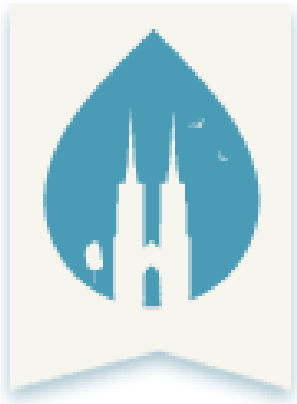
Testujemy Drupala

Czyli o testach jednostkowych, funkcjonalnych i wydajnościowych w Drupal-u

Jarosław Sobiecki

Media Regionalne

13 kwietnia 2013



Kilka słów wstępu:

- Mówimy o testach automatycznych.
- Każdy z omawianych rodzajów testów zasługuje na odzielną prelekcję.
- Ja poświęcę około 10 minut na każdy.
- Niestety będzie trzęsło :)
- To tylko sygnalizacja, wierzchołek góry lodowej.

Agenda na dzisiaj

Testowanie

Proces związany z wytwarzaniem oprogramowania. Jest to jeden z procesów zapewnienia jakości oprogramowania. Testowanie ma na celu weryfikację oprogramowania oraz walidację oprogramowania. Weryfikacja oprogramowania ma na celu sprawdzenie, czy wytwarzane oprogramowanie jest zgodne ze specyfikacją. Walidacja sprawdza, czy oprogramowanie jest zgodne z oczekiwaniami użytkownika.

`http://pl.wikipedia.org/wiki/Testowanie_oprogramowania`

Po co nam automatyczne testy?

- Ręcznie, to znaczy: żmudno, nudno i zawodnie.
- Obrona przed regresją. (Po co robić to samo wiele razy).
- Oszczędność czasu*
- Dobrze zrobiony zestaw testów wykrywa błędy zanim wdrożymy je na produkcję.
- Debugowanie problemów na produkcji? To nigdy nie jest fajne.

Czego się tu bać?

- Spory koszt wejścia
- Sporo narzędzi do ogarnięcia
- Kilka dodatkowych wyzwań (np automatyzacja procesu testowania).
- Testy, to dodatkowy czas dewelopmentu.*
- Testy trzeba utrzymywać. (To także część aplikacji).

Zawsze istnieje trzecia droga :)

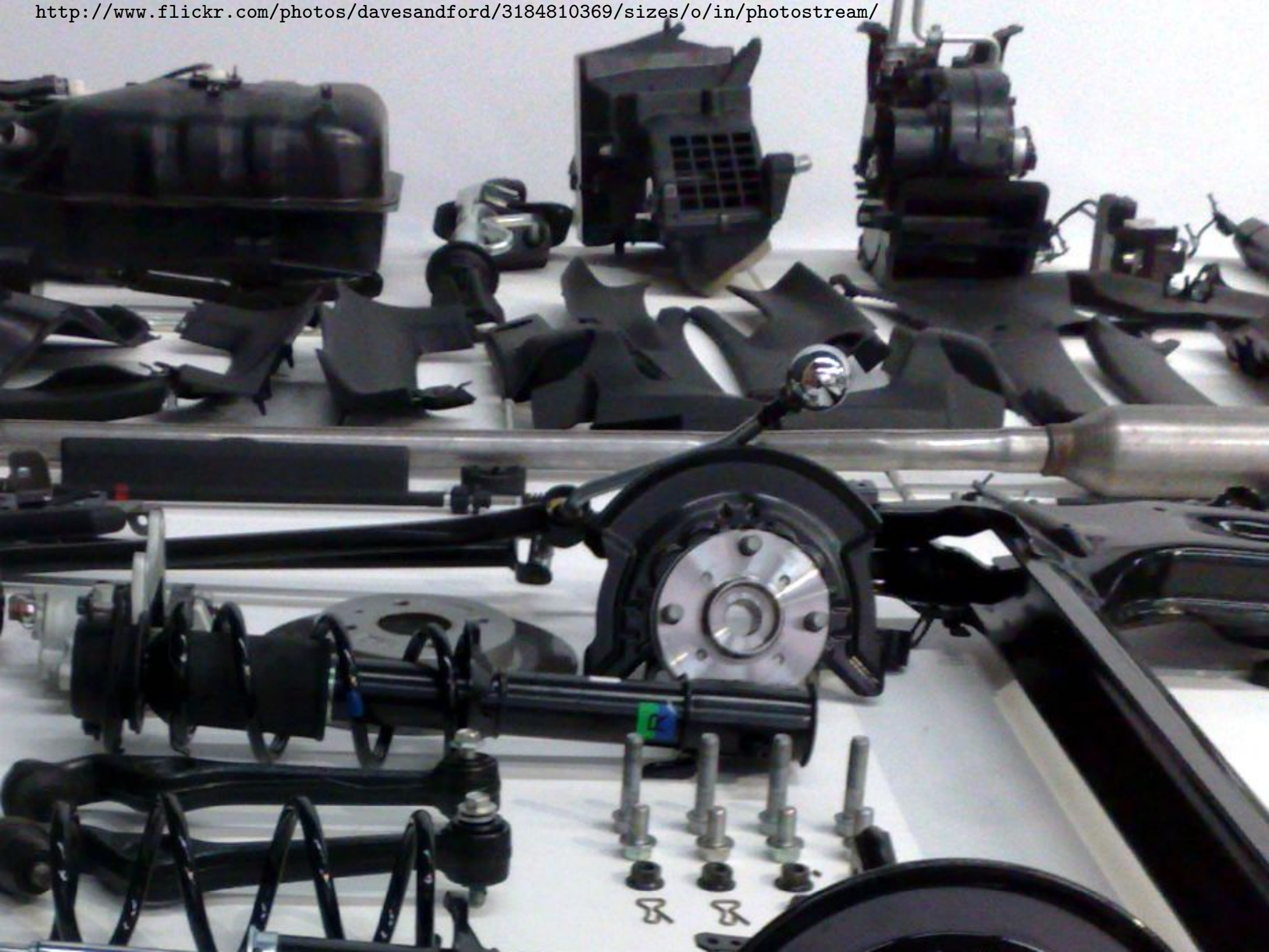
1. Przygotowujemy zmianę w aplikacji.
2. Wrzucamy bez testów na środowisko produkcyjne.
3. Czekamy na telefony.

