

Who am I?

- Brian Ford
- Language geek, Ruby enthusiast, Rails developer
- *caveat emptor*: I am not a TDD or BDD expert

Goals for this talk

- Briefly introduce **BDD**
- Highlight some uses of BDD
- Suggest some answers to the question:

“Why should I switch to BDD?”

Assumptions

- You know Ruby
- You've read about or tried TDD

OR

- You're a designer, manager, or client

Test-driven development

“This isn’t a book about testing.”

— David Astels

*test-driven development:
A Practical Guide*

Test-driven development

- TDD is not synonymous with “testing”
- Red, Green, Refactor
- The importance of refactoring

“You aren’t going to get it right the first time because you don’t even know what ‘right’ is.”

Is there room for TDD and BDD?

- Tests written first are better than specs written after
- The available tools make a big difference
- The suitability of the language is important

Behavior-driven development

How is it similar to TDD?

- Red, Green, Refactor
- The importance of refactoring

Behavior-driven development

How is it different than TDD?

- Focus on behavior
- Danger of mixing implementation with interface
- DSL — lower barrier to entry
- Aesthetics

Behavior-driven development

Focus on **behavior**

- Behavior is a continuum
- Behavior depends on perspective
- Behavior is “objective”

Behavior-driven development

Aesthetics are important

```
class PagesControllerTest < Test::Unit::TestCase
  def test_restore_should_make_page_active_again
    login_as :aaron
    post :restore, :id => 1
    assert assigns(:page)
    assert_redirected_to all_pages_path
  end
end
```

Behavior-driven development

Aesthetics are important

```
context "An admin user" do
  setup do
    login_as :aaron
  end

  specify "should be able to restore an archived page" do
    controller.should_redirect_to all_pages_path
    post :restore, :id => 1
    assigns[:page].should_not_be_nil
  end
end
```

BDD meets IxD

- Design is essential for quality, usable software
- Most designers are not programmers
- Most programmers do not design well
- We need artifacts to communicate
- All forms of communication are not equal

Spec'ing Rails Views

```
context "/person/list" do

  setup do
    @smith = mock_model(Person)
    @jones = mock_model(Person)
    @smith.stub!(:name).and_return("Joe")
    @jones.stub!(:name).and_return("Joe")
    assigns[:people] = [@smith, @jones]
  end

  specify "should have a <div> tag with :id => 'a'" do
    render "/person/list"
    response.should_have_tag 'div', :attributes =>{:id => "a"}
  end

  specify "should have a <hr> tag with :id => 'spacer'" do
    render "/person/list"
    response.should_have_tag 'hr', :attributes => {:id => "spacer"}
  end
end
```

From: <http://rspec.rubyforge.org/documentation/rails/writing/views.html>

Resources and references

- *test-driven development: A Practical Guide* [Astels 2003]
- *Refactoring* [Fowler 2003]
- <http://www.testdriven.com>
- <http://www.behavior-driven.org>
- <http://www.dialogue-driven.org>
- <http://rspec.rubyforge.org>
- http://blog.daveastels.com/files/BDD_Intro.pdf