Regression Therapy – Contentful Testing

Anthony Bailey

http://anthonybailey.net

3 April 2008, Scotland on Rails

Teaser

The conventional wisdom:

```
# Assert explicitly and exactly what we care about.
assert_select("divs#of_interest", /my_desires/)
```

My dumb straw man alternative:

```
# Everything should just be exactly like yesterday.
assert_equals(File.new("expected.html").read,
   @response.body)
```

Setting the scene

- I won’t try to behave.
- When I say view content, I mean view content.
- When I don’t, I mean everything…
- …even other languages and frameworks.

Setting the scene: terminology

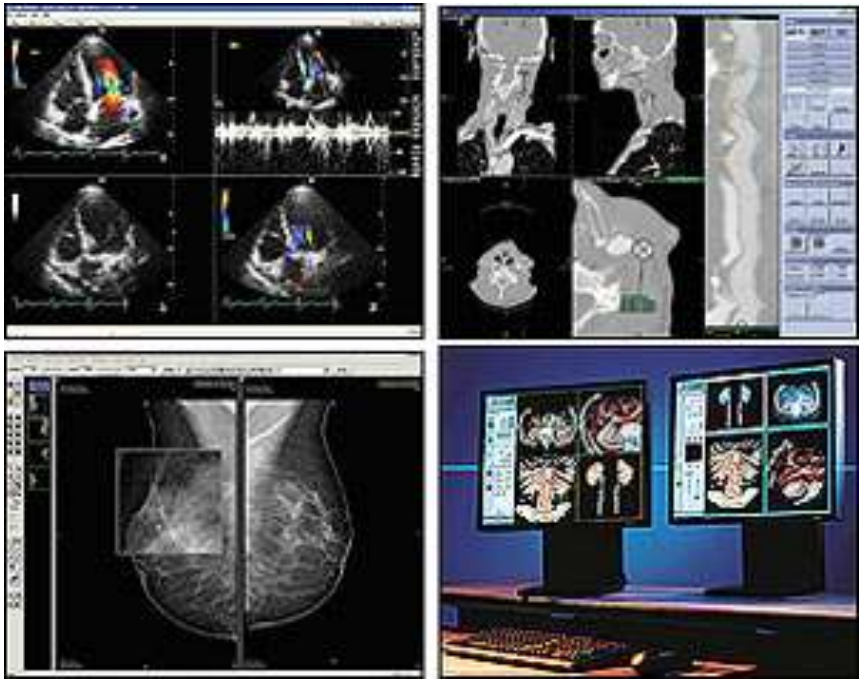
- I won’t try to behave

Setting the scene: scope

- When I say view content, I mean view content.
  - not the models the views display
  - not the presentation logic they use to display it
  - not the controller logic invoked by interacting with the view
  - not the workflow that the views comprise
  - just the tags
  - (at least for now)

Setting the scene: scope

- When I don't say views, I mean everything.
  - Contentful testing can work in other domains