Definicja

Test jednostkowy (ang. unit test)

W programowaniu metoda testowania tworzonego oprogramowania poprzez wykonywanie testów weryfikujących poprawność działania pojedynczych elementów (jednostek) programu – np. metod lub obiektów w programowaniu obiektowym lub procedur w programowaniu proceduralnym. Testowany fragment programu poddawany jest testowi, który wykonuje go i porównuje wynik (np. zwrócone wartości, stan obiektu, wyrzucone wyjątki) z oczekiwanymi wynikami – tak pozytywnymi, jak i negatywnymi (niepowodzenie działania kodu w określonych sytuacjach również może podlegać testowaniu).

`http://pl.wikipedia.org/wiki/Test_jednostkowy`


http://www.flickr.com/photos/davesandford/3184810369/sizes/o/in/photostream/

- Te testy to testy “white-box”. Mamy dostęp do kodu aplikacji, i testujemy jej komponenty.
- Skupiamy się na działaniu pojedynczego elementu, a nie interakcji między nimi.
- Dobrą praktyką byłoby, gdyby testy te tworzył ten sam programista.
- Powinno testować się przede wszystkim interfejs naszego komponentu (API).
- Te testy muszą być potwarzalne!

Możliwe rozwiązania

- PHPUnit - Bardzo rozbudowany i popularny framework do realizacji testów w świecie PHP-owym. Ale o nim mówić nie będziemy.
- Simpletest - A właściwie fork tego rozwiązania. Framework do realizacji testów funkcjonalnych i jednostkowych, dostosowany do potrzeb testowania Drupala. Wydaje się być trochę przestarzały.

Simpletest - jak to się je?

- Z perspektywy programisty, simpletest do biblioteka do tworzenia testów.
- Tworzenie testów - to implementacja klasy dziedziczącej z klasy bazowej, oraz implementacja magicznych metod (`getInfo()`, `tearDown()`, `setUp`)
- Z pudełka do wyboru mamy dwie klasy bazowe: `DrupalWebTestCase`, oraz `DrupalUnitTestCase`
- Metody, których nazwa zaczyna się od `test`, traktowane są jako przypadki testowe.
- W każdej takiej metodzie implementujemy serię akcji i asercji (czyli testów).

Jak działa DrupalWebTestCase?

1. Tworzona jest nowa, czysta instalacja Drupala (wykorzystanie metody setUp)
2. W razie potrzeby, w metodzie setUp włączamy inne moduły.
3. Wykonanie zestawu testów (metody `test...`)
4. Czyszczenie środowiska (wykorzystanie metody `tearDown`).
5. Ta klasa bardziej pasuje do testów funkcjonalnych, wg schematu: Wykonujemy zapytanie do aplikacji, sprawdzamy, jak się zachowała (jaki kod wygenerowała)
6. To jest naprawdę wolne.
7. Testy są bezpieczne, tzn nie popsujemy sobie bieżącej instalacji.

