Testing

Gavin Stark
@gavinstark
gavin@gstark.com

Testing

The rabbit hole is quite deep

Many opinions

- Should I test?
- What should I test?
- When should I test?
- What tools should I use to test?

WARNING: Wars Ahead

MiniTest

TestUnit

RSpec

Cucumber

TDD

BDD

Testing

My path to testing

- Started testing with a large code base
  - 200,000+ lines of code
  - Safety critical system
  - ***Everything*** had to be tested
- Some unit testing
- Some functional testing
- Some integration testing
- Formal linking of requirements to test

Ruby Testing

Ruby (and Rails) projects emphasise testing

Makes testing a prime feature

Rails generates tests from scaffold

Hard to start

Defect Driven Development

WHY do we test

WARNING: MANY opinions here (disclaimer: these are mine)

Test as Design

Tests are first “client” of your code

Hard to test is probably hard to use/change

Helps understand a new problem/feature/API

Tests to Ensure You Don’t Break Stuff

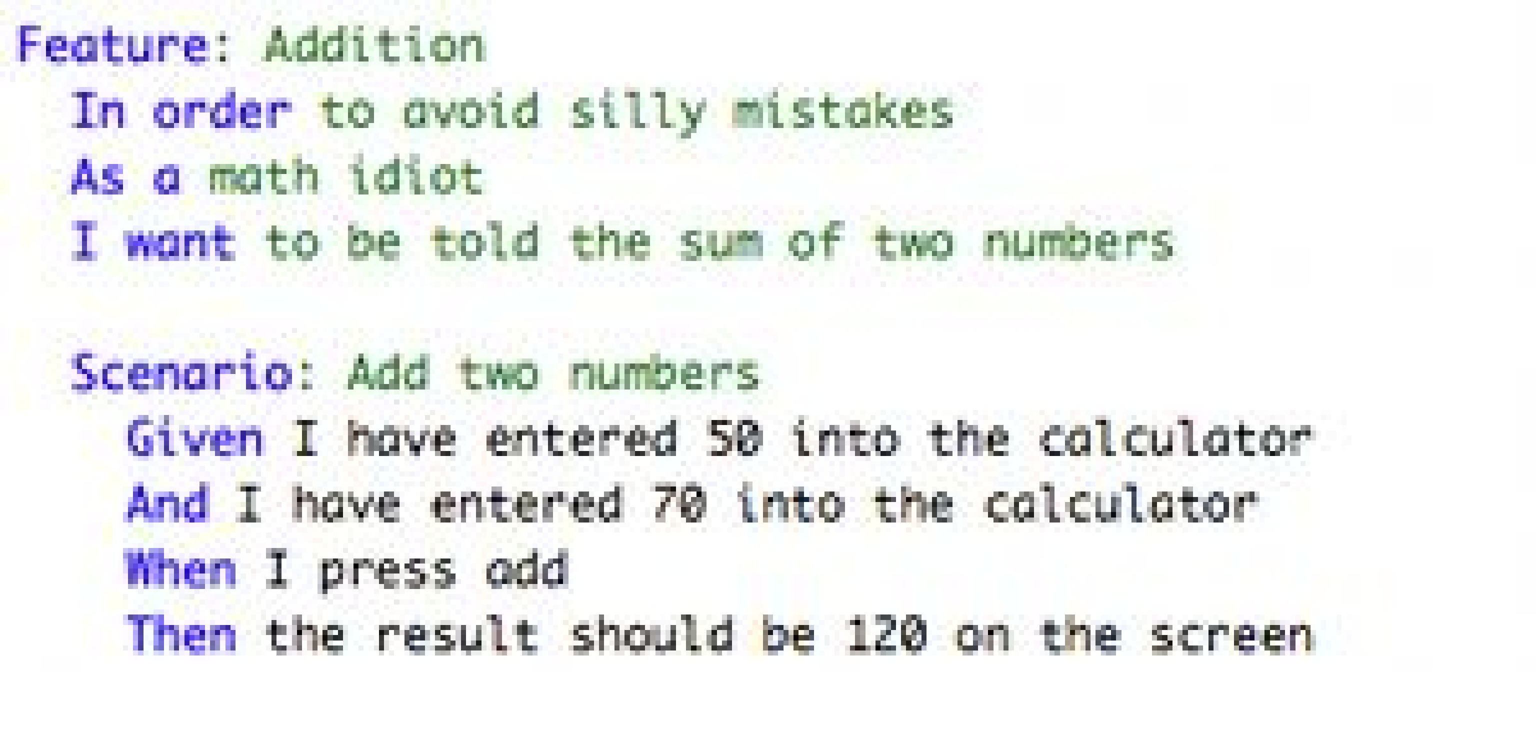
Famous last words: “I should be able to change this feature here without breaking something waaaaaaay over there”

Not always true (e.g. change a method signature/implementation not knowing another class depends on it)

Tests help ensure this.