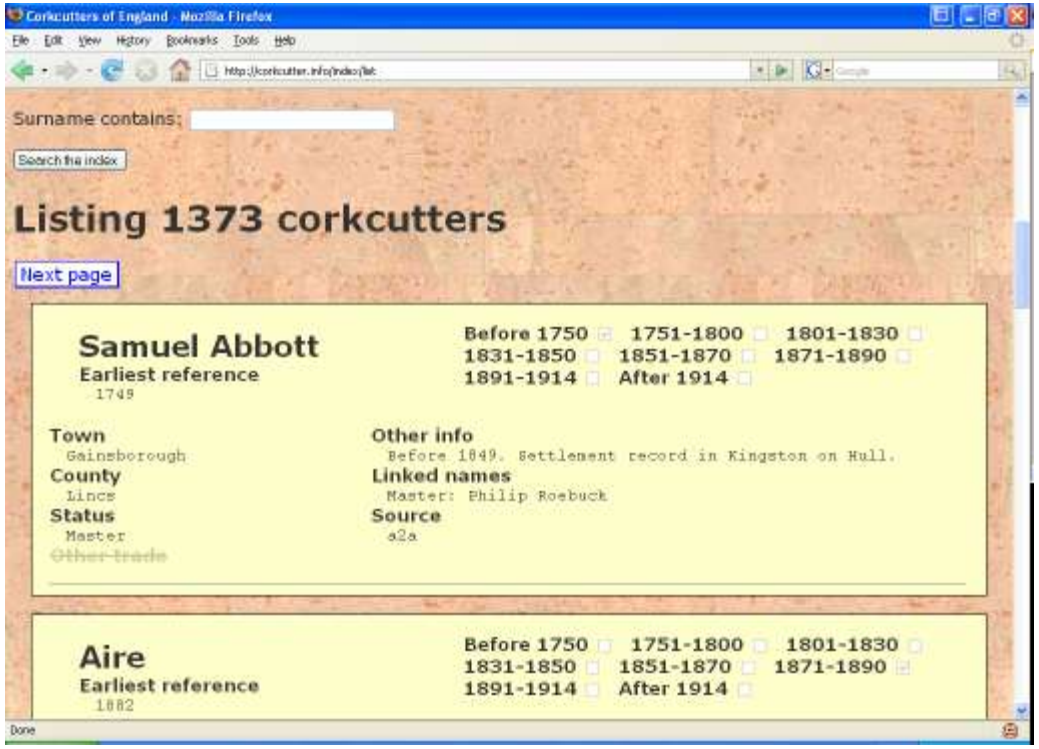


Setting the scene: scope

- Other languages and frameworks
  - Contentful testing doesn't need Ruby or Rails,
    - used it in a C++ desktop app, a Java/Spring web app
  - But this talk will focus on a Rails plug-in:
    - http://contentful.rubyforge.org/

Setting the scene: personal

- The plug-in was born of genuine need
  - but in a small, simple, CRUDdy app.



Why do we test?

- To maintain known good behavior and catch accidental bad changes

  .....!..!.....!.....

- To drive intended changes

  >>

- To explore and understand emergent changes

  .....!!??????.....!!??????.....!!??????.....

How do Rails devs test the view?

- Functional test, plus assert_select

```
class StoreControllerTest < Test::Unit::TestCase
  # ...
  def test_index
    get :index
    assert_select "div.entry span.price", /23.45/
  end
end
```

How do Rails devs test the view?

- Decoupled unit test, plus assert_select

```
class StoreViewTest < Test::Rails::ViewTestCase
  fixtures :products

  def test_index
    assigns[:products] = [products(:rails_book)]
    render :index
    assert_select "div.entry span.price", /23.45/
  end
end
```

But views are not very assertive

- Assertion tests
  - don't help exploration
  - are great for test-driving,
    - but view content doesn't get much value from this.
  - have selective coverage
  - are verbose when coverage needs to be high;
    - translation takes time and space
  - are expensive to update when things change

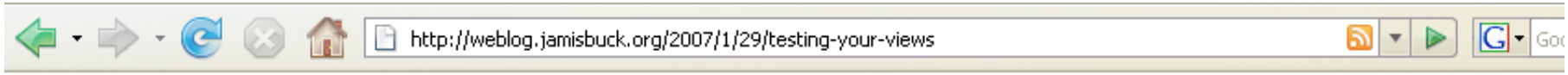
How much do Rails devs test the view?

- Not much!

- *"I find that the View tests that RSpec and Zentest provide are fragile and provide dubious value. Rarely do they find bugs and often they break for unimportant reasons. I just don't find them to have a good return on investment.“*
  – Jay Fields, Thoughtworks.

- err.the_blog’s groupthink thought similar

assert_select



Testing your views

Posted by Jamis on January 29, 2007 @ 01:42 PM

Here's a quick little "advice" tip: don't use assertions to test the structure of your views.