Jak działa DrupalUnitTestCase

1. Blokowany jest dostęp do bazy danych (a więc testy nie mogą niczego popsuć). Przygotowujemy testy (`setUp`).
2. Wykonanie są kolejne testy (metody `test...`)
3. Dużo szybsze, ale mamy mniejsze możliwości.
4. Tutaj niestety nie możemy korzystać z bazy danych. Ale też niczego nie popsujemy.
5. Takie testy wykonują się bardzo szybko.

Testy jednostkowe - przykład (1)

Chcemy przetestować prostą funkcję z modułu simpletest_example:

```
<?php
/**
 * A simple self-contained function used to demonstrate unit tests.
 *
 * @see SimpletestUnitTestExampleTestCase
 */
function simpletest_example_empty_mysql_date($date_string) {
  if (
    empty($date_string) ||
    $date_string == '0000-00-00' ||
    $date_string == '0000-00-00 00:00:00'
   ) {
    return true;
  }
  return false;
}
```

Testy jednostkowe - przykład (2)

```
public static function getInfo() {
  return array(
    'name' => 'Simpletest Example unit tests',
    'description' => 'Test that simpletest_example_empty_mysql_date works properly.',
    'group' => 'Examples',
  );
}

function setUp() {
  drupal_load('module', 'simpletest_example');
  parent::setUp();
}
```

Testy jednostkowe - przykład (3)

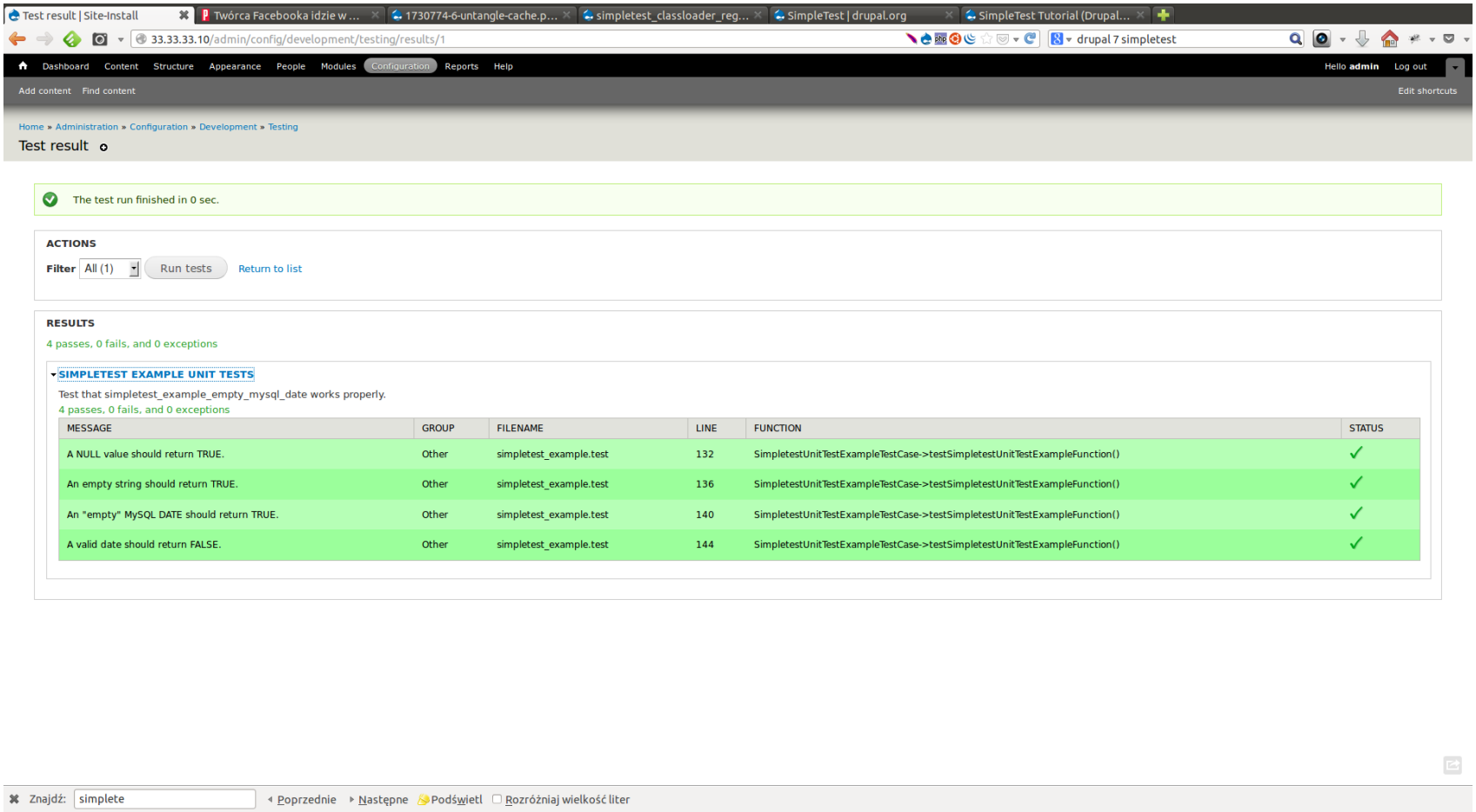
```
/**
 * Call simpletest_example_empty_mysql_date and check that it returns correct
 * result.
 *
 * Note that no environment is provided; we're just testing the correct
 * behavior of a function when passed specific arguments.
 */
public function testSimpletestUnitTestExampleFunction() {
  $result = simpletest_example_empty_mysql_date(NULL);
  // Note that test assertion messages should never be translated, so
  // this string is not wrapped in t().
  $message = 'A␣NULL␣value␣should␣return␣TRUE.';
  $this->assertTrue($result, $message);

  $result = simpletest_example_empty_mysql_date('');
  $message = 'An␣empty␣string␣should␣return␣TRUE.';
  $this->assertTrue($result, $message);

  $result = simpletest_example_empty_mysql_date('0000-00-00');
  $message = 'An␣"empty"␣MySQL␣DATE␣should␣return␣TRUE.';
  $this->assertTrue($result, $message);

  $result = simpletest_example_empty_mysql_date(date('Y-m-d'));
  $message = 'A␣valid␣date␣should␣return␣FALSE.';
  $this->assertFalse($result, $message);
}
```

