WARSZTATY Z TESTOWANIA OPROGRAMOWANIA
TESTOWANIE APLIKACJI NAPISANYCH W ANGULARJS

UWAGI OGÓLNE

Testowanie jednostkowe jest techniką, która pomaga deweloperom zweryfikować prawidłowe działanie izolowanych części aplikacji. Testy integracyjne (E2E – *end to end testing*) są natomiast użyteczne wtedy, kiedy chcemy upewnić się, że zbiór zintegrowanych modułów działa w oczekiwany przez nas sposób. AngularJS – nowoczesny framework MVC dla języka JavaScript – oferuje pełne wsparcie zarówno dla testów jednostkowych jak i E2E.

Na pracowni będziemy używać Jasmine – jako frameworka do testów oraz Karmy – jako środowiska uruchomieniowego testów. Jako przykład do pisania testów posłuży nam prosta aplikacja do tworzenia notatek.

Aby móc uruchamiać testy w powyższych technologiach potrzebujemy wykonać następujące kroki:

1. Pobrać i zainstalować Node.js (http://nodejs.org/)
2. Zainstalować Karmę, wykonując z poziomu konsoli polecenie `npm install -g karma`

W spakowanej paczce z pracownią znajduje się aplikacja, którą będziemy testować oraz kilka przykładowych testów. Testy znajdują się w katalogach `test/unit` oraz `test/e2e`. Główny plik HTML to `app/notes.html`. Aby wykonać testy jednostkowe należy wykonać skrypt `scripts/test.bat` (dla użytkowników systemu Windows) lub wykonać z konsoli polecenie `karma start config/karma.conf.js`. Aby wykonać testy e2e należy najpierw uruchomić lokalny serwer mający dostęp do katalogów aplikacji. Najłatwiej zrobić to za pomocą dołączonego prostego serwera napisanego w JavaScripcie, wykonując z poziomu konsoli polecenie `node web-server.js`. Aby uruchomić testy integracyjne wystarczy potem przejść w przeglądarce na adres: http://localhost:8000/test/e2e/runner.html.

Powodzenia,
Mateusz Fedkowicz

ZADANIA DO WYKONANIA

Zadanie 1. (1pkt) Zapoznaj się z kodem aplikacji znajdującym się w pliku `app/js/app.js`. Aplikacji korzysta z jednych z najważniejszych komponentów dostępnych w bibliotece AngularJS – kontrolera, dyrektywy, filtra oraz fabryki serwisów. Zauważ jak serwis `notesFactory` wstrzykiwany jest do kontrolera widoku. Sprawdź także, jak powyższe konstrukcje wykorzystywane są w pliku html aplikacji.

Zadanie 2. (2pkt) W pliku `test/unit/servicesSpec.js` znajdują się testy jednostkowe dla serwisu `notesFactory`. Ponieważ testy jednostkowe nie powinny zależeć od zewnętrznych modułów, przed wykonaniem każdego z testów wstrzykujemy mockową implementację komponentu `localStorage`. Bazując na napisanych już testach, dopisz metody testujące usuwanie jednej notatki, jak i wszystkich notatek. Zauważ, że w tym celu trzeba zamockować funkcje `removeItem` oraz `clear` komponentu `localStorage`.

(Wskazówka – żeby zamockować funkcję `removeItem` *należy zmodyfikować zmienną* `store`*. Należy pamiętać, że jest to obiekt, a nie tablica, dlatego żeby usunąć z niego jakiś wpis najlepiej skonstruować nowy obiekt z interesującymi nas polami, pod obiekt* `store` *podstawić najpierw pusty obiekt - czyli* `{}`*, a następnie nasz spreparowany obiekt)*

Zadanie 3. (2pkt) W pliku `test/unit/controllersSpec.js` znajduje się pojedynczy test kontrolera widoku. Tutaj również korzystamy z techniki mockowania. Mockujemy cały serwis testowany przez nas w poprzednim zadaniu. Twoim zadaniem będzie podobnie jak poprzednio dopisanie do mocka funkcji `remove` a następnie przetestowanie usuwania notatek czyli funkcji `removeNote` z kontrolera `TodoController`.

(Wskazówka – w implementacji funkcji mocka - `remove` *– nie zapomnij o wywołaniu* `rootScope.$broadcast("noteFactoryEvent")`*)*

Zadanie 4. (3pkt) W pliku `test/unit/directivesTest.js` napisz test dyrektywy. Dyrektywa wykorzystywana przez aplikację ustawia tło elementu na zadany kolor (przykład wykorzystania – linia 90 pliku html aplikacji). Aby napisać test potrzebujemy, tak jak w testach kontrolera, wstrzyknąć dostarczany przez Angular'a serwis `$rootScope`. Ponadto potrzebujemy wstrzyknąć standardowy serwis `$compile`. W teście tworzymy element html:

```
elem = angular.element("/kod taga html z wykorzystaniem dyrektywy custom-color/");
```

a następnie go kompilujemy:

```
$compile(elem)(scope);
```

Ostatnią rzeczą, którą chcemy zrobić jest sprawdzenie, czy tło elementu ma zadany przez nas kolor. Wykorzystujemy funkcję `expect(...).toEqual` oraz właściwość `elem.css("/atrybut css/")`, od której oczekujemy, żeby zwracała zadany przez nas kolor (np. `rgb(128,128,128)`).

Zadanie 5. (1pkt) Napisz w pliku `test/unit/filtersTest.js` test filtra ucinającego zbyt długi tekst i dodającego na koniec trzy kropki. Szablon funkcji dla testu filtra jest taki sam jak szablon dla kontrolera lub dyrektywy. Nie potrzebujemy wstrzykiwać do metody testu nic innego oprócz samego filtra. Filtr wstrzykujemy przez jego nazwę i sufix *Filter*, czyli następująco:

```
inject(function(truncateFilter) { …
```

w teście, filtr wywołujemy w następujący sposób:

```
truncateFilter("/jakiś tekst/", /liczba znaków, do której chcemy uciąć tekst/);
```

Zadanie 6. (2pkt) W pliku `test/e2e/scenarios.js` znajduje się jeden test integracyjny, sprawdzający czy dodawanie notatek w aplikacji działa poprawnie. Napisz test sprawdzający, czy także usuwanie notatek działa w oczekiwany sposób. W tym celu można dodać kilka notatek podobnie jak w istniejącym już teście, a następnie za pomocą:

```
elem("/selektor css/").query( …
```

znaleźć przyciski usuwające notatki i wykonać funkcję `click()` na jednym z nich. Na koniec należy sprawdzić czy liczba notatek zmniejszyła się o jeden.

Pomocne linki.

- https://docs.angularjs.org/tutorial - tutorial ze strony Angulara, można się nauczyć instalacji środowisk i podstaw dotyczących testów
- https://docs.angularjs.org/api - oficjalna dokumentacja AngularJS
- http://www.sitepoint.com/unit-and-e2e-testing-in-angularjs/ - tutorial, na podstawie którego powstały zadania na pracownię