That is to say, don't do this:

```
1 #make sure the text field is in the table cell
2 assert_select "table td input[type=text]"
3
4 # make sure the person's name is in the h2 header tag
5 assert_select "h2", person.name
```

Why not? Because you want your views to be very fluid. You want to keep the cost of change is so low that you have no hesitation to jump in and rearrange things to make the view cleaner. If you are using explicit tag names in your tests, you reduce that fluidity. Your views become rigid, because your tests imply that using anything but a table to format your form is wrong. Want to use an h1 for the person's name, or a div? Don't you dare, *it'll break the tests*.

The better way to test your views is to think about what you are really wanting to test. First of all, don't test static content, like table structures and the order of form fields on a page. Instead, test the *dynamic* parts of your view, especially those parts that are subject to conditional rendering. Secondly, test *semantically*, not *syntactically*. That is to say, don't base a test on the *type* of the tag, but rather on what you want the content to represent. Use CSS classes and DOM id's instead of explicit tag names.

Past life regression

- Regression tests make some things a bit better
  - Any one test is simple to express
    - “the expected output is exactly as in this file”
  - Expected output needs no translation
    - - it’s just a straight copy
- But they make other things much worse
  - Very expensive to maintain, because
    - each test is brittle, breaking on minor changes
    - a single production code change can break many tests

Regression therapy

- One step back, two steps forward
- Dampen the noise
  - diff the DOM, not the text - normalize
  - DRY up tested content
- Smooth the workflow
  - easily create tests
  - conveniently inspect changes
  - quickly accept changes
  - …and these should all work well in batch

Contentful plug-in

```
% ruby script/plugin install
  svn://rubyforge.org/var/svn/contentful
```

```ruby
class StoreControllerTest < Test::Unit::TestCase
  # ...
  def test_index
    get :index
    assert_select "div.entry span.price", /23.45/
    assert_contentful
  end
end
```

```
C:\depot>ruby \depot\test\functional\store_controller_test.rb
Loaded suite /depot/test/functional/store_controller_test
Started
!
Finished in 0.23 seconds.

  1) Contentful Notification:
test_index(StoreControllerTest):
Generated C:/depot/config/../test/contentful/store_controller/in
dex/expected.html

1 tests, 1 assertions, 0 failures, 0 errors
```

```
<div class="entry">
  <img src="<%= product.image_url %>" />
  <h3><%= h(product.title) %></h4>
  <%= product.description %>
  <span class="price"><%= number_to_currency(produ
  <% form_remote_tag :url => { :action => :add_to_c
    <%= submit_tag "Add to Cart" %>
  <%
</di
<% e
```

```
91:in `parsed_response_body'
     C:/depot/vendor/plugins/contentful/lib/assertion_helper.rb:
21:in `assert_contentful'
     /depot/test/functional/store_controller_test.rb:19:in `test
_index'
     C:/depot/vendor/plugins/contentful/lib/patch_test_case_run.
rb:11:in `run']:
Invalid mark-up in content:
ignoring attempt to close h3 with h4
  opened at byte 1842, line 48
  closed at byte 1878, line 48
  attributes at open: {}
  text around open: "=\"rails.png\" />\n    <h3>Agile Web Develo
"
  text around close: "velopment With Rails</h4>\n    Dummy desc"


1 tests, 1 assertions, 1 failures, 0 errors
```

```
<div class="entry">
  <img src="<%= product.image_url %>" />
  <h3><%= h(product.title) %></h3>
  Accidental change
  <%= product.description %>
  <span class="price"><%= number_to_currency(produc
  <% form_remote_tag :url => { :action => :add_to_ca
    <%= submit_tag "Add to Cart" %>
  <%
</di
<% er
```

```
Finished in 0.271 seconds.

  1) Failure:
test_index(StoreControllerTest)
    [C:/depot/vendor/plugins/contentful/lib/assertion_helper.rb:
40:in `assert_contentful'
     /depot/test/functional/store_controller_test.rb:19:in `test
_index'
     C:/depot/vendor/plugins/contentful/lib/patch_test_case_run.
rb:11:in `run']:
diff test/contentful/store_controller/index/*.to_diff
to see the content change
.

1 tests, 2 assertions, 1 failures, 0 errors
```

```
  1) Failure:
test_index(StoreControllerTest)
    [C:/depot/vendor/plugins/contentful/lib/assertion_helper.rb:
40:in `assert_contentful'
     /depot/test/functional/store_controller_test.rb:19:in `test
_index'
     C:/depot/vendor/plugins/contentful/lib/patch_test_case_run.
rb:11:in `run']:
diff test/contentful/store_controller/index/*.to_diff
to see the content change
.

1 tests, 2 assertions, 1 failures, 0 errors

C:\depot>ls test/contentful/store_controller/index
changed.html  changed.to_diff  expected.html  expected.to_diff

C:\depot>diff test/contentful/store_controller/index/*.to_diff
213d212
<     Accidental change
249d247
<     Accidental change

C:\depot>_
```

```
<div class="entry">
  <img src="<%= product.image_url %>" />
  <h3><%= h(product.title) %></h3>
  <%= product.description %>
  <span class="price"><%= number_to_currency(produ
  <% form_remote_tag :url => { :action => :add_to_c
    <%= submit_tag "Add to Cart" %>
  <%
</di
<% e
```