Simpletest - demo



Uwagi

- Simpletest - w porównaniu z PHPUnit, wydaje się toporny, ale ma swoje zalety. Przede wszystkim lepsza integracja z Drupalem.
- Integracja z systemem obsługi zgłoszeń na http://drupal.org
- Samo DrupalUnitTestCase często nie wystarcza. Można próbować quasi-jednostkowo, czyli testować kod modułu, przy wykorzystaniu DrupalWebTestCase. Ale to nie wystarcza.
- Można się zastanawiać, czy do testów funkcjonalnych nie lepiej użyć Selenium.

http://www.flickr.com/photos/wwarby/3296379139/sizes/l/in/photostream/

Testy wydajnościowe

W inżynierii oprogramowanie, testowanie wydajności to przeprowadzenie ogólnych testów w celu weryfikacji, jak system zachowuje się podczas określonego obciążenia. Taka forma testowania pozwala także na sprawdzenie, jak system się skaluje, oraz jakiś zasobów potrzebuje, by obsłużyć konkretne obciążenie.

Testować, ale czym?

- **siege** - Proste i szybkie narzędzie do generowania obciążenia serwisu.
- **jmeter** - Prawdziwy kombajn. Pozwala tworzyć zaawansowane scenariusze testów.

Siege

Siege, to proste narzędzie, pozwalające na przeprowadzenie podstawowych testów wydajnościowych. Oparte na interfejsie tekstowym. Wyśmienicie się sprawdza przy potrzebie szybszego wygenerowania konkretnego obciążenia na testowanym środowisku. Jak sama nazwa wskazuje, wykonuje **oblężenie** wskazanej witryny. Stosować z ostrożnością!

