**Yet another frickin’ blog app**

gem install rails -v 3.0

rails new bloog --skip-test-unit --skip-prototype

--skip-test-unit == don’t auto generate blank test files??

--skip-prototype == don’t use Prototype javascript files (guess we’re going to use jquery)

bundle install

identical app/controllers/blog\_controller.rb

route get "blog/index"

invoke erb

exist app/views/blog

identical app/views/blog/index.html.erb

invoke test\_unit

create test/functional/blog\_controller\_test.rb

invoke helper

identical app/helpers/blog\_helper.rb

invoke test\_unit

create test/unit/helpers/blog\_helper\_test.rb

SHOULD WE SKIPPING TESTS – Anyone else seeing this?

Looks like bug in rails 3.0 that was fixed later?

Generate Controller -> View -> Controller -> Model

**Adding blog entries**

Tdd model test running 0.000831 seconds! On my machine. Now that’s what I’m talking about, previously trying to TDD in rails I could not get into any regular “flow” it seemed to be constant waiting for rails to spin up.

ruby spec/models/blog\_spec.rb

Loaded suite spec/models/blog\_spec

Started

E

Finished in 0.000831 seconds.

1) Error:

test\_0001\_has\_no\_entries(BlogSpec):

NoMethodError: undefined method `entries' for #<Blog:0x000001008b0af0>

spec/models/blog\_spec.rb:10:in `block (2 levels) in <main>'

1 tests, 0 assertions, 0 failures, 1 errors, 0 skips

Get the test passing:

ruby spec/models/blog\_spec.rb

Loaded suite spec/models/blog\_spec

Started

.

Finished in **0.000656** seconds.

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

Test run options: --seed 50188

RSPEC on the same test:

rspec ./spec/models/blog\_spec.rb

.

Finished in **0.00032** seconds

1 example, 0 failures

As well as the speed of these tests Avidi goes on to say:

By writing “plain old tests” which don’t rely on any special Rails helpers, we keep the tests isolated and force ourselves to be deliberate about creating any dependencies between our objects. We anticipate that this will have a salutary effect on the object design which emerges from our TDD process.

As a welcome side effect, the tests run ridiculously fast.

**Placeholder blog entries**

We’re back to making changes directly in the controller (no tests???)

The code we’ve written in the controller has shown we need 2 new methods Blog#new\_post and Blog#publish

**Making new entries**

**class Blog** *# ...*

**attr\_writer** :post\_source *# ...*

**private**

**def** post\_source

@post\_source ||= Post.public\_method(:new) **end**

**end**

???Changing “production” code to allow it to be tested???

Why Post.public\_method(:new) and not Post.new ???

???What do you think of the new mini spec syntax copied from rspec???

describe Blog **do**

subject { Blog.new(->{entries}) }

let(:entries) { [] }

it "has no entries" **do**

subject.entries.must\_be\_empty

**end**

*# ...*

**end**

**Posts vs. Entries**

No sure about this I’d probably use Blog.posts what do you think?

The upgrade goes easily, and everything is working great. Then one day a clever user realizes that if he can guess the ID of someone else’spost, he can modify its content! Why is this possible? Because all those calls to Post.find() bypassed any scope constraints imposed by accessing posts from a Blog instance instead of fetching them directly.

??? Having an aggregate root get around this problem???

**The Post class**

We’re back to test first.

Not sure I would test the reading and writing of the attributes… But I would Definitely test the use of the publish method.

Reading and writing of blog is actually tested twice once as an attribute then again in the specific publish test.

Using mocks here to test in complete isolation for the Blog class. True “unit” test.

ANTHONY: I did not get the require mocks problem you mentioned..? Worked as is on OSX.

**Why “publish”?**

Agree with this one sounds like using the ubiquitous language to me.

**Adding entries to the blog**

Driving out Post has revealed that we need one more method on Blog, one which will actually add the post to the blog. We’ll quickly spec it out and add it.

???Did you notice we didn’t get a failing test here as we are using mock objects…. We’re in mock la la land! ??? For me these are easily missed although in this case the site would fall over as soon as you go to add a blog “entry”.

**NOTE: When writing these specs I really want a quick combo key to run the method/class spec. I think it would really improve my flow. If I can’t have this maybe some sort of autotest/spec runner would do…**

entries << post

???Looking at the code we have only defined an attr\_reader for entries yet we can append to the array????

Listing 18: Implementing Blog#new\_post Now we’re passing the arguments along, but we still need to implement keyword arguments on the Post initializer.

NOTE: we’re back in mock la la land…

Listing 20 this is where Post.new stops working and I had to change back to :

Post.public\_method(:new)

Any idea why???

NOTE: Creating partial view. Ruby Mine has a nice auto create partial here.