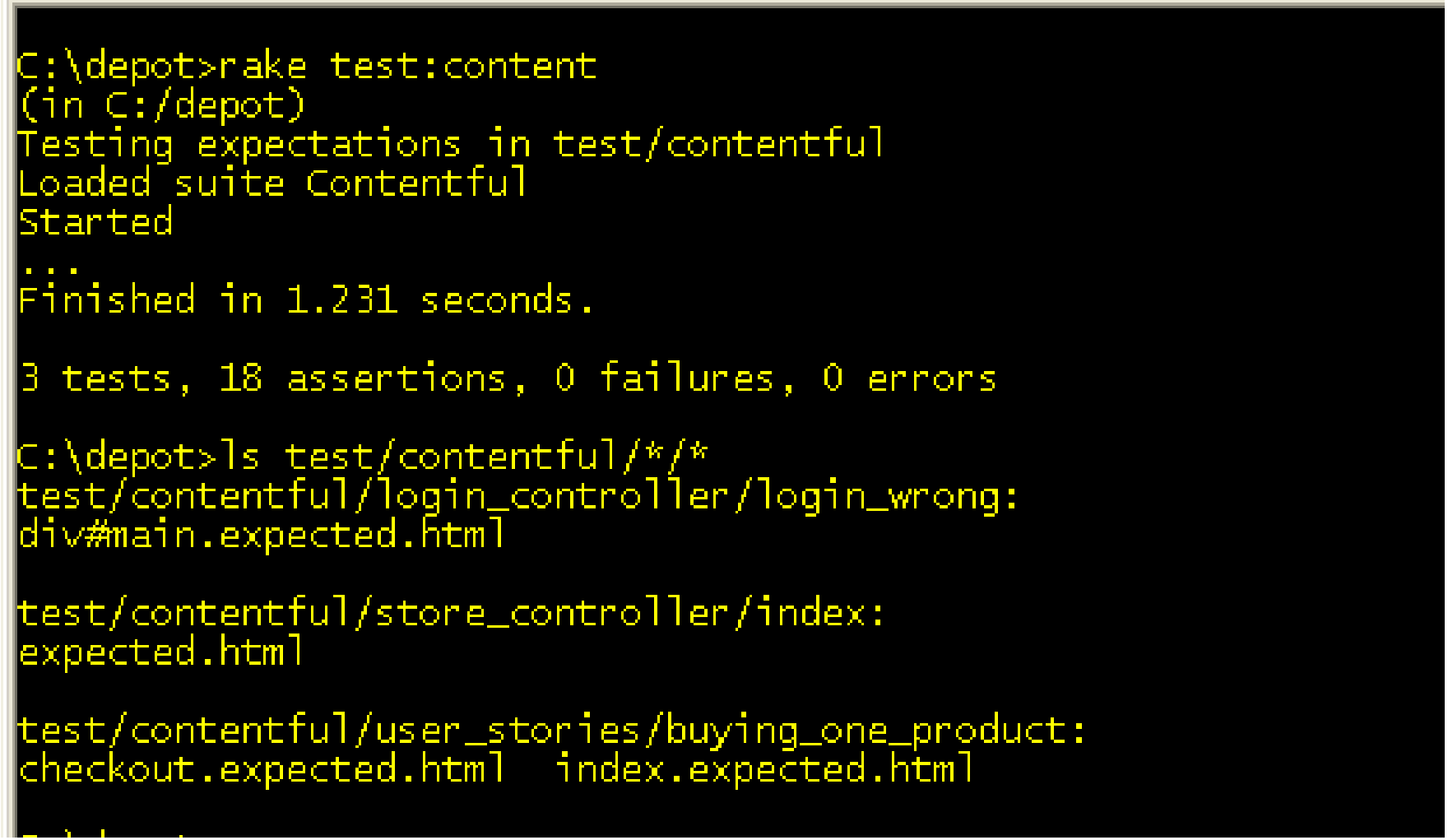
```
C:\depot>ruby \depot\test\functional\store_controller_test.rb
Loaded suite /depot/test/functional/store_controller_test
Started
.
Finished in 0.231 seconds.

1 tests, 1 assertions, 0 failures, 0 errors

C:\depot>ls test/contentful/store_controller/index
expected.html

C:\depot>
```

```
def test_buying_one_product
  LineItem.delete_all
  Order.delete_all
  ruby_book = products(:ruby_book)

  get '/store/index'
  assert_response :success
  assert_template 'index'
  assert session[:cart].items.empty?
  assert_contentful :index

  xml_http_request '/store/add_to_cart', :id => ruby_book
  assert_response :success

  post '/store/checkout'
  assert_response :success
  assert_contentful :checkout
  assert_template 'checkout'

  order_fields = {
    :name => 'Dave Thomas', :address => '123 The Street
```

```
C:\depot>ruby test/integration/user_stories_test.rb
Loaded suite test/integration/user_stories_test
Started
!!
Finished in 0.601 seconds.

  1) Contentful Notification:
test_buying_one_product(UserStoriesTest):
Generated C:/depot/config/../test/contentful/user_stories/buying
_one_product/index.expected.html

  2) Contentful Notification:
test_buying_one_product(UserStoriesTest):
Generated C:/depot/config/../test/contentful/user_stories/buying
_one_product/checkout.expected.html

1 tests, 14 assertions, 0 failures, 0 errors

C:\depot>ls test/contentful/user_stories/buying_one_product
checkout.expected.html  index.expected.html

C:\depot>_
```

