Object Oriented WEB Programming Using Ruby

Day 6: 24/May/2012

TDD (Test Driven Development)

Today's Goal

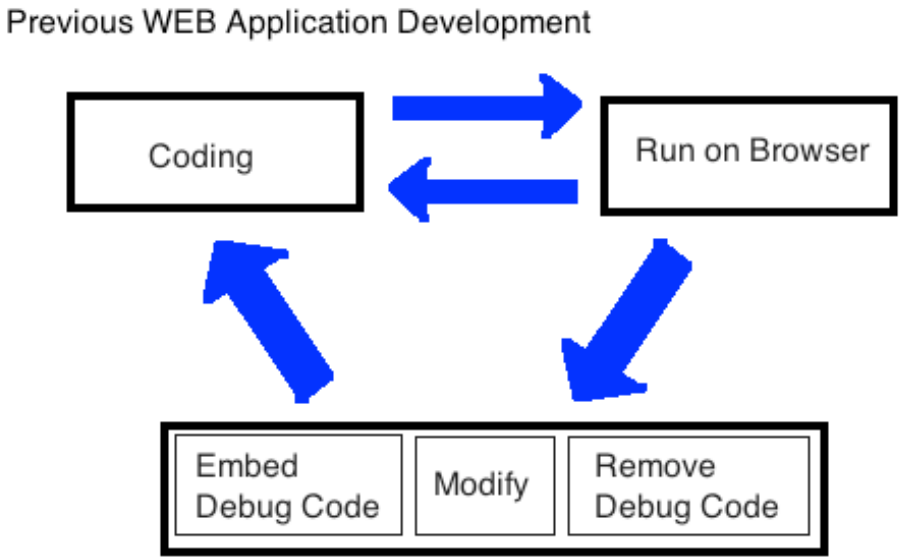
- Understand what TDD (Test Driven Development) is.
  - Understand the words related to the Test Driven Development
  - Get used to the 'Rails-Way' of TDD
- We apply TDD to the "guests" table, which we had generated last week, to get the sketch of TDD.

What TDD is?

- Test Driven Development
  - Method, and the environment.
- One thing we should learn the most, when we use Ruby on Rails environment.
  - On rails, it is so easy to use.
- We can obtain the highly-proved source codes, also.
- http://guides.rubyonrails.org/testing.html

Our previous WEB application Development

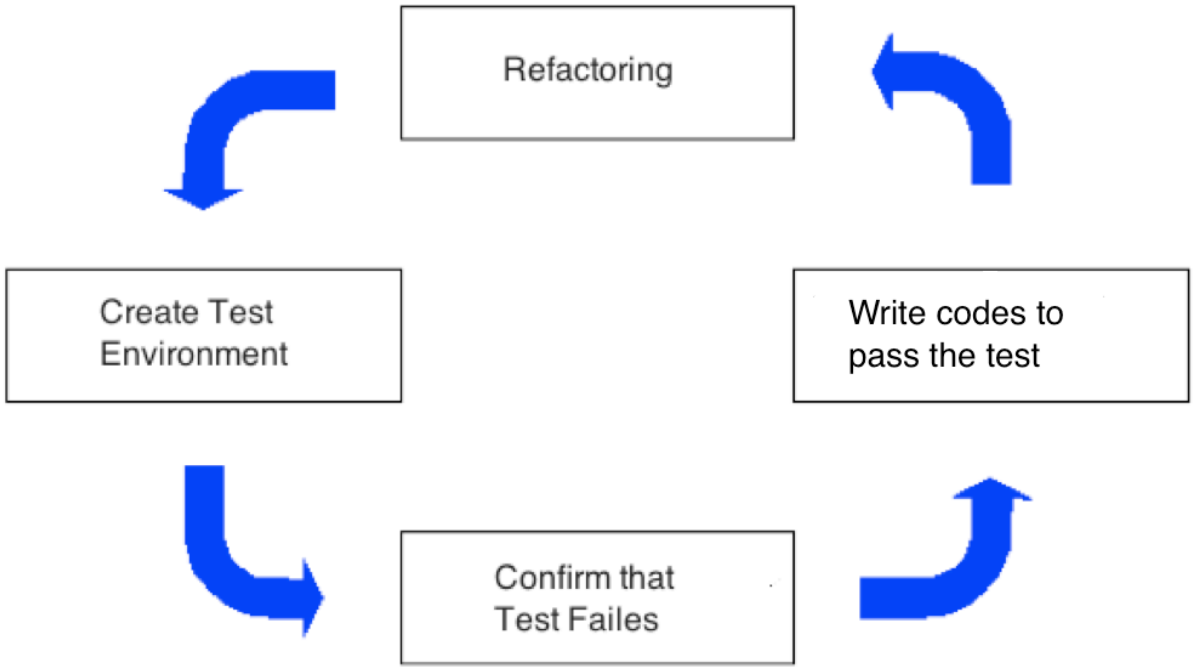
- We had written the code, run, and checked with its actual runtime environment.



