STU FIIT

Testovanie webových aplikácií

MICHAL BARLA

Čo je cieľom testovania

- Nájsť chybu
  - v špecifikácií
  - v návrhu
  - v implementácií
  - ...
- Ukázať, že to funguje
  - Kolegovi
  - **Zákazníkovi** (a dostať zaplatené)
- Zbaviť sa strachu zo zmien
- Sústrediť sa na to, čo je podstatné

Spôsoby testovania

- Manuálne vs. automatizované
- Statické vs. dynamické
- White box/Black box/Grey box

- Testovanie bezpečnosti
- Testovanie výkonnosti
- Testovanie stability
- Testovanie použiteľnosti

Automatizované testovanie

- Čo potrebujem?

- Testovacie dáta – fixtures
- Testovacie prostredie
  - Databáza
  - Server? (Continuous integration)
- Rámec, ktorý zabezpečí spúšťanie a vyhodnocovanie testov

TDD – Motivácia, Cyklus práce

- TDD nás „núti“ napísať na každú *feature* nášho programu testy ***pred*** naprogramovaním samotnej *feature*
  - Pomáha vyjasniť si čo vlastne treba spraviť, kedy sa to dá vyhlásiť za hotové
  - Zabraňuje pozlacovaniu kódu
  - Umožňuje skutočný refaktoring – pokiaľ sú testy zelené, tak sa nemusíte báť meniť svoj kód
- Cyklus
  - Napíš test ➔ Nechaj ho zlyhať (overíš test) ➔ napíš minimálny kód, ktorý prejde testom (a nepokazí iné testy) ➔ upravuj kód bez toho, aby test sčervenal

TDD – úrovne, ukážky

- Testovanie na základných jednotiek programu - Unit test
  - V OOP väčšinou metóda triedy, trieda
  - Napr. testovanie modelu v MVC web. aplikácii
- Funkcionálne testy
  - Testovanie funkčnosti jedného modulu
  - Napr. testovanie controlleru v MVC web. aplikácii
- Integračné testy
  - Testovanie spolupráce modulov
  - Napr. testovanie interakcie controllerov
- Systémové/akceptačné testy
  - Testovanie voči návrhu/požiadavkám

Unit test (java, ruby)

```
public void WordIsEnglishAndSlovakTest(){
  Word word = new Word(“never”, “nikdy”);
  assertEquals(word.eng, “never”);
  assertEquals(word.sk, “nikdy”);
}

def test_word_is_english_and_slovak
  word = Word.new :eng=>'never',:sk=>'nikdy
  assert_equal 'never', word.eng
  assert_equal 'nikdy', word.sk
end
```

Funkcionálny test

```
class UsersControllerTest < ActionController::TestCase
  test "should_get_login" do
    get :login
    assert_response :success
  end

  test "should_login_successfully" do
    referer = 'http://test.host/'
    @request.env['HTTP_REFERER'] = referer
    assert_difference('User.count')  do
      post(:login, {:username => "a", :password => "a"})
    end
    assert_equal "You have successfully logged in.",
    flash[:notice]
      assert_redirected_to referer
  end
end
```

Testovanie webových aplikácií

- Komplexnosť webových aplikácií
  - Úroveň databázy – pohľady, funkcie
  - Aplikačná logika
  - Frontend – javascript, oznamy (flash), presmerovania
- Ako to otestovať?
  - komplexné scenáre, ktoré preveria integráciu všetkých vrstiev
    - uhorka

Vyššie testy (feature, scenár)

```
Scenario: Editor cannot manage page
  permissions
     When I go to the main page
     And I login as "johno"
     And I create "/" page
     And I logout
     And I login as "crutch"
     And page "/" is editable by "crutch"
     When I go to the main page
     Then I should not see "Manage"
     And I should see "Edit"
```

Záver – ako a na čo to použiť

- Testovanie sa dá automatizovať na rôznych úrovniach
- Odbúrava strach meniť a vylepšovať program
- Vylepšuje sledovateľnosť projektu
- Pomáha vo vyjasnení si špecifikácie
  - Ak neviem napísať test, tak musím znova za analytikom
- Upriamuje pozornosť na to, čo je podstatné
  - Nebudem písať viac, ako je potrebné pre zelený test
  - Test-katy