Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Askalot meets Harvard Courses at edX

[Askalot2edX]

Metodika pre testovanie

Tím: číslo 6, AskEd
Vedúci tímu: Ing. Ivan Srba
Členovia tímu: Černák Martin, Gallay Ladislav, Hnilicová Eva, Huňa Adrián, Jandura Filip, Žuffa Tibor
Akademický rok: 2015/2016
Autor: Adrián Huňa
Verzia číslo: 2
Dátum poslednej zmeny: 13.12.2015

Úvod

Vývoj funkcionality systému prebieha metódou, kde sú testy vytvárané ešte pred samotnou funkcionalitou. Každý člen tímu testuje časť, na ktorej pracuje.

Účelom tejto metodiky je opísať a zadefinovať základné postupy pri testovaní softvéru. Metodika sa zameriava prevažne na viaceré typy testov od jednotkových (angl. Unit) až po akceptačné testy a ich využitie pre testovanie webových aplikácií. Podobne ako pri metodike pre programovanie, aj táto metodika je silne ovplyvnená konvenciami, ktoré zaviedol predchádzajúci tím naRuby[1]. V predkladanej metodike sa na písanie testov využíva testovací rámec Rspec[2] a jeho nadstavba Capybara[3].

Metodika bola inšpirovaná konvenciami pre formátovanie a písanie testov v Rspec. Je prevažne určená pre tímy využívajúce Rspec v spolupráci s rámcom pre vývoj webu Ruby on Rails. Organizácia tohto dokumentu je inšpirovaná preferovaným postupom pri definovaní testov od akceptačných až po integračné a jednotkové testy.

Kvôli prechodu na modulárnu architektúru pomocou *rails engines* sú aj testy rozdelené do adresárov podľa príslušných *engines*.

[1] http://labss2.fiit.stuba.sk/TeamProject/2013/team13is-si/, [09.12.2015]
[2] https://github.com/rspec, [09.12.2015]
[3] https://github.com/jnicklas/capybara, [09.12.2015]

1. Použité skratky a značky

DSL	Domain Specific Language ● Doménovo špecifický jazyk.
TDD	Test Driven Development ● Prístup k tvorbe softvéru preferujúci písanie testov ako prvý krok implementácie softvéru.
Rspec	Testovací rámec v jazyku Ruby, ktorý definuje intuitívne DSL pre tvorbu testov.
Capybara	Nadstavba nad Rspec pre testovanie webových stránok.
Example	Blok zdrojového kódu v Rspec, ktorý obsahuje samotný test.
Feature	Funkcionalita, ktorá podporuje používateľský príbeh
Mock	Simulovaný objekt, ktorý imutuje správanie reálneho objektu.

2. Postupy

Nastavenie testovacieho prostredia pre Rspec
Najskôr je potrebné nainštalovať potrebné knižnice pomocou príkazu `bundle install`. Pre nastavenie testovacej databázy treba spustiť príkaz `RAILS_ENV=fiit_test rake db:structure:load`, ktorý načíta aktuálnu schému databázy do testovacej databázy podľa prostredia, ktoré sa testuje. Jednotlive existujúce testy spustite v rámci prostredia dekorovaného knižnicou Bundler[4] pomocou príkazu `RAILS_ENV=fiit_test bundle exec rspec specs/shared/features/category_spec.rb`. Posledný parameter je cesta k súboru. Pre spustenie všetkých testov pre konkrétny modul treba spustiť *rake* úlohu `RAILS_ENV=fiit_test bundle exec rake rspec:test`. Podľa nastaveného prostredia sa spustia všetky testy buď pre *`shared`* a *`university`* alebo *`shared`* a *`mooc`* `engine`.

Príprava akceptačných testov pre scenáre používateľského príbehu
Všetky akceptačné testy pre používateľské príbehy (angl. User Story) sa umiestnené v adresári `spec/shared/features`, `spec/mooc/features` alebo `spec/university/features` v závislosti od modulu, ktorého sa týkajú. Konkrétne testy pre jeden používateľský príbeh sú umiestnené v jednom súbore v adresári `features`.

