

**Maintaining balance**  
**while**  
**reducing duplication**

David Chelimsky

<http://drw.com>



DRW TRADING GROUP

About Us

Trading

Technology

<http://rubygems.org/gems/rspec>

# rspec 2.1.0

Meta-gem that depends on the other rspec gems

**INSTALL** > `gem install rspec`

 Edit  Download  Subscribe  Stats

<http://relishapp.com/rspec>



RSpec

## rspec (RSpec)

RSpec Expectations

Versions: 2.1, 2.0

RSpec Mocks

Versions: 2.1, 2.0

RSpec Rails

Versions: 2.1, 2.0

# Shared example group

Shared example groups let you describe behaviour of types or modules. When declared, a shared group's content is stored. It is only realized in the context of another example group, which provides any context the shared group needs to run.

A shared group is included in another group using the `it_behaves_like()` or `it_should_behave_like()` methods.

## **Scenario:** shared example group applied to two groups

Given a file named "collection\_spec.rb" with:

```
require "set"

shared_examples_for "a collection" do
  let(:collection) { described_class.new([7, 2, 4]) }
end
```

<http://pragprog.com/titles/achbd/the-rspec-book>

The  
Pragmatic  
Programmers

At printer on Nov 12!

# The RSpec Book

Behaviour Driven Development  
with RSpec, Cucumber,  
and Friends

*David Chelimsky*  
with *Dave Astels,*  
*Zach Dennis,*  
*Aslak Hellesøy,*  
*Bryan Helmkamp,*  
and *Dan North*

*Edited by Jacquelyn Carter*

The Facets  of Ruby Series

This talk is **not** about  
**RSpec**

**This talk **is** about  
**duplication****

**About **duplication,**  
**is** this talk**

**Duplication is bad**

# Code smell: Duplicated Code

**“Duplicate code is the  
root of all evil in  
software design.”**

# The **DRY** principle

**Don't**  
**Repeat**  
**Yourself**



ruby rspec

About 2,540,000 results (0.28 seconds)



ruby rspec dry|

About 154,000 results (0.21 seconds)

6%



ruby on rails

About 3,820,000 results (0.19 seconds)



ruby on rails dry

About 103,000 results (0.22 seconds)

3%



ruby language

About 9,850,000 results (0.34 seconds)



ruby language dry

About 1,120,000 results (0.34 seconds)