Instalacja (Ubuntu / Debian)

```
apt-get install siege
```

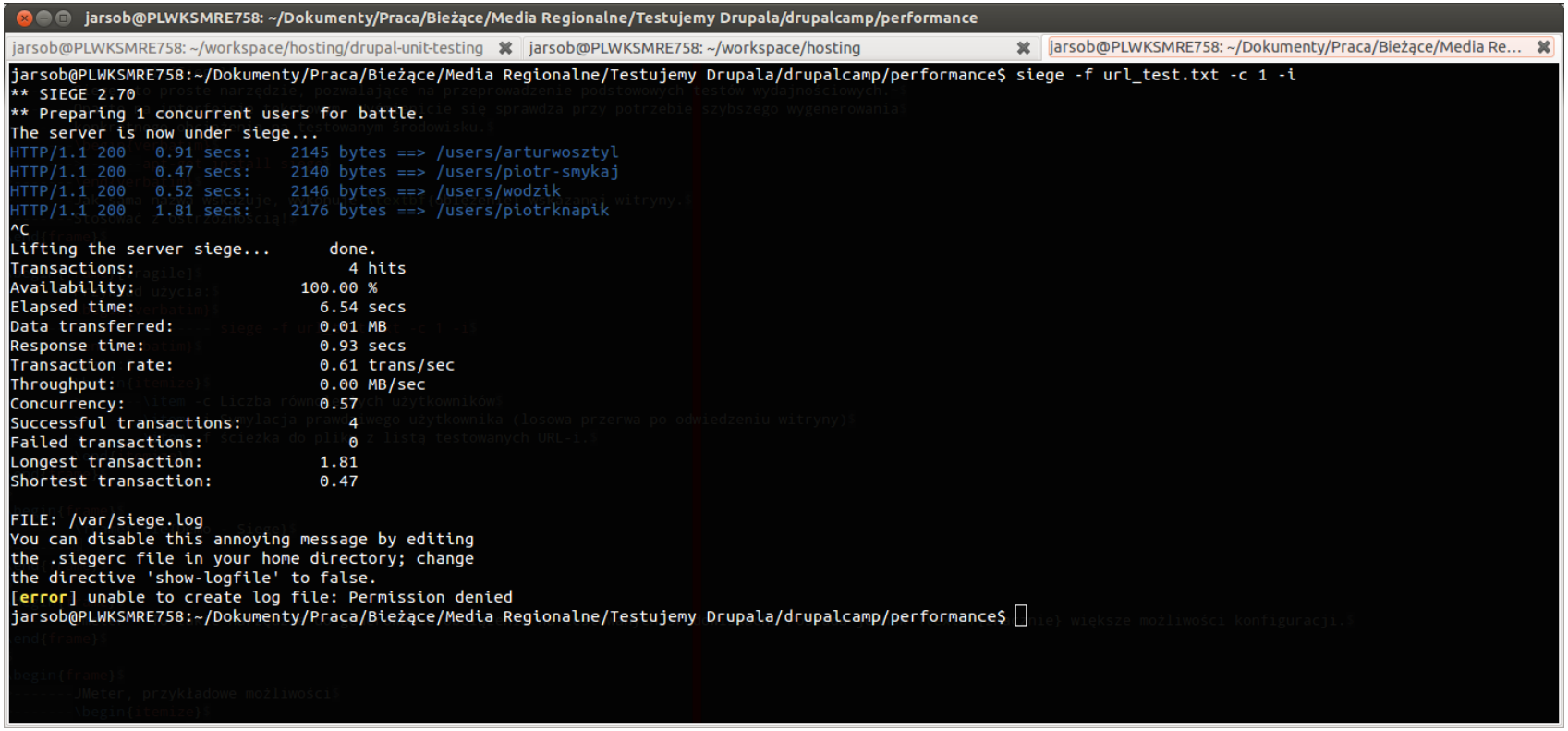
Siege - przykład użycia

```
siege -f url_test.txt -c 1 -i
```

Gdzie:

- -c Liczba równoległych użytkowników
- -i Symulacja zachowania prawdziwego użytkownika (losowa przerwa po odwiedzeniu witryny)
- -f ścieżka do pliku z listą testowanych URL-i.

Demo - Siege


jarsob@PLWKSMRE758: ~/Dokumenty/Praca/Bieżące/Media Regionalne/Testujemy Drupala/drupalcamp/performance
jarsob@PLWKSMRE758: ~/workspace/hosting/drupal-unit-testing
jarsob@PLWKSMRE758: ~/workspace/hosting
jarsob@PLWKSMRE758: ~/Dokumenty/Praca/Bieżące/Media Re...
jarsob@PLWKSMRE758:~/Dokumenty/Praca/Bieżące/Media Regionalne/Testujemy Drupala/drupalcamp/performance$ siege -f url_test.txt -c 1 -i
** SIEGE 2.70
** Preparing 1 concurrent users for battle.
The server is now under siege...
HTTP/1.1 200 0.91 secs: 2145 bytes ==> /users/arturwosztyl
HTTP/1.1 200 0.47 secs: 2140 bytes ==> /users/piotr-smykaj
HTTP/1.1 200 0.52 secs: 2146 bytes ==> /users/wodzik
HTTP/1.1 200 1.81 secs: 2176 bytes ==> /users/piotrknapik
^C
Lifting the server siege... done.
Transactions: 4 hits
Availability: 100.00 %
Elapsed time: 6.54 secs
Data transferred: 0.01 MB
Response time: 0.93 secs
Transaction rate: 0.61 trans/sec
Throughput: 0.00 MB/sec
Concurrency: 0.57
Successful transactions: 4
Failed transactions: 0
Longest transaction: 1.81
Shortest transaction: 0.47
FILE: /var/siege.log
You can disable this annoying message by editing
the .siegerc file in your home directory; change
the directive 'show-logfile' to false.
[error] unable to create log file: Permission denied
jarsob@PLWKSMRE758:~/Dokumenty/Praca/Bieżące/Media Regionalne/Testujemy Drupala/drupalcamp/performance$

