simuliraju backend:
  - Funkcije za očekivanje zahteva
    - `expectGET, expectPOST, expectPUT, expectDELETE`
    - provere da li će se izvršiti HTTP zahtevi koji su navedeni (i to u navedenom redosledu)
    - opcioni odgovor na zahtev - `expectGET(<url>).respond(<podaci>)`
  - Funkcije za definiciju backenda
    - `whenGET, whenPOST, whenPUT, whenDELETE`
    - postavlja mock backend koji vraća podatke kad se napravi zahtev
    - odgovor na zahtev je obavezan, ali nije obavezno da se izvrši zahtev
- Pošto su HTTP zahtevi asinhroni da ih ne bi čekali `$httpBackened` ima `flush (<broj zahteva>)` funkciju slično kao `$interval`

primer 6

Testiranje direktiva

- Koraci za testiranje direktiva:
  - Formirati HTML fragment koji sadrži direktivu
  - Kompajlirati i povezati HTML za scope
  - Proveriti da li izgenerisan HTML sadrži potrebne atribute (putem jQlite)
  - Ako direktiva stvara scope ili menja stanje scope-a proveriti promene
- Ukoliko treba prilikom testiranja uneti neku vrednost u input element koristi se `$setViewValue(<vrednost>)` funkcija
- Ukoliko imamo asinhrone operacije, a nemamo pristup `flush()` metodi (odnosno ne radimo sa `$interval`, `$timeout` ili `$httpBackend`) koristimo `$scope.$digest()` da garantujemo da se asinhrona operacija izvrši pre nastavka izvršavanja koda
- Testiranje template-a (da li sadrži sve potrebne elemente, itd.) se najčešće vrši putem `karma-ng-html2js-preprocessor`

primer 7

Testiranje rutiranja

- Moguće je testirati konfiguraciju rutiranja kako bi bili sigurni da se postavljaju ispravna putanja, parametri i kontroler kada se promeni adresa
- Ukoliko putanja ima resolve polje moguće je proveriti da li su navedene funkcije stvarno izvršene
- Da bi se izvršila tranzicija sa jedne putanje na drugu potrebno je izvršiti `$rootScope.$digest()`
- Napomena: nezavisno od rutiranja, ukoliko nešto vraća promise potrebno je pokrenuti `$digest` ciklus da bi mogli da izvršavamo `then` blokove

primer 8

Protractor

- Proces E2E testiranja web aplikacije podrazumeva pokretanje aplikacije u pravom web čitaču i proveravanje ponašanja aplikacija spram stanja interfejsa
- Automatizacija korisničke interakcije
- Testovi se pišu u Jasmine radnom okviru, proširenim protractor funkcijama
- U pozadini Protractor spram Jasmine komandi šalje instrukcije Selenium serveru preko HTTPa, nakon kojih Selenium komunicira sa web čitačom i daje mu instrukcije (koristi se *WebDriver Wire Protocol*)
- Instalacija: `npm install -g protractor`
- Ažuriranje Selenium servera: `webdriver-manager update`
- Pokretanje Selenium servera: `webdriver-manager start`
- Selenium server mora biti upaljen sve vreme dok se rade e2e testovi, što se može proveriti na `http://localhost:4444/wd/hub`

Protractor pokretanje i debagiranje

- Protractor takođe koristi konfiguracionu datoteku i bitna polja su:
  - specs - direktorijum u kom se nalaze E2E testovi
  - baseUrl - korenski url aplikacije (da ne navodimo `http://localhost:8080`)
  - seleniumAddress - adresa Selenium servera
- Pokretanje testova: `protractor <putanja do datoteke>/protractor.conf.js`
- Selenium server i običan server moraju biti pokrenuti
- Ako želimo da debagujemo, potrebno je postaviti komandu `browser.debugger()` gde želimo da pauziramo izvršavanje koda, i da pokrenemo protractor u debug režimu: `protractor debug <putanja do datoteke>/protractor.conf.js`
- U tom režimu kada unesemo “c” u terminal skačemo na sledeći `browser.debugger`
- Koristiti i `console.log` za testiranje
- https://github.com/angular/protractor/blob/master/docs/debugging.md