11%

**Custom Matchers :**  
**Shouldn't they follow**  
**DRY** principle ?

```
describe WidgetsController do
  describe "POST create" do
    it "returns 201 (created)"
      post :create, :widget => { :name => 'ACME Widget' }
      response.should be_created
    end
  end
end
```

```
describe WidgetsController do
  describe "POST create" do
    it "creates the widget and returns 201 (created)" do
      post :create, :widget => { :name => 'ACME Widget' }
      widget = Widget.find_by_name('ACME Widget')
      response.should have_created_resource(widget)
    end
  end
end
```

```
describe WidgetsController do
  describe "POST create" do
    it "returns 201 (created)"
      post :create, :widget => { :name => 'ACME Widget' }
      response.should be_created
    end
  end
end
```

```
describe WidgetsController do
  describe "POST create" do
    it "creates the widget and returns 201 (created)" do
      post :create, :widget => { :name => 'ACME Widget' }
      widget = Widget.find_by_name('ACME Widget')
      response.should have_created_resource(widget)
    end
  end
end
```

```
RSpec::Matchers.define :be_created do
  match do |response|
    response.code == "201"
  end

  failure_message_for_should do |response|
    "expected the response code to be 201 (created) but was #{response.code}"
  end
end

RSpec::Matchers.define :have_created_resource do |resource|
  match do |response|
    resource && response.code == "201"
  end

  failure_message_for_should do |response|
    if response.code != "201"
      "expected the response code to be 201 (created) but was #{response.code}"
    else
      "expected the resource not to be nil"
    end
  end
end
end
```

```
RSpec::Matchers.define :be_created do
  match do |response|
    response.code == "201"
  end
end
```

```
failure_message_for_should do |response|
  "expected the response code to be 201 (created) but was #{response.code}"
end
end
```

```
RSpec::Matchers.define :have_created_resource do |resource|
  match do |response|
    resource && response.code == "201"
  end
end
```

```
failure_message_for_should do |response|
  if response.code != "201"
    "expected the response code to be 201 (created) but was #{response.code}"
  else
    "expected the resource not to be nil"
  end
end
end
end
```

```
RSpec::Matchers.define :be_created do
  match do |response|
    response.code == "201"
  end
end
```

```
failure_message_for_should do |response|
  "expected the response code to be 201 (created) but was #{response.code}"
end
end
```

```
RSpec::Matchers.define :have_created_resource do |resource|
  match do |response|
    resource && response.code == "201"
  end
end
```

```
failure_message_for_should do |response|
  if response.code != "201"
    "expected the response code to be 201 (created) but was #{response.code}"
  else
    "expected the resource not to be nil"
  end
end
end
end
```

```
RSpec::Matchers.define :be_created do
```

```
  match do |response|
```

```
    response.code == "201"
```

```
  end
```

```
  failure_message_for_should do |response|
```

```
    "expected the response code to be 201 (created) but was #{response.code}"
```

```
  end
```

```
end
```

```
RSpec::Matchers.define :have_created_resource do |resource|
```

```
  match do |response|
```

```
    resource && response.code == "201"
```

```
  end
```

```
  failure_message_for_should do |response|
```

```
    if response.code != "201"
```

```
      "expected the response code to be 201 (created) but was #{response.code}"
```

```
    else
```

```
      "expected the resource not to be nil"
```

```
    end
```

```
  end
```

```
end
```

```
RSpec::Matchers.define :be_created do
  match do |response|
    response.code == "201"
  end

  failure_message_for_should do |response|
    "expected the response code to be 201 (created) but was #{response.code}"
  end
end
```

```
RSpec::Matchers.define :have_created_resource do |resource|
  match do |response|
    if !be_created.matches?(response)
      @message = be_created.failure_message_for_should(response)
    elsif !be_nil.matches?(resource)
      @message = "expected the resource not to be nil"
    end
  end
end

  failure_message_for_should do |response|
    @message
  end
end
```

```
RSpec::Matchers.define :be_created do
  match do |response|
    response.code == "201"
  end
end
```

```
failure_message_for_should do |response|
  "expected the response code to be 201 (created) but was #{response.code}"
end
end
```

```
RSpec::Matchers.define :have_created_resource do |resource|
  match do |response|
    if !be_created.matches?(response)
      @message = be_created.failure_message_for_should(response)
    elsif !be_nil.matches?(resource)
      @message = "expected the resource not to be nil"
    end
  end
end
```

```
failure_message_for_should do |response|
  @message
end
end
```

```
RSpec::Matchers.define :be_created do
  match do |response|
    response.code == "201"
  end
end
```

```
failure_message_for_should do |response|
  "expected the response code to be 201 (created) but was #{response.code}"
end
end
```

```
RSpec::Matchers.define :have_created_resource do |resource|
  match do |response|
    if !be_created.matches?(response)
      @message = be_created.failure_message_for_should(response)
    elsif !be_nil.matches?(resource)
      @message = "expected the resource not to be nil"
    end
  end
end
```

```
failure_message_for_should do |response|
  @message
end
end
```

```
RSpec::Matchers.define :be_created do
  match do |response|
    response.code == "201"
  end

  failure_message_for_should do |response|
    "expected the response code to be 201 (created) but was #{response.code}"
  end
end
```

```
RSpec::Matchers.define :have_created_resource do |resource|
  match do |response|
    if !be_created.matches?(response)
      @message = be_created.failure_message_for_should(response)
    elsif !be_nil.matches?(resource)
      @message = "expected the resource not to be nil"
    end
  end
end
```

```
failure_message_for_should do |response|
  @message
end
end
```