Jmeter

Jmeter - To także narzędzie do generowania obciążenia na testowanych środowiskach. Posiada jednak **znacznie** większe możliwości konfiguracji. Opiera się na GUI, ale stanowi tak naprawdę również framework programistyczny (oparty na Javie). Może być również uruchamiany w formie tekstowej, co znacznie ułatwia automatyzację. Pozwala testować aplikacje za pomocą protokołu HTTP, ale nie tylko.

Instalacja (Ubuntu / Debian)

```
apt-get install jmeter jmeter-http
```

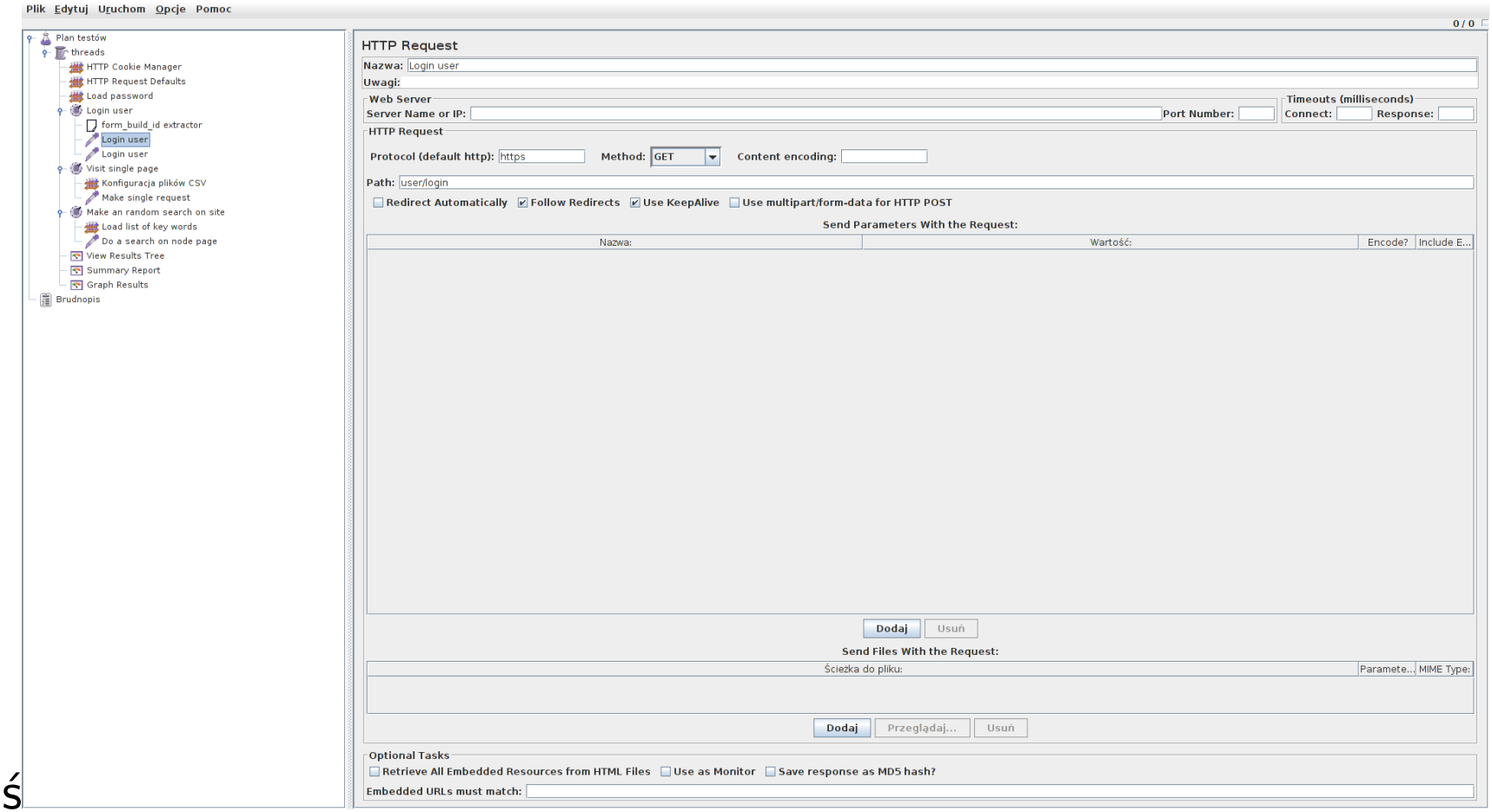
JMeter - jak to się je? (1)