Postup:
1. Vytvoriť súbor pre test používateľského príbehu alebo jeho feature.
   Formát:
   - `<názov feature>_spec.rb`

   Príklad:
   - `authentication_spec.rb`
   - `change_password_spec.rb`
2. Do súboru `<názov feature>_spec.rb` pridať základnú definíciu testu pomocou kľúčového slova `describe`.
3. Textový opis pre `describe` zvoliť na základe názvu používateľského príbehu alebo jeho feature.
   Formát:
   - `describe 'Názov feature'`

   Príklad:
   - `describe 'Authentication'`
   - `describe 'Change Password'`

Vytvorenie opisu akceptačného testu pre scenár používateľského príbehu
Kontext používateľa pri realizácii používateľského príbehu reprezentujeme v Rspec pomocou kľúčového slova `context`. Opis kontextu vyjadrujeme podmienečne pomocou spojky *when* alebo vyjadrením vstupu pomocou predložky *with*. Cieľ testu je reprezentovaný kľúčovým slovom `it` a opis cieľu je vyjadrený vždy v tretej osobe jednotného čísla.

Postup:
1. Pridať do bloku ohraničenom `describe` kontext používateľa pomocou kľúčového slova `context`.
   Formát:
   - `context 'when/with ...'`

[4] https://github.com/bundler, [09.12.2015]

Príklad:
- `context 'when logged in'`
- `context 'with service credentials'`

2. Do bloku `context` pridať cieľ testu pomocou kľúčového slova `it`.
   Formát:
   - `it '<sloveso v tretej osobe jetnotného čísla> ...'`

   Príklad:
   - `it 'allows user to edit profile'`
   - `it 'signs user up'`

V prípade akceptačných testov je `context` vždy vnorený maximálne dvakrát.

Vytvorenie akceptačného testu pre scenár používateľského príbehu
Pre interakciu s webovou stránkou používame rozhranie nástroja Capybara.

Pravidlá pre použitie rozhrania Capybara

- Pre navigáciu na konkrétne URL adresy vždy použiť metódu `visit` s URL adresou ako parametrom.
  Príklad:
  - `visit new_user_registration_path`
- Na simulovanie klikania tlačidiel a hyperliniek použiť metódy `click_button` a `click_link`. Jednotlivé elementy pre uvedené metódy identifikovať vždy len na základe textu, ktorý obsahujú.
- Na vypĺňanie polí vo formulároch použiť vždy metódu `fill_in`. Ako identifikátor formulárového poľa je vždy značka (angl. label) daného poľa.
  Príklad:
  - `fill_in 'user_name', with: 'Peter'`

- Obsah webovej stránky testovať vždy pomocou metódy `have_content`. Testovať vždy len textový obsah stránky.
  Príklad:
  - `expect(page).to have_content('Úspešne prihlásený.)`

- Úspešné nastavenie atribútov v odoslanom formulári testovať vždy len na vyplnených poliach znovu načítaného formulára pomocou metódy `have_field` alebo pomocou `have_content` na obsahu presmerovanej stránky.
  Príklad:
  - `expect(page).to have_field('user_name', with: 'Peter')`
  - `expect(page).to have_content('Peter')`

Vytvorenie inštancií modelov pri testovaní
Pre tvorbu inštancií v modeloch používame knižnicu FactoryGirl[5]. Všetky definície inštancií pre FactoryGirl sú umiestnené bez ohľadu na modul v adresári `spec/factories`. Definície pre model sú uvedené individuálne v súbore s názvom modelu v množnom čísle.

Postup:
1. Vytvoriť súbor pre FactoryGirl definície pre daný model.
2. Vytvoriť definíciu pre model.

[5] https://github.com/thoughtbot/factory_girl, [09.12.2015]

a. Nastaviť v definícii základné parametre pre inštanciu modelu.
b. Ak sa inštancia môže nachádzať vo viacerých stavoch, t.j. vo viacerých kombináciách jej atribútov, definovať viaceré kombinácie pomocou kľúčového slova `trait`.
c. Asociácie na iné modely vždy vyjadriť pomocou kľúčového slova `association`, pokiaľ sa jedná o `belongs_to` asociáciu.
d. Unikátne atribúty, ktoré musia byť odlišné pre všetky vytvorené inštancie, nastaviť vždy pomocou kľúčového slova sequence.
e. Príklad:

```
factory :user do
  sequence(:login) { |n| "user_#{n}" }
  sequence(:email) { |n| "user_#{n}@example.com" }

  password 'password'
  password_confirmation 'password'

  association :school

  trait :as_ais do
    sequence(:login) { |n| "xuser#{n}" }
    sequence(:ais_login) { |n| "xuser#{n}" }
    sequence(:ais_uid) { |n| n }

    password nil
    password_confirmation nil
  end
end
```