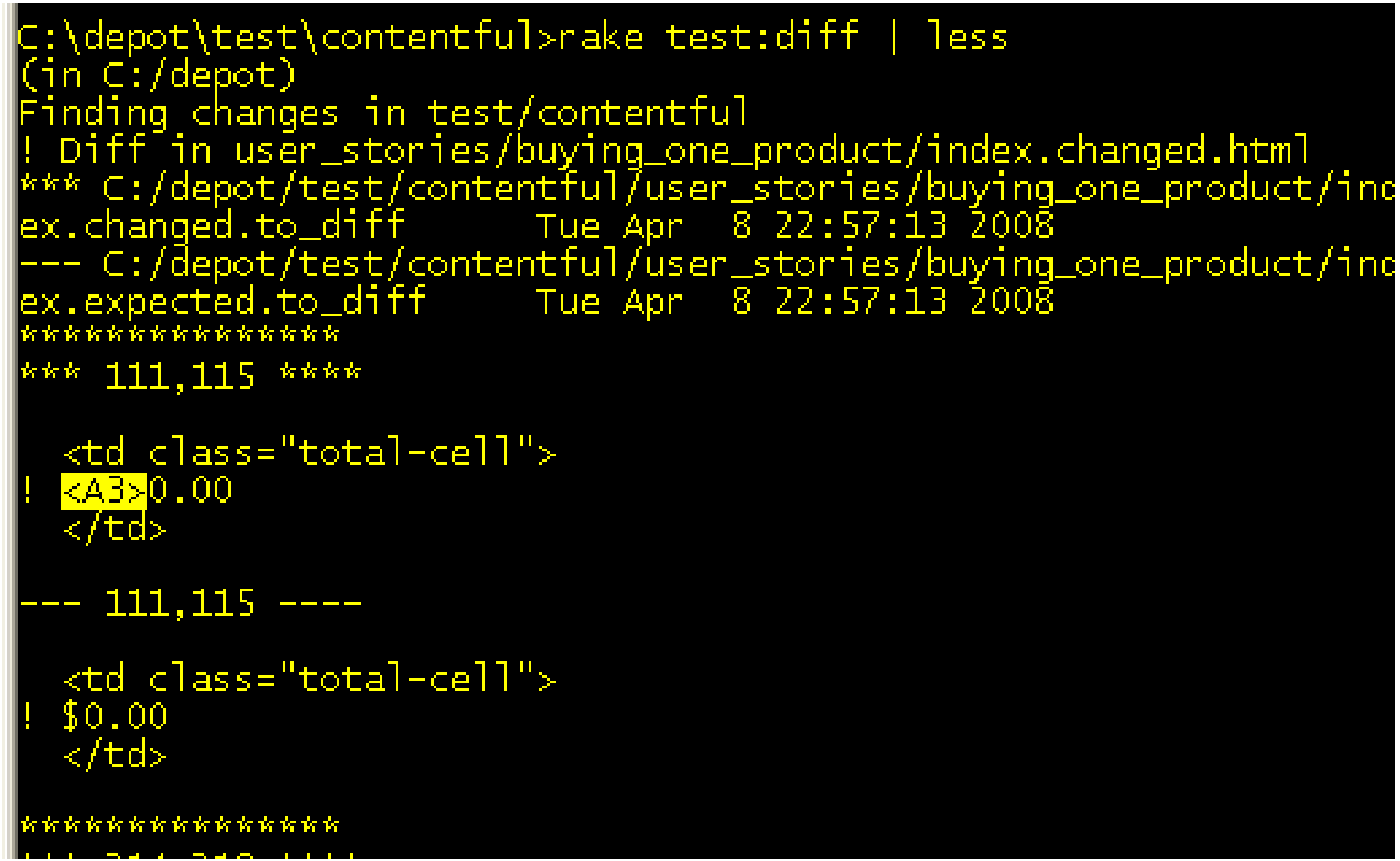
```
  def test_login_wrong
    post :login, :name => users(:dave).name, :password => 'w
    assert_template "login"
    select_contentful "div#main"
    as
    as
  enc

  def
    pc
    as
    as
  enc
end
```

```
C:\depot>ruby test/functional/login_controller_test.rb
Loaded suite test/functional/login_controller_test
Started
...!
Finished in 0.19 seconds.

  1) Contentful Notification:
test_login_wrong(LoginControllerTest):
Generated C:/depot/config/../test/contentful/login_controller/lo
gin_wrong/div#main.expected.html

4 tests, 10 assertions, 0 failures, 0 errors

C:\depot>cat test/contentful/login_controller/login_wrong/*
<div id="main">
          <div id="notice">Invalid user/password combination</di
v>
          <div class="depot-form">
  <fieldset>
    <legend>Please Log In</legend>

    <form action="/login/login" method="post">
      <p>
```