- Scenariusze testowe składają się z elementów konfiguracji, słuchaczy, samplerów, post-procesorów i timery.
- Sampler - ten element pozwala nam wykonywać test (czyli np. zapytanie GET do serwera).
- Post procesor - ten element pozwala wyciągać nam dodatkowe informacje z wyniku działania samplera
- Timery - "udają" zachowanie zwykłego użytkownika. Na ich podstawie po każdym zapytaniu wątek robi sobie losową przerwę (o ustalonym rozkładzie).

JMeter - jak to się je? (2)

- Elementy konfigurujące - te elementy wpływają na działanie innych elementów. Np. Pozwalają na autoryzację po HTTP, obsługę ciastek etc. Czy wczytywanie konfiguracji z plików CSV.
- Słuchacze - te elementy reagują na dane zbierane przez samplery i prezentują je użytkownikowi. Np. Rysowanie wykresów, przedstawienie czasu odpowiedzi w tabelce etc.

Demo - JMeter



Ś

Uwagi

- Pamiętaj, że warunki testowe muszą być powtarzalne.
- Co może zaburzyć testy?
  - Sieć (problemy z łącznością)
  - Rozmiar danych (Np. więcej artykułów w witrynie może spowodować spadek wydajności).
  - Czynniki losowe (np. czy teraz nie wykonuje się backup?)
  - Czy te środowisko testowe jest używane tylko przez Ciebie.
  - Czy konfiguracja Drupal-a jest taka sama?


http://www.flickr.com/photos/24539119@N07/5878811403/sizes/l/in/photostream/

Definicja

Testy funkcjonalne (testy czarnej skrzynki)

Testy czarnej skrzynki to weryfikacja funkcjonalności (np. tego co robi aplikacji) bez dostępu do detali implementacji i mechanizmów aplikacji (w przeciwieństwie np. do testów jednostkowych).

Testowanie aplikacji WWW - jak to wygląda ręcznie

- Żmudne “przeklikiwanie” aplikacji w poszukiwaniu błędów.
- Klikamy, wprowadzamy dane i sprawdzamy co się stanie.
- Być może, mamy jakiś formularz testów do wypełnienia.
- Nikt tego nie lubi. To naprawdę nie działa dobrze.
- Jeżeli robisz coś powtarzalnego i niezbyt ciekawego, łatwo się pomylić.

