

RubyOnRails ile Test Temelli Programlama

alper.karapinar@yh.com.tr



YeniHayat
Bilişim Teknolojileri A.Ş.

Sunum İçeriği

- Agile Development
- Test Driven Development
- Rspec
- Cucumber

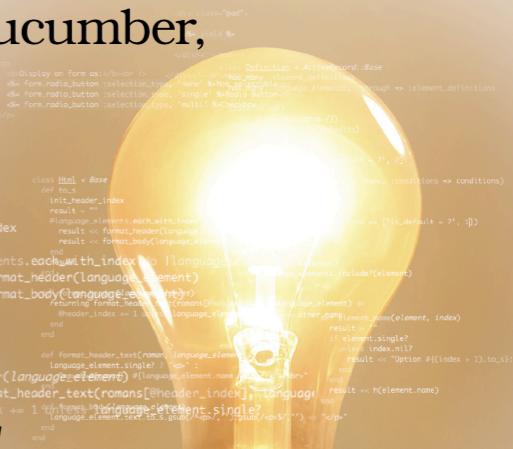
The RSpec Book

David Chelimsky
pragprog.com

Pragmatic
Programmers

The RSpec Book

Behaviour Driven Development
with RSpec, Cucumber,
and Friends



```
class Html < Base
  def to_s
    init_header_index
    result = ''
    @language_elements.each_with_index do |language_element, index|
      result += format_header(language_element)
      result += format_body(language_element)
    end
    result
  end
  def format_header_text(roman, language_element)
    language_element.single? ? "<p>" : "<><b>#{"romans[header_index]}</b> " +
      "#{"language_element.name.upcase}"</p>"
  end
  def format_header(language_element)
    header_index += 1 if !language_element.single?
    returning format_header_text(romans[header_index], language_element)
  end
end
```

David Chelimsky
with Dave Astels,
Zach Dennis,
Aslak Hellesøy,
Bryan Helmkamp,
and Dan North

Edited by Jacquelyn Carter

The Facets of Ruby Series

Agile Development

- Çok hızlı tamamlanan ilk sürüm

- İlk sürümden itibaren “çalışan” uygulama

- Özelliklere (feature) dayalı sürümler
- Kısa aralıklarla, ardışılı sürümler

Her şeyi kapsayan
“mükemmel” ilk dizayn
yerine,

Gelişen ve değişen dizayn

Test kodu yazılmayan kod
kırılgandır

Test Temelli Uygulama Geliştirme



'60 lar,
Mercury Projesi ...



... önce test etmişler.

Üç Temel Aşama

- Red
- Green
- Refactor

Rails ...

Rails ...

... MVC pattern

Rails ...

... MVC pattern

Rspec ...

Üç Konfigürasyon

- Development
- Test
- Production

Test Ortamı

- Test gemleri
- Test konfigürasyonu
- Test DB

RSpec

Rspec ve Unit Testleri

```
class Article < ActiveRecord::Base  
end
```

```
class CreateArticles < ActiveRecord::Migration  
  def self.up  
    create_table :articles do |t|  
      t.string :title  
      t.text :content  
  
      t.timestamps  
    end  
  end  
  
  def self.down  
    drop_table :articles  
  end  
end
```

Rspec ve Unit Testleri

```
class Article < ActiveRecord::Base  
end
```

```
require 'spec_helper'  
  
describe Article do  
  it "is valid with valid attributes"  
  
  it "is not valid without title"  
    it "is not valid without content"  
end
```

Rspec ve Unit Testleri

```
class Article < ActiveRecord::Base  
end
```

Rspec ve Unit Testleri

```
class Article < ActiveRecord::Base  
end
```

Pending:

Article is valid with valid attributes

Not Yet Implemented

./spec/models/article_spec.rb:4

Article is not valid without title

Not Yet Implemented

./spec/models/article_spec.rb:6

Article is not valid without content

Not Yet Implemented

./spec/models/article_spec.rb:8

Rspec ve Unit Testleri

```
require 'spec_helper'

describe Article do
  before(:each) do
    @article = Article.new(:title => "foo", :content => "bar")
  end

  it "is valid with valid attributes" do
    @article.should be_valid
  end

  it "is not valid without title" do
    @article.title = nil
    @article.should_not be_valid
  end

  it "is not valid without content" do
    @article.content = nil
    @article.should_not be_valid
  end
end
```

Rspec ve Unit Testleri

```
Article
```

```
  is valid with valid attributes  
  is not valid without title (FAILED - 1)  
  is not valid without content (FAILED - 2)
```