3. V testoch vytvoriť inštanciu modelu vždy pomocou metódy `create`. Iné hodnoty atribútov je potrebné špecifikovať pomocou asociatívneho poľa pre metódu `create`.
   Formát: `create :<názov modelu>`
   Príklad:
   - `create :user`
   - `create :user, login: 'samuel', password: 'password'`

Príprava testov pre model

Všetky testy pre modely sa nachádzajú v závislosti od modulu v adresári `spec/shared/models`, `spec/mooc/models`, `spec/university/models`. Testy pre model sú umiestnené v súbore s názvom modelu v jednotnom čísle.

Postup:

1. Vytvoriť súbor pre testy modelu.
   Formát:
   - `<názov modelu>_spec.rb`
   Príklad:
   - `user_spec.rb`
1. Do súboru `<názov modelu>_spec.rb` pridať základnú definíciu testu pomocou kľúčového slova describe a ako opis uviesť triedu modelu.
   Príklad: `describe User`

Vytvorenie jetnotkového testu pre validáciu modelu
Všetky testy pre validácie modelu sa nachádzajú na najvyššej úrovni v describe bloku daného modelu. Opis bloku it vždy obsahu ako prvé slovo *requires* alebo *validates*. Ak validácia kontroluje viaceré typy hodnôt pre daný atribút, kontext testov pre danú validáciu odlíšime pomocou kľúčového slova `context`.
Príklad:

- `it 'requires password'`
- ```
  context 'with AIS credentials' do
    it 'does not require password'
  ```

Vytvorenie jednotkového testu pre povolenia inštancie modelu
Všetky testy pre povolenia inštancie sú umiestnené v describe bloku s názvom *Abilities* na najvyššej úrovni `describe` bloku daného modelu. Každý opis bloku `it` začína vždy slovom *allows* alebo *disallows*. Odlíšenie kontextu pre testy jedného povolenia realizujeme kľúčovým slovom `context`.

Príklad:

- `it 'disallows changing of name'`

Vytvorenie jednotkového testu pre metódu modelu
Všetky testy pre metódy modelu sa nachádzajú na najvyššej úrovni v `describe` bloku daného modelu. Test pre metódu vždy definujeme pomocou describe bloku. Opis `describe` bloku pre metódu vždy obsahuje len názov metódy prefixovaný znakom „#“ v prípade inštančnej metódy a znakom „.“ v prípade metódy na úrovni triedy.

Postup:

- Vytvoriť `describe` blok pre metódu.
  Príklad:
  - `describe '.find_by'`
  - `describe '#activate'`
- Rozdeliť rôzne kontexty alebo typy parametrov metódy pomocou kľúčového slova `context`.
- Pridať opis kontextu v rovnakom formáte ako pri opise akceptačného testu používateľského príbehu.
- Vytvoriť `it` bloky s cieľom testu.

Vytvorenie integračného testu pre servisný objekt
Všetky testy pre servisné objekty sa nachádzajú v závislosti od modulu v adresári `spec/shared/services`, `spec/mooc/services`, `spec/university/services`. Testy pre metódy používajú rovnakú metodiku testovania ako v predchádzajúcej kapitole. Všetky objekty, ktoré spolu interagujú v rámci servisného objektu, majú imitované správanie pomocou syntaxe knižnice Rspec-mocks[6] . Mock objektu sa vždy vytvára kľúčovým slovom `double`. Metódy bez `parametrov` s konštantou návratovou hodnotou vždy uvedieme ako asociatívne pole pre kľúčové slovo double.

Postup:

- Vytvorí `describe` blok pre metódu.
- Vytvoriť mock pre všetky objekty interagujúce v rámci metódy servisného objektu.

[6] https://github.com/rspec/rspec-mocks, [09.12.2015]
  ```

Príklad:

- `service = double(:service)`
- `user = double(:user, name: 'Peter')`

- Priradiť jednotlivé mock inštancie servisnému objektu.
- Zavolať testovanú metódu.