P304, Kiso Ruby on Rails, Impress Japan, 2007

Procedure of TDD

- Create Test Environment First.



P304, Kiso Ruby on Rails, Impress Japan, 2007

About RSpec …

It is often mentioned that RSpec is better than original Testing Environment of Rails.

http://ruby.railstutorial.org/chapters/static-pages#sec:TDD

In this course, we use original Test Environment.

- Because, I am not get used so much to explain RSpec, but there is no other reason.
- So, if you could, try to apply RSpec.
- I may modify this course material to use RSpec.

In Test, “Failure” has meaning

- We prepare “Test” before we start writing programs.
- We Test first before we write program, the test should “Fail” and that proves the “Test works properly.”
- Here the “Failure” is not an “Error”

4 steps to introduce Test

- Step 1: Write "Test"
  - Make Specification Clear, and write "Test" according to how it should work.
- Step 2: Confirm it "fails" before writing program.
  - Prepare "Test Script" and execute test to prove it works before writing programs. (Debug the test script.)
- Step 3: Coding
  - So that the program passes the test
- Step 4: Refactoring
  - Keep it passes the test, and clean the source code.

Prepare Database for Test

- On Rails, we use 3 databases.
  - Open config/database.yml, and find
    - db/development.sqlite3
    - db/test.sqlite3
    - db/production.sqlite3

- Type the following command,

  rake db:test:prepare

  - When it runed successfully, ‘test.sqlite3’ is generated in db folder.

What kinds of test? (1/2)

- Unit Test
  - If model 'search' methods obtained data properly,
  - If model 'update' methods obtained data properly,
  - If model 'update' methods showed error messages properly against improper values.

- Functional Test
  - If proper template was selected,
  - If proper values were stored into instance variables,
  - If rendered properly, or redirected properly,
  - If model 'update' actions updated database properly.

What kinds of test? (2/2)

- Integrated Test
  - All operations work properly.

- Test related files are stored in test directory.

```
kobayashi-ikuo-no-MacBook:test kobayashi$ ls -Fal
total 8
drwxr-xr-x   8 kobayashi  staff  272  5  4 15:22 ./
drwxr-xr-x  20 kobayashi  staff  680  5  4 15:58 ../
drwxr-xr-x   5 kobayashi  staff  170  5 10 01:27 fixtures/
drwxr-xr-x   5 kobayashi  staff  170  5 10 01:27 functional/
drwxr-xr-x   3 kobayashi  staff  102  5  4 15:22 integration/
drwxr-xr-x   3 kobayashi  staff  102  5  4 15:22 performance/
-rw-r--r--   1 kobayashi  staff  454  5  4 15:22 test_helper.rb
drwxr-xr-x   6 kobayashi  staff  204  5 10 01:27 unit/
kobayashi-ikuo-no-MacBook:test kobayashi$
```