Rspec ve Unit Testleri

Article

```
is valid with valid attributes
is not valid without title (FAILED - 1)
is not valid without content (FAILED - 2)
```

Failures:

- 1) Article is not valid without title

```
Failure/Error: @article.should_not_be_valid
expected valid? to return false, got true
# ./spec/models/article_spec.rb:14
# /Users/alper/.rvm/gems/ruby-1.8.7-p174/gems/activ
```

- 2) Article is not valid without content

```
Failure/Error: @article.should_not_be_valid
expected valid? to return false, got true
# ./spec/models/article_spec.rb:19
# /Users/alper/.rvm/gems/ruby-1.8.7-p174/gems/activ
```

Rspec ve Unit Testleri

```
class Article < ActiveRecord::Base
  validates :title, :presence => true
  validates :content, :presence => true
end
```

Rspec ve Unit Testleri

```
class Article < ActiveRecord::Base
  validates :title, :presence => true
  validates :content, :presence => true
end
```

```
Article
  is valid with valid attributes
  is not valid without title
  is not valid without content
```

Unit Testlerinde Neler Test Edilmeli?

- Validasyonlar
- Business Logic

published_at

```
it "should have published_at field" do
  @article.should respond_to :published_at
end
```

```
class AddPublishedAtToArticles < ActiveRecord::Migration
  def self.up
    add_column :articles, :published_at, :datetime
  end

  def self.down
    remove_column :articles, :published_at
  end
end
```

published?

```
it "should have published? method" do
  @article.should respond_to :published?
end
```

```
class Article < ActiveRecord::Base
  validates :title, :presence => true
  validates :content, :presence => true

  def published?
  end
end
```

#published? specs

```
describe "#published?" do
  it "should be true when published_at is in past"
  it "should be false when published_at is in future"
  it "should be false when if published_at is not set"
end
```

#published? specs

```
describe "#published?" do
  it "should be true when published_at is in past" do
    @article.published_at = 2.days.ago
    @article.published?.should be_true
  end

  it "should be false when published_at is in future" do
    @article.published_at = 2.days.from_now
    @article.published?.should be_false
  end

  it "should be false when if published_at is not set" do
    @article.published_at = nil
    @article.published?.should be_false
  end
end
```

Red ...

```
bundle exec /Users/alper/.rvm/gems/ruby-1.8.7-p174/gems/rspec-core-2.0.0.rc1/bin/rspec -c
.....F...
```

Failures:

- 1) Article#published? should be true when published_at is in past
Failure/Error: @article.published?.should be_true
expected nil to be true
./spec/models/article_spec.rb:33
/Users/alper/.rvm/gems/ruby-1.8.7-p174/gems/activesupport-3.0.0/lib/active_support/core_ext/object/delegation.rb:11:in `method_missing'

Finished in 0.03771 seconds
8 examples, 1 failure

```
class Article < ActiveRecord::Base
  validates :title, :presence => true
  validates :content, :presence => true

  def published?
    !published_at.nil? and published_at.past?
  end
end
```

Green ..

```
bundle exec /Users/alper/.rvm/gems/ruby-1.8.7
.....
Finished in 0.03604 seconds
8 examples, 0 failures
```

Green ..

```
bundle exec /Users/alper/.rvm/gems/ruby-1.8.7
.....
Finished in 0.03604 seconds
8 examples, 0 failures
```

Article

```
is valid with valid attributes
is not valid without title
is not valid without content
should have published_at field
should have published? method
#published?
  should be true when published_at is in past
  should be false when published_at is in future
  should be false when if published_at is not set
```

Fonksiyonel Testlerde Neler Test Edilmeli?

- Request başarılı mı?
- Kullanıcı doğru sayfaya yönlendirildi mi?
- Kullanıcı başarılı şekilde giriş yaptı mı?
- Gerekli değişkenler doğru şekilde atandı mı?
- Kullanıcıya doğru mesaj iletildi mi?

Cucumber



Behaviours Driven Development with elegance and joy

1: Describe behaviour in plain text

```
Feature: Addition
  In order to avoid silly mistakes
  As a math idiot
  I want to be told the sum of two numbers
```

```
Scenario: Add two numbers
  Given I have entered 50 into the calculator
  And I have entered 70 into the calculator
  When I press add
  Then the result should be 120 on the screen
```

4: Write code to make the step pass

```
class Calculator
  def push(n)
    @stack[-1] = n
  end
end
```

Cucumber

```
1 cucumber/features/addition.feature
Feature: Addition
  In order to avoid silly mistakes
  As a math idiot
  I want to be told the sum of two numbers
  Scenario: Add two numbers
    Given I have entered 50 into the calculator
    And I have entered 70 into the calculator
    When I press add
    Then the result should be 120 on the screen
```

7: Repeat 1-6 until the money runs out

Cucumber lets software development teams describe how software should behave in plain text. The text is written in a common language that can be understood by both humans and serves as documentation, automated tests and development aid – all rolled into one.