Protractor dodaci na Jasmine radni okvir

- `browser` - globalni objekat koji kontroliše akcije na nivou web čitača
  - `browser.get(<url>)` - otvara url u web čitaču
- `by` - objekat za lociranje elementa na HTML stranici, na osnovu id-a, klase, ng-model, ng-repeat i drugih atributa (preporučeno koristiti angular konstrukcije)
  - `by.id("radnikForm")` - locira element sa `id="radnikForm"`
- `element` - u kombinaciji sa `by` vraća jedan ili više elemenata
  - `element.all(by.className("btn"))` - vraća listu elemenata sa datom klasom
- Nad `element` se izvršavaju funkcije za interakciju sa korisničkim interfejsom
  - `element(by.id("btn1")).click();`
  - `element(by.id("input1")).sendKeys("value1")`
- Protractor API: https://angular.github.io/protractor/#/api

primer 9

E2E testiranje

- Putem protractora je moguće ulančavati lokatore tako da izvučemo elemente/atribute koji nisu jasno definisani (nemaju id, klasu, itd.)
- Ako funkcija vraća promise koristimo `then` funkciju, a ako vraća `ElementFinder` objekat koristimo `element` ili `all`:

```
element.all(by.repeater('emp in employees')).first()
    .element(by.tagName('img')).then(function(img) {
        img.getAttribute('src').then(function(src) {
            expect(src).toMatch(/employees\/profileImage/);
        });
    });
```

- Nađi sve elemente po `ng-repeat="emp in employees"` , uzmi prvi, zatim u okviru tog elementa nađi element `"img"`, zatim u okviru tog elementa nađi atribut `"src"` zatim proveri da li vrednost atributa odgovara šablonu `"employees/profileImage"`

ngMockE2E

- Za testiranje je potrebno obezbediti odvojen test podataka, prvo zato što ne želimo da testovi unose podatke u standardnu bazu, a drugo zato što nam treba izolovan skup podataka koji je uvek isti na početku testa
- Postavka mock backend-a za E2E testiranje:
  - Napraviti nov `app` modul (zvaćemo ga `appe2e`) koji ima zavisnost na `app` modul i `ngMockE2E` modul
  - Promeniti `ng-app` deklaraciju da se vezuje za `appe2e`
  - Definisati mock HTTP implementacije koristeći `$httpBackend.when` funkcije u `appe2e` modulu
- Ako je potrebno propustiti neke zahteve (npr. učitavanje html stranica) koristimo: `$httpBackend.when*(...).passThrough()`

primer 10

Page object pattern

- Osnovna ideja je da se koristi objekat koji će predstavljati HTML stranicu tako što će imati atribute koji predstavljaju elemente stranice (putem `element` i `by` funkcija)
- Na ovaj način izbegavamo korišćenje lokatora u testovima (samo se pozivaju atributi objekta)

primer 11

Gulp i testiranje

- Da bi izvršavali unit testove putem gulp alata potrebno je imati gulp i karmu instaliranu, dodatni gulp dodaci nisu potrebni
- Da bi izvršavali E2E testove potrebno je instalirati, pored protractor i gulp alata, gulp dodataka `gulp-protractor` (`npm install gulp-protractor --save-dev`)
- Da bi se izvršavali E2E testovi potrebno je da budu pokrenuti HTTP i Selenium serveri
- HTTP server pokrećemo preko `gulp-webserver` plugina
- Selenium server pokrećemo putem konzole (ako koristimo u `protractor.conf.js` `seleniumAddress` atribut) ili u sklopu gulp taska (kad koristimo `seleniumServerJar` atribut u `protractor.conf.js`)
- Kad koristimo `seleniumServerJar` treba ga instalirati putem npm-a

```
npm install selenium-server-jar --save-dev
```

primer 12