File for Test

- See the file test/unit/guest_test.rb, which had been generated automatically.

```ruby
require 'test_helper'

class GuestTest < ActiveSupport::TestCase
  # test "the truth" do
  #   assert true
  # end
end

```

```
$ rails generate scaffold memo title:text neme:string
      invoke  active_record
      create    db/migrate/20120425151305_create_memos.rb
      create    app/models/memo.rb
      invoke    test_unit
      create      test/unit/memo_test.rb
      create      test/fixtures/memos.yml
       route  resources :memos
      invoke  scaffold_controller
      create    app/controllers/memos_controller.rb
      invoke    erb
      create      app/views/memos
      create      app/views/memos/index.html.erb
      create      app/views/memos/edit.html.erb
      create      app/views/memos/show.html.erb
      create      app/views/memos/new.html.erb
      create      app/views/memos/_form.html.erb
      invoke    test_unit
      create      test/functional/memos_controller_test.rb
      invoke    helper
      create      app/helpers/memos_helper.rb
      invoke      test_unit
      create        test/unit/helpers/memos_helper_test.rb
      invoke  assets
      invoke    coffee
      create      app/assets/javascripts/memos.js.coffee
      invoke    scss
      create      app/assets/stylesheets/memos.css.scss
      invoke  scss
      create    app/assets/stylesheets/scaffolds.css.scss

admin@ADMIN-501MOKPBO /cygdrive/c/Users/admin/My Documents/Aptana Studio 3 Works
pace/memopad
$
```

test/unit/guest_test.rb

```
require 'test_helper'

class GuestTest < ActiveSupport::TestCase
  test "the truth" do
    assert true
  end
end
```

- At the beginning, there is a test which will success always. Let us un-comment the test “the truth”.

What is to assert?

- Assertion is to check if any condition were true.
  - The ‘not null’ field is not empty.
  - The value is within the given range.

Run guest_test.rb anyway

- Perform test for only one file,

- Type the following command

  ruby –Itest test/unit/guest_test.rb

```
kobayashi-ikuo-no-MacBook:spielberg kobayashi$ ruby -Itest test/unit/guest_tes
t.rb
Loaded suite test/unit/guest_test
Started
.
Finished in 0.031848 seconds.

1 tests, 1 assertions, 0 failures, 0 errors
kobayashi-ikuo-no-MacBook:spielberg kobayashi$
```

4 result values are given

- They are

  Test, Assetions, Failures, Errors

  - Test：Number of test methods
  - Assertions: Number of Assertion methods.
  - Failures: Number of Failed assertions.
  - Errors：Number of bugs of test methods description, database error, and such trouble with test preparation.

If you face with ERROR now…

- We have not written any program yet, so the result must be NO failure, NO error.

- If you face with error now, we can assume the following reason.

  You have not run

  ```
  rake db:test:prepare
  ```

  yet, or development.sqlite3 had not been created yet by the last class.

  You had not migrated yet.

Once again, the difference between Failure and Error

"Failure":

- Program failed to judge the abnormal data as "unusual," and tried to hand it in to database,
- Give the "ordinary" data to the system, but the system failed to recognize that the data had been "normal."

- Both the case, we should decide that the there were "mis-programming."

"Error"

- Either related file, Test data, and test description itself may contain grammatical or semantic error.

Today's Theme

- Update

  test\unit\guest_test.rb

  - Try to describe the "Perfect" test against the data and system error.
- Use the features data as default.
- Remove all the errors from the result test execution.
- Also, we finish installing all other tables than guests.