We want swag!



The money raised for this campaign will be spent to produce Cucumber swag to promote Cucumber: T-shirts, cups and other things.

2: Write a step definition in Ruby

```
Given I have entered 50 into the calculator
And I have entered 70 into the calculator
When I press add
Then the result should be 120 on the screen
```

3: Run and watch it fail

```
1 cucumber/features/addition.feature
Feature: Addition
  In order to avoid silly mistakes
  As a math idiot
  I want to be told the sum of two numbers
  Scenario: Add two numbers
    Given I have entered 50 into the calculator
    And I have entered 70 into the calculator
    When I press add
    Then the result should be 120 on the screen
```

6: Repeat 2-5 until green like a cucumber

```
1 cucumber/features/addition.feature
Feature: Addition
  In order to avoid silly mistakes
  As a math idiot
  I want to be told the sum of two numbers
  Scenario: Add two numbers
    Given I have entered 50 into the calculator
    And I have entered 70 into the calculator
    When I press add
    Then the result should be 120 on the screen
```

Download

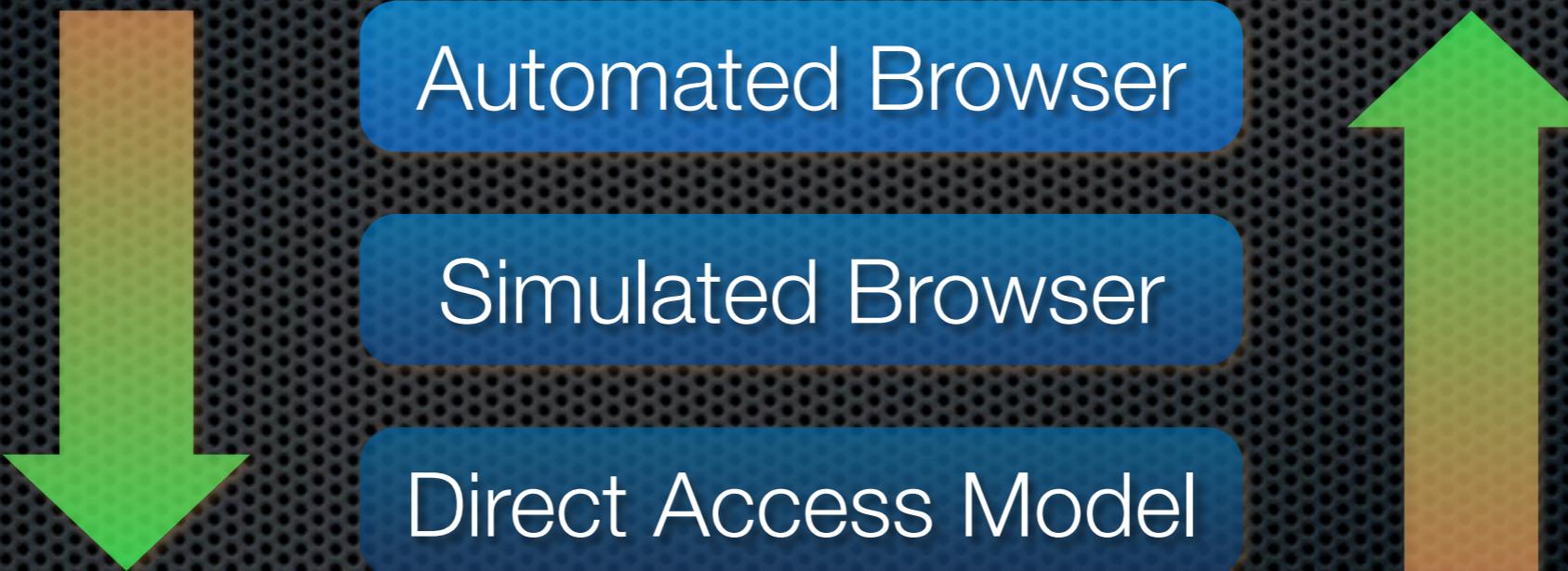
You need Ruby installed. Then just run `gem install cucumber` from a command prompt. Now, run `cucumber --help`.

The [wiki](#) has more information.