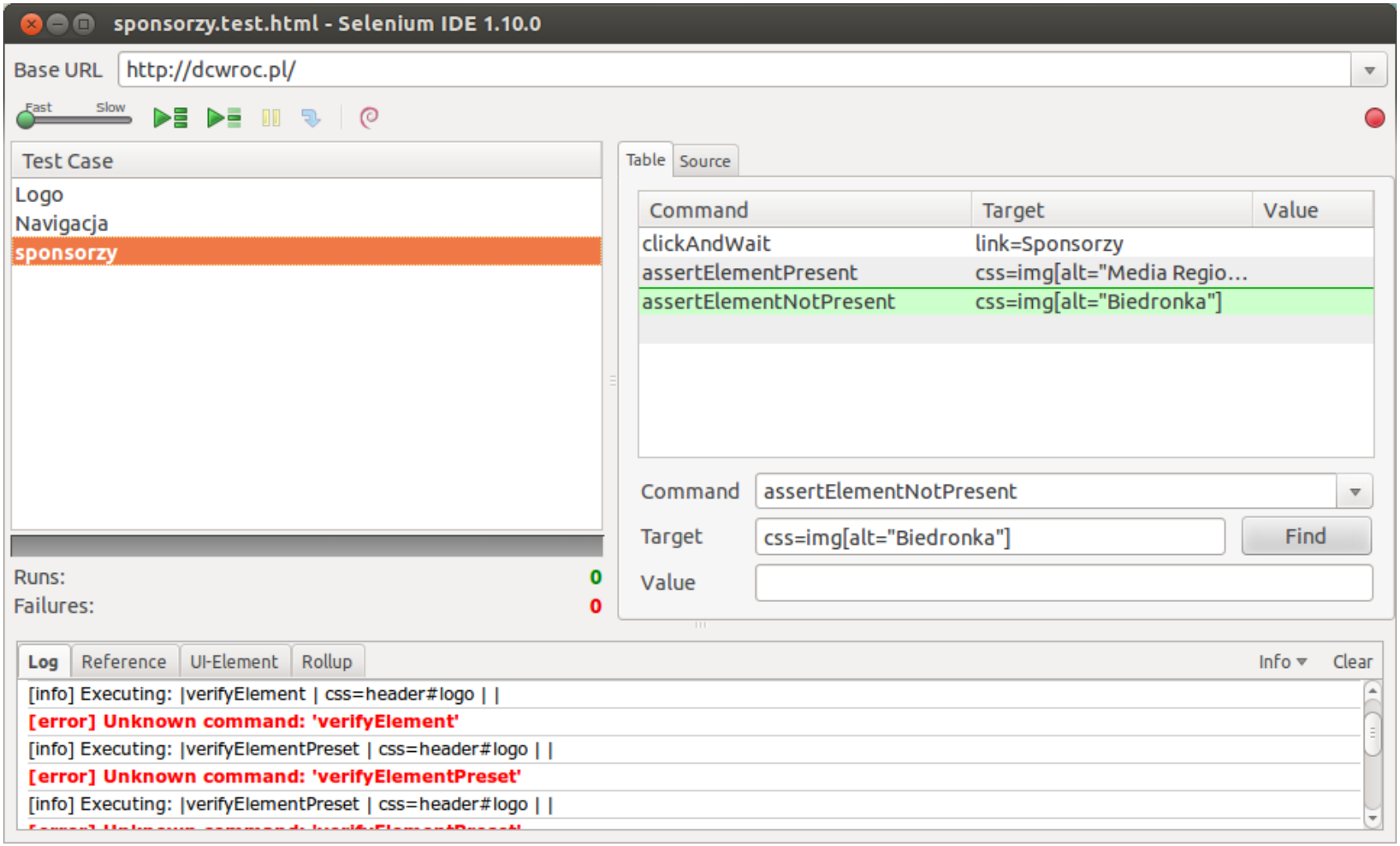
Selenium na ratunek

- Selenium to tak naprawdę zestaw narzędzi.
- Selenium IDE (o tym będe mówił) - to wtyczka do firefoksa, która pozwala przygotowywać testy funkcjonalne a następnie je wykonywać automatycznie.
- Selenium Server - pozwala automatyzować wykonywanie testów selenium
- Selenium WebDriver - pozwala wykonywać testy w różnych frameworkach testujących
- Selenium Grid - Pozwala wykonywać testy funkcjonalne na różnych maszynach i systemach operacyjnych równocześnie.

Selenium IDE

- Wtyczka do firefoksa, do zainstalowania stąd
- Działania użytkownika mogą być nagrywane i zapisane w formie komend, np: kliknij na. . . , wpisz do pola x wartość y, etc.
- Dodatkowo wśród komend mamy polecenia testujące. Polecenia `assert` oraz `verify`

Demo selenium



Słabe strony rozwiązania

- Selenium ma dostęp tylko do tego, do czego ma dostęp javascript.
- Czasami występują problemy, np. okienko drukowania, autoryzacja http.
- W takich wypadków skrypt testujący “wisi” i trudno to zdiagnozować.
- Ponieważ rozwiązanie opiera się na graficznej przeglądarce, automatyzacja rozwiązania może być kłopotliwa (chociaż możliwa).

Automatyzacja - czyli integracja z ANT i Jenkins

- Ten temat był już dzisiaj http://dcwroc.pl/sesje/automatyzacja-w-tworzeniu-aplikacji-opartych-o-drupala-drush-jenkins-ci-phpunit
- Wybrane przez nas narzędzie (ANT, Phing, Maven lub dowolne inne narzędzie) wykonuje wybrane narzędzie testujące.
- Narzędzie wypluwa wyniki do wybranego pliku.
- Narzędzie jenkins, za pomocą wtyczki czyta i interpretuje wyniki.
- Narzędzie testujące - to cokolwiek możecie sobie wyobrazić.
- Przy odpowiedniej konfiguracji Jenkins nasłuchuje na zmiany kodu aplikacji, i wtedy wykonuje odpowiednie testy.

Jak przyspieszyć działanie simpletest

- Testy simpletest tworzą i usuwają **dużo** tabel.
- Jeżeli używamy MyISam, to nie jest to straszne.
- Jeżeli używamy silnika InnoDB, to może naprawde zaboleć.
- Jest też trzecia droga: MyISAM w ramdysku.
- Więcej szczegółów: `http://drupal.org/node/466972`.
- Można użyć profilu vagranta z bazą danych w ramdysku.
- Moje doświadczenia: 2 krotne przyspieszenie działania testów.

Dziękuję za uwagę

jaroslaw[kropa]sobiecki[malpa]mediaregionalne.pl