guests.yml

```
# Read about fixtures at http://
api.rubyonrails.org/classes/
ActiveRecord/Fixtures.html

one:
  login: koba@hosei.com
  age: 20
  sex: 1

two:
  login: yashi*hosei.or.jp
  age: 200
  sex: 2

three:
  login: iku+o@hosei@example.jp
  age:
  sex: 3

four:
  login:
  age: 5
  sex:
```

test\unit\guest_test.rb

```
require 'test_helper'

class GuestTest < ActiveSupport::TestCase
  fixtures :guests
  test "the truth" do
    assert true
  end
  test "data should be valid" do
    reg = Regexp.new("^([a-zA-Z0-9_.%+-]+)@([a-zA-Z0-9.-]+?)(¥.[a-zA-Z0-9_.-]*)$")
    data = guests(:one)
    assert( data.valid?, "data one should be valid" )
    assert_not_nil( data.login, "login of data one should be not nil" )
    assert_match(  reg, data.login, "data one login address should match." )
    assert_not_nil( data.age, "age of data one should be not nil" )
    assert_not_nil( data.sex, "sex of data one should be not nil" )
    data = guests(:two)
    assert_no_match(  reg, data.login, "data two login address should not match." )
    assert(data.age<1 || data.age>130, "age of data two should be out of rang [1..130]")
    data = guests(:three)
    assert_no_match(  reg, data.login, "data three login address should not match." )
    assert_nil( data.age, "age of data three should be nil" )
    assert_not_nil( data.sex, "sex of data three should be not nil" )
    assert(data.sex<1 || data.sex>2, "sex of data three should be out of range[1..2]")
  end
end
```

Assertions Available (1/3)

Assertion	Purpose
`assert( boolean, [msg] )`	Ensures that the object/expression is true.
`assert_equal( obj1, obj2, [msg] )`	Ensures that `obj1 == obj2` is true.
`assert_not_equal( obj1, obj2, [msg] )`	Ensures that `obj1 == obj2` is false.
`assert_same( obj1, obj2, [msg] )`	Ensures that `obj1.equal?(obj2)` is true.
`assert_not_same( obj1, obj2, [msg] )`	Ensures that `obj1.equal?(obj2)` is false.
`assert_nil( obj, [msg] )`	Ensures that `obj.nil?` is true.
`assert_not_nil( obj, [msg] )`	Ensures that `obj.nil?` is false.
`assert_match( regexp, string, [msg] )`	Ensures that a string matches the regular expression.

Assertions Available (2/3)

Assertion	Purpose
`assert_no_match( regexp, string, [msg] )`	Ensures that a string doesn't match the regular expression.
`assert_in_delta( expecting, actual, delta, [msg] )`	Ensures that the numbers `expecting` and `actual` are within `delta` of each other.
`assert_throws( symbol, [msg] ) { block }`	Ensures that the given block throws the symbol.
`assert_raise( exception1, exception2, ... ) { block }`	Ensures that the given block raises one of the given exceptions.
`assert_nothing_raised( exception1, exception2, ... ) { block }`	Ensures that the given block doesn't raise one of the given exceptions.
`assert_instance_of( class, obj, [msg] )`	Ensures that `obj` is of the `class` type.

Assertions Available (3/3)

Assertion	Purpose
`assert_kind_of( class, obj, [msg] )`	Ensures that `obj` is or descends from `class`.
`assert_respond_to( obj, symbol, [msg] )`	Ensures that `obj` has a method called `symbol`.
`assert_operator( obj1, operator, obj2, [msg] )`	Ensures that `obj1.operator(obj2)` is true.
`assert_send( array, [msg] )`	Ensures that executing the method listed in `array[1]` on the object in `array[0]` with the parameters of `array[2 and up]` is true. This one is weird eh?
`flunk( [msg] )`	Ensures failure. This is useful to explicitly mark a test that isn't finished yet.

Regular Expr. for Mail Address

We use assert_match() to check the email-address using regular expression.