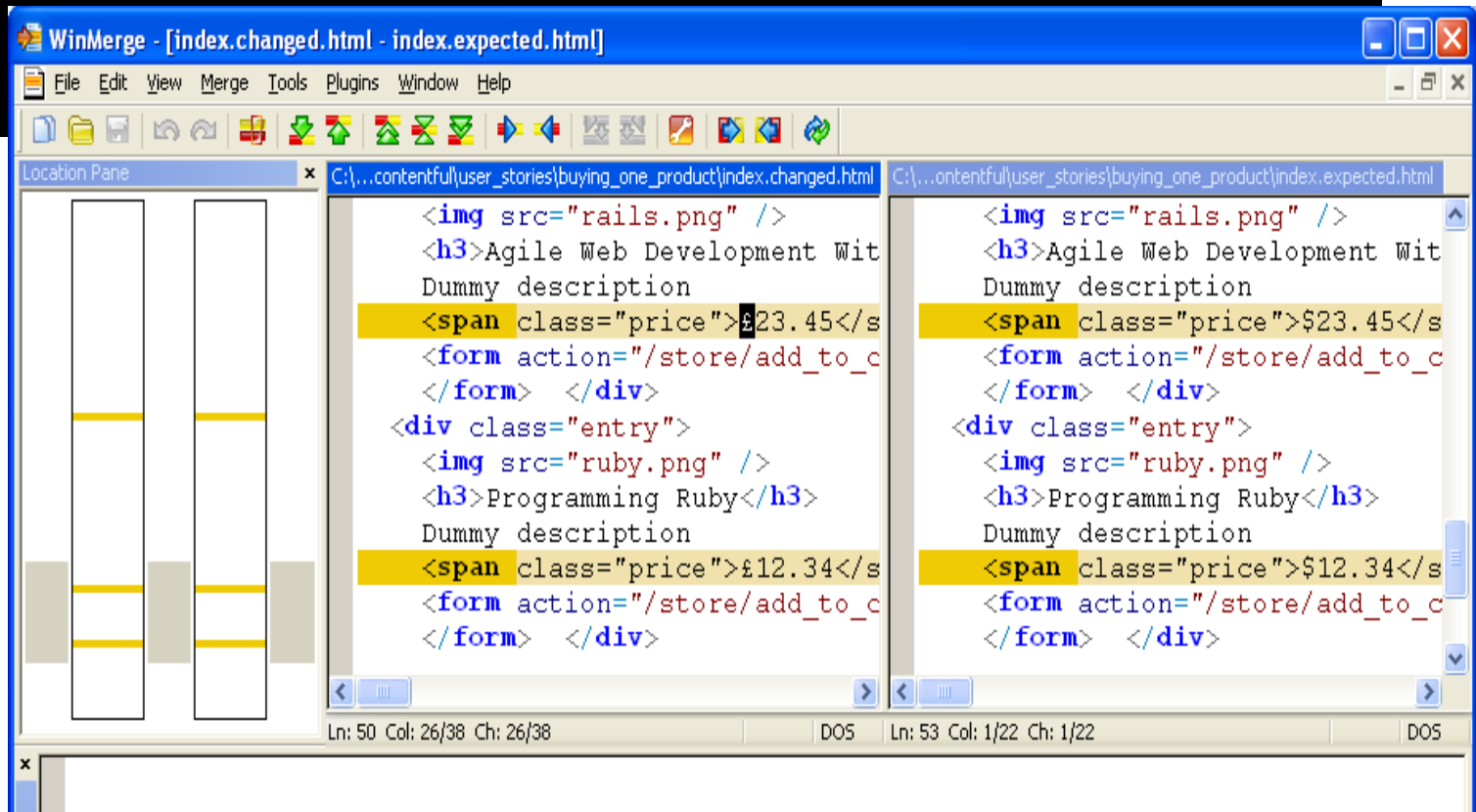

C:\depot>rake test:content
(in C:/depot)
Testing expectations in test/contentful
Loaded suite Contentful
Started
...
Finished in 1.231 seconds.
3 tests, 18 assertions, 0 failures, 0 errors
C:\depot>ls test/contentful/*/*
test/contentful/login_controller/login_wrong:
div#main.expected.html
test/contentful/store_controller/index:
expected.html
test/contentful/user_stories/buying_one_product:
checkout.expected.html index.expected.html

```ruby
module ApplicationHelper

  def number_to_currency(price)
#    sprintf("$%0.02f", price)
     sprintf("£%0.02f", price)
  end

end
```

```
C:\depot\test\contentful>rake test:content
(in C:/depot)
Testing expectations in test/contentful
Loaded suite Contentful
Started
.FF
Finished in 0.701 seconds.

  1) Failure:
test_index(StoreControllerTest)
    [C:/depot/vendor/plugins/contentful/lib/assertion_helper.rb:
40:in `assert_contentful'
     C:/depot/config/../test/contentful/../functional/store_cont
roller_test.rb:19:in `test_index'
     C:/depot/vendor/plugins/contentful/lib/patch_test_case_run.
rb:11:in `run']:
diff test/contentful/store_controller/index/*.to_diff
to see the content change
.

  2) Failure:
test_buying_one_product(UserStoriesTest)
    [C:/depot/vendor/plugins/contentful/lib/assertion_helper.rb:
40:in `assert_contentful'
     C:/depot/config/../test/contentful/../integration/user_stor
ies_test.rb:15:in `test_buying_one_product'
     C:/depot/vendor/plugins/contentful/lib/patch_test_case_run.
rb:11:in `run'
     c:/ruby/lib/ruby/gems/1.8/gems/actionpack-1.13.3/lib/action
_controller/integration.rb:453:in `run']:
diff test/contentful/user_stories/buying_one_product/index.*.to_
diff
to see the content change
.

3 tests, 9 assertions, 2 failures, 0 errors

C:\depot\test\contentful>ls */*/*changed.html
store_controller/index/changed.html
user_stories/buying_one_product/index.changed.html
```


C:\depot\test\contentful>rake test:diff | less
(in C:/depot)
Finding changes in test/contentful
! Diff in user_stories/buying_one_product/index.changed.html
*** C:/depot/test/contentful/user_stories/buying_one_product/inc
ex.changed.to_diff Tue Apr 8 22:57:13 2008
--- C:/depot/test/contentful/user_stories/buying_one_product/inc
ex.expected.to_diff Tue Apr 8 22:57:13 2008
****
*** 111,115 ****
<td class="total-cell">
