

```
context "Testing" do
  should "be fun" do
    ...
  end
end
```

David Lowenfels
david@internautdesign.com



1/22/08 Ruby Meetup

Shoulda

<http://www.thoughtbot.com/projects/shoulda>

- Released by Tammer Saleh last August.
- A wrapper for Test::Unit.
- “Shoulda makes it easy to write elegant, understandable, and maintainable tests.”
- BDD flavor without the fat.

Why?

Shoulda helps you make your tests more:

- readable
- semantic
- organized
- DRY
- Fun!

“Stop killing your fingers with all of those underscores...
Name your tests with plain sentences!”

```
def test_up_success
```

```
  ...
```

```
end
```

ever seen cryptic tests
named like this before? :-p

```
def test_#up!_increments_counter
```

```
  ...
```

```
end
```

better...

```
context “A Foo class” do
```

```
  should “increment it’s counter when sent #up!” do
```

```
    ...
```

```
  end
```

```
end
```

wow!

Organize your tests by context.

```
context "A Foo class" do
  should "increment it's counter when sent #up!" do
    ...
  end
  should "decrement it's counter when sent #down!" do
    ...
  end
end
```

```
context "A Foo instance" do
  should "return it's value when sent #value" do
    ...
  end
end
```

Context blocks can be nested.

```
class UserTest < Test::Unit::TestCase

  context "A User instance" do
    setup do
      @user = User.find(:first)
    end

    should "return its full name"
      assert_equal 'John Doe', @user.full_name
    end

    context "with a profile" do
      setup do
        @user.profile = Profile.find(:first)
      end

      should "return true when sent #has_profile?"
        assert @user.has_profile?
      end
    end
  end
end

end
```

(DRY)

setup and teardown
blocks as needed

Outputs standard Test::Unit code

```
define_method "test: A User instance should return its full name." do
  @user = User.find(:first)
  assert_equal 'John Doe', @user.full_name
end
```

```
define_method "test: A User instance with a profile should return true
when sent #has_profile?." do
  @user = User.find(:first)
  @user.profile = Profile.find(:first)
  assert @user.has_profile?
end
```

therefore can be mixed and
matched inline with Test::Unit

Rails-specific Macros

- `should_have_many`
 - `should_belong_to`
 - `should_be_restful`
 - `should_render_template`
-
- Model
- Controller
- ```
graph LR; Model --> should_have_many; Model --> should_belong_to; Controller --> should_be_restful; Controller --> should_render_template;
```

# Controller Macros

```
context "on GET to :show for first record" do
 setup do
 get :show, :id => 1
 end

 should_assign_to :user
 should_respond_with :success
 should_render_template :show
 should_not_set_the_flash

 should "do something else really cool" do
 assert_equal 1, assigns(:user).id
 end
end
```

# Custom Assertions

- `assert_same_elements [:a, :b, :c], [:c, :a, :b]`
- `assert_contains ['a', '1'], /\d/`
- `assert_contains ['a', '1'], 'a'`

# Pros:

- Low cost of entry.
- Lightweight.
- no deep Object hacks needed for fancy DSL.

# Cons:

- Stack trace gets funky due to context metaprogramming.

(fixed with quietbacktrace gem)

- No Textmate support.

(until I wrote a bundle)

# Sources

- <http://www.thoughtbot.com/projects/shoulda>
- <https://svn.thoughtbot.com/plugins/shoulda/trunk/>
- [http://blog.internautdesign.com/2007/11/2/a-yaml\\_to\\_shoulda-rake-task](http://blog.internautdesign.com/2007/11/2/a-yaml_to_shoulda-rake-task)