Regular Expression for mail address is

/^[a-zA-Z0-9_.%+\-]+@[a-zA-Z0-9.-]+?(\.[a-zA-Z0-9_.\-]*)$/

If our system do not allow using ‘%’ or ‘+’ in mail address, the regular expression would be

/^[a-zA-Z0-9_.\-]+@[a-zA-Z0-9.-]+?(\.[a-zA-Z0-9_.\-]*)$/

The result of test

- Type

  ruby -I test test/unit/guest_test.rb

- When it passes, the result should be

  0 failures, 0 errors.

```
kobayashi-ikuo-no-MacBook:spielberg kobayashi$ ruby -I test test/unit/guest_test
.rb
Loaded suite test/unit/guest_test
Started
F.
Finished in 0.054051 seconds.

  1) Failure:
test_data_should_be_valid(GuestTest) [test/unit/guest_test.rb:20]:
data three login address should not match.
</^[a-zA-Z0-9_.%+-]+@[a-zA-Z0-9.-]+?(.[a-zA-Z0-9_.-]*)$/> expected to not match
<"iku+o@hosei@example.jp">.

2 tests, 9 assertions, 1 failures, 0 errors
kobayashi-ikuo-no-MacBook:spielberg kobayashi$
```

Regular Expr for mail address

Well, the result shown in the previous page was not the result I had been expected.

Apparently, I should “debug” the regular expression of mail address of

```
/^[a-zA-Z0-9_.%+\-]+@[a-zA-Z0-9.-]+?(\.[a-zA-Z0-9_.\-]*)$/
```

But… time out in preparing course material (this file) for the lecture…

You, brilliant guys, please fix it and let me know. Thanks in advance. The below is an useful page.

http://www.regular-expressions.info/email.html

Functional Test

- While Unit Test was to test the Model part, Functional Test is to check the controller part.

- When we generate scaffold, under test/ functional/ directory,
  - XXXXX_controller_test.rb is generated.

Let see, guests_controller_test.rb

test/functional/guests_controller_test.rb

- Automatically Generated Test

```ruby
require 'test_helper'

class GuestsControllerTest < ActionController::TestCase
  setup do
    @guest = guests(:one)
  end

  test "should get index" do
    get :index
    assert_response :success
    assert_not_nil assigns(:guests)
  end

  test "should get new" do
    get :new
    assert_response :success
  end

  test "should create guest" do
    assert_difference('Guest.count') do
      post :create, :guest => { :age => @guest.age, :login => @guest.login, :sex => @guest.sex }
    end

    assert_redirected_to guest_path(assigns(:guest))
  end
```

Rendering, and Redirection

What is Rendering?

- When a template is chosen by “Action”, values from controllers are embedded in HTML source code. This is “rendering”.

What is Redirection?

- Force to show new URL

Test of Rendering and Redirection

In Functional Test, presences of parameters and validity of values are checked, before they are embedded in html.

```
37  test "should update guest" do
38    put :update, :id => @guest, :guest => { :age => @guest.age, :login => @guest.login, :sex => @guest.sex
39    assert_redirected_to guest_path(assigns(:guest))
40  end
41
42  test "should destroy guest" do
43    assert_difference('Guest.count', -1) do
44      delete :destroy, :id => @guest
45    end
46
47    assert_redirected_to guests_path
48  end
```

Integration Test

- It requires the total flow description, such as “login → update database → logout,” so integration test cannot be generated automatically.

Perform all tests

- Type

  rake test

```
C:¥Users¥Ikuo¥work¥KjgsLearning>rake test
(in C:/Users/Ikuo/work/KjgsLearning)
Loaded suite C:/Ruby192/lib/ruby/1.9.1/rake/rake_test_loader
Started
..
Finished in 6.068400 seconds.

2 tests, 6 assertions, 0 failures, 0 errors, 0 skips

Test run options: --seed 62694
Loaded suite C:/Ruby192/lib/ruby/1.9.1/rake/rake_test_loader
Started
..............
Finished in 7.176000 seconds.

14 tests, 20 assertions, 0 failures, 0 errors, 0 skips

Test run options: --seed 41870

C:¥Users¥Ikuo¥work¥KjgsLearning>
```

Report themes for today

none

Prepare for the Next Week

We will learn Database Access via model.

We will write codes to describe relational links between tables, for the Problem Solving Engine.