Entegrasyon Testleri

Yavaş

Entegre



Hızlı

İzole

Ana sayfa:

Normal bir kullanıcı anasayfada yalnızca
yayınlanmış makaleleri görebilmeli

Mesela,

- * “İstanbul” isimli yayınlanmış bir makalem ve
- * “Ankara” isimli yayınlanmamış bir makalem olsun,
- * kullanıcı anasayfaya geldiğinde
- * “İstanbul” isimli makaleyi görebilmeli,
- * “Ankara” isimli makaleyi görememeli

Cucumber Features

Feature: Home page contents

As a visitor

I should only see published articles in homepage

Scenario: Show only published articles

Given I have a published article titled "Recent Article"

And I have an unpublished article titled "Planned Article"

When I go to the home page

Then I should see "Recent Article"

And I should not see "Planned Article"

Steps

```
Given /^I have a published article titled "([^"]*)"$/ do |title|
  Article.create!(
    :title => title,
    :content => "some content",
    :published_at => 2.days.ago)
end

Given /^I have an unpublished article titled "([^"]*)"$/ do |title|
  Article.create!(
    :title => title,
    :content => "some content",
    :published_at => 2.days.from_now)
end
```

HTML Output

Cucumber Features

Feature: Home page contents

As a visitor
I should only see published articles in homepage

Scenario: Show only published articles

Given I have a published article titled "Recent Article"

And I have an unpublished article titled "Planned Article"

When I go to the home page

Then I should see "Recent Article"

And I should not see "Planned Article"

i18n

```
# language: tr
```

Özellik: Toplama

Gülünç hatalardan sakınmak için

Matematikten anlamayan bir ahmak olarak

Hesap makinasının bana iki sayının toplamını bulmasını istiyorum

Senaryo taslağı: İki sayıyı topla

Diyelim ki hesap makinesine <girdi_1> girdim

Ve hesap makinesine <girdi_2> girdim

Eğer ki <düğme> tuşuna basarsam

O zaman ekrandaki sonuç <çıktı> olmalı

Örnekler:

girdi_1	girdi_2	düğme	çıktı
20	30	topla	50
2	5	topla	7
0	40	topla	40

Test Ortamında Veri Yönetimi

- Test Database
- Fixtures
- Mocks

Fixtures

```
# products.yml

kaneppe:
    name: Kanepe
    price: 399
    manufacturer: minimalist
    categories: mobilya

tv_sehpasi:
    name: TV Sehpası
    price: 149
    manufacturer: futurist
    categories: mobilya, elektronik
```

```
# manufacturers.yml

minimalist:
    name: MinimalistMobilya

futurist:
    name: FuturistElektronik
```

```
# categories.yml

mobilya:
    name: Mobilya

elektronik:
    name: Elektronik
```

Fixture Replacements

- Machinist
- FactoryGirl



Factory Girl

Factory Girl provides a framework and DSL to define and use factories which create data in test suites. It is less error-prone, more explicit, and easier to work with than fixtures.

[Source ➔](#)

FactoryGirl

Lazy attributes

```
factory :user do
  # ...
  activation_code { User.generate_activation_code }
end
```

FactoryGirl

Dependent attributes

```
factory :user do
  first_name 'Ahmet'
  last_name  'Sarı'
  email { "#{first_name}.#{last_name}@example.com".downcase }
end
```

```
Factory(:user, :last_name => 'Kara').email
# => "ahmet.kara@example.com"
```

FactoryGirl

Inheritance

```
factory :article do
  title 'title'
  content 'content'
end

factory :removed_article, :parent => :post do
  removed true
  association :approver, :factory => :user
end
```

Diğer Araçlar

- Autotest
- Rcov
- Watir

Learning Ruby

WITH THE EDGE CASE RUBY KOANS

The Koans walk you along the path to enlightenment in order to learn Ruby. The goal is to learn the Ruby language, syntax, structure, and some common functions and libraries. We also instill you culture. Testing is not just something we pay lip service to, but something we live. It's essential to your quest to learn and conquer things in the language.

rubykoans.com

[DOWNLOAD THE KOANS](#)

NIX

WINDOWS

1. INTRODUCTION

2. THE STRUCTURE

3. INSTALLING RUBY

4. WHAT EDITORS CAN USE

5. THE PATH TO ENLIGHTENMENT

6. AUTHORS

7. INSPIRATION

8. OTHER RESOURCES



The Structure

about_array_assignment.rb
about_arrays.rb
about_asserts.rb
about_blocks.rb
about_class_methods.rb
about_classes.rb
about_constants.rb
about_control_statements.rb
about_date_time.rb

The koans are broken out into areas by file, hashes are covered in about_hashes.rb, modules are introduced in about_modules.rb, etc. They are presented in order in the path_to_enlightenment.rb file.