Test as Acceptance Check

Does the function / class / controller / view / application *do* what it is supposed to.



```
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
```

Tests For Security

Ensure customer A doesn’t see customer B’s data

Test that users must login

Test that user’s can’t post malicious data

Unit Testing

- Testing small units of code
    - In isolation
    - Defined inputs, expected outputs

Testing Fundamentals

Setup environment

Supply inputs

Execute code

Gather output

Compare to expected results

… and Assert

Testing Fundamentals

Unit Test

Integration Test

Functional Test

Unit Testing

Demo

Integration Testing

Tests multiple components

Requires more setup

Input to first part of system

Tests outputs of last part of system

Integration Testing

Demo

Functional Testing

Does the system do what it is supposed to?

- Satisfies user’s requirements
- System behaves as it should

Functional Testing

Demo

Performance Testing

Do pages load fast enough?

Does your API return results within the service level agreements the salespeople sold?

How many background jobs per hour can you do?

How long does a new user’s avatar take to show up?

Minitest

By the seattle.rb group -- now included in ruby

Default Rails tests based on these

Minitest

assert / refute / assert_equal /....

https://github.com/seattlerb/minitest

http://docs.seattlerb.org/minitest/Minitest/Assertions.html

RSpec

“DSL” (Domain Specific Language) framework

All of the same features as minitest but syntactically “prettier”

- note Minitest has a “spec” style implementation too

RSpec

Demo

Rails Testing

Built in testing support

test/ directory when creating scaffold

Not a bad place to start:

http://guides.rubyonrails.org/testing.html

Lets walk that guide!

Github Repo to Clone

https://github.com/tiy-tpa/TheIronYardTesting

9e592837d51a2c92b5ba46d52cca5fc15024bf69 Rails new

Just a plain rails app

62443ec794be6f560a5a73e92b08c31cbd0e837c

```
rails generate scaffold article title:string body:text && rake db:migrate
```

fa6e7321d7fea83cec54e90a5d04ebd27eda6433

Adds a failing test

TDD - Test Driven

RED

- Add failing test
- Add/Change the code to “change the message”

GREEN

- Add code to make test pass

REFACTOR

- Change code (keeping green) to make the code better

Very fast / tight loops is the goal

e6595cce1472f715b6f90a437c86736fc9034115

Improves test failure message

220dc515ed6f319131f23a524465abf3aaee95f2

Adds code to make test pass

94faac0bf5452a06f9781aa4bdd582df5f21c496

Adds checking of flash message to a functional test

No RED phase here since tests already exists

How can we make sure we are *really* testing something.

Lets Add New Feature Driven By Tests

Adding SEARCH to articles

6566c8cfc095f431d562019050 25d96cfdd0dda9

Adds failing test

Fixtures!

Not sure where to go?

Read the error message and try to “fix” it

SMALL steps

Error says “no route” -- so lets add the route.

8472d97097a100a31945cc52cd06378e2406775e

Adds route to routes.rb -- changes the message -- now error is that there is no action

No action? Lets add the action itself, but NO CODE inside.

ca7f9190f188f7afbaffa4b42238417bf2e8de7a

Adds action -- changes message -- now complaining that we are missing a view template

Hmm, must need a view, lets add the view but NOTHING inside.

746e7ec02effa6a3004497d1edac7b7e287a1ff8

Adds an empty view -- finally an error that what was expected didn't match the actual

Finally!!!!!1!!

Seem like a lot of work?

e1cae8f85b7c9f83de0abbe38b1fb7a7ea0cf378

Adds query to test - adds implementation to search -- tests passing

Now, at last, writing *production* code.

5f2bcce53402d288635a2f964ede598ff126cdfd

Adds failing view test

Now lets make sure the articles show up on the search result page

95281afe76c54b461b48d4dc34d91391baef5c67

Adds view to make test pass

c9835cabfe7c7edcda49eb5b799b6e591d32c3f3

Adds test to require the title of the article be shown in the list

594a61e391e2d675dd1c4ffb3f7616ceb7db47aa

Updates view so tests pass

What have we just done

Added new feature

Tests exist to make sure it doesn’t break

Confirmed that there is something visible on a search result page

What ***DIDN’T*** we do?

Never loaded a browser….

dd72194df5b4da4f70ea9e959805c61d64809eb7

```
rails generate integration_test article_list_flow
```

Add test to visit /articles

b0f69e4ce7bcde1856a87a4b5896fb3ab218e2f3

Add failing test to ensure a “search” dom element (could be link/form/etc)

5c78a97e0d94f80d48735ccaa8ae91dfbdd7589f

Adds a search form

Other integration tests

Successful login

Unsuccessful login

Logged in users only see their own books (security)