! <A3>0.00
</td>
--- 111,115 ----
<td class="total-cell">
! $0.00
</td>
****

```
C:\depot\test\contentful>rake test:diff DIFF=diff | grep "[<>]"
< £0.00
> $0.00
< £23.45
> $23.45
< £12.34
> $12.34
< £0.00
> $0.00
< £23.45
> $23.45
< £12.34
> $12.34
```



```
C:\depot\test\contentful>rake test:accept
(in C:/depot)
Finding changes in test/contentful
! Accepting user_stories/buying_one_product/index.changed.html
! Accepting store_controller/index/changed.html
```

```
3 tests, 9 assertions, 2 failures, 0 errors

C:\depot\test\contentful>cd store_controller

C:\depot\test\contentful\store_controller>rake test:accept
(in C:/depot)
Finding changes in test/contentful/store_controller
! Accepting store_controller/index/changed.html

C:\depot\test\contentful\store_controller>cd ..

C:\depot\test\contentful>rm -rf user_stories

C:\depot\test\contentful>ruby ../integ*/user_stories_test.rb
Loaded suite ../integration/user_stories_test
Started
!!
Finished in 0.39 seconds.

  1) Contentful Notification:
test_buying_one_product(UserStoriesTest):
```

Other paths to contentfulness

- That was just one implementation
  - Really, I want to sell the contentful testing *pattern* more than my particular Rails *plug-in*
- Capture a normalized from of entire output
- and, automate new/test/review/accept *somehow*
    - others Rails devs might prefer different Rake tasks
    - other devs might want something other than Rake
    - maybe use some IDE other than the command-line

Pushing contentful testing

- Leveraging the coverage/upkeep trade-off
  - I pay the tax on dense coverage to get value from serendipitous discovery of changes
- Testing more than the view content
  - cover for exploration, spiking, and follow-on
- Bonus value in regular assertion tests
  - If you're assertion testing controllers anyway, an extra assert_contentful is almost free
  - I (guiltily) trade decoupling for convenience

An obvious contentful usecase

- Use the machinery as a temporary testing vise

  (term due to Michael Feathers)

  – Pin down all view content during a big refactor
    - extracting partials and helpers
    - refactoring form builders
    - changing template system

- `CONTENTFUL_AUTO = true`

  – `assert_contentful` in every functional test

http://contentful.rubyforge.org