**every time you  
reduce duplication  
you  
increase coupling  
by introducing  
new dependencies**

**it is important to  
depend on the  
right things**

# The Dependency Inversion Principle

depend on  
**abstractions**

in **Ruby**, almost  
everything *is* an  
**abstraction**

**depend on things  
which are  
less likely to change**

```
RSpec::Matchers.define :be_created do
  match do |response|
    response.code == "201"
  end

  failure_message_for_should do |response|
    "expected the response code to be 201 (created) but was #{response.code}"
  end
end

RSpec::Matchers.define :have_created_resource do |resource|
  match do |response|
    resource && response.code == "201"
  end

  failure_message_for_should do |response|
    if response.code != "201"
      "expected the response code to be 201 (created) but was #{response.code}"
    else
      "expected the resource not to be nil"
    end
  end
end
```

```
RSpec::Matchers.define :be_created do
```

```
  match do |response|
```

```
    response.code == "201"
```

```
  end
```

```
  failure_message_for_should do |response|
```

```
    "expected the response code to be 201 (created) but was #{response.code}"
```

```
  end
```

```
end
```

```
RSpec::Matchers.define :have_created_resource do |resource|
```

```
  match do |response|
```

```
    resource && response.code == "201"
```

```
  end
```

```
  failure_message_for_should do |response|
```

```
    if response.code != "201"
```

```
      "expected the response code to be 201 (created) but was #{response.code}"
```

```
    else
```

```
      "expected the resource not to be nil"
```

```
    end
```

```
  end
```

```
end
```

```
RSpec::Matchers.define :be_created do
```

```
  match do |response|
```

```
    response.code == CREATED
```

```
  end
```

```
  failure_message_for_should do |response|
```

```
    "expected the response code to be 201 (created) but was #{response.code}"
```

```
  end
```

```
end
```

```
RSpec::Matchers.define :have_created_resource do |resource|
```

```
  match do |response|
```

```
    resource && response.code == CREATED
```

```
  end
```

```
  failure_message_for_should do |response|
```

```
    if response.code != CREATED
```

```
      "expected the response code to be 201 (created) but was #{response.code}"
```

```
    else
```

```
      "expected the resource not to be nil"
```

```
    end
```

```
  end
```

```
end
```

```
RSpec::Matchers.define :be_created do
```

```
  match do |response|
```

```
    response.code.created?
```

```
  end
```

```
  failure_message_for_should do |response|
```

```
    "expected the response code to be 201 (created) but was #{response.code}"
```

```
  end
```

```
end
```

```
RSpec::Matchers.define :have_created_resource do |resource|
```

```
  match do |response|
```

```
    resource && response.code.created?
```

```
  end
```

```
  failure_message_for_should do |response|
```

```
    if !response.code.created?
```

```
      "expected the response code to be 201 (created) but was #{response.code}"
```

```
    else
```

```
      "expected the resource not to be nil"
```

```
    end
```

```
  end
```

```
end
```

```
RSpec::Matchers.define :be_created do
  match do |response|
    response.code.created?
  end
end
```

```
failure_message_for_should do |response|
  "expected the response code to be 201 (created) but was #{response.code}"
end
end
```

```
RSpec::Matchers.define :have_created_resource do |resource|
  match do |response|
    resource && response.code.created?
  end
end
```

```
failure_message_for_should do |response|
  if !response.code.created?
    "expected the response code to be 201 (created) but was #{response.code}"
  else
    "expected the resource not to be nil"
  end
end
end
```

```
RSpec::Matchers.define :be_created do
```

```
  match do |response|
```

```
    response.code.created?
```

```
  end
```

```
  failure_message_for_should do |response|
```

```
    FailureMessages.expected_response_code_for(:created).got(response.code)
```

```
  end
```

```
end
```

```
RSpec::Matchers.define :have_created_resource do |resource|
```

```
  match do |response|
```

```
    resource && response.code.created?
```

```
  end
```

```
  failure_message_for_should do |response|
```

```
    if !response.code.created?
```

```
      FailureMessages.expected_response_code_for(:created).got(response.code)
```

```
    else
```

```
      "expected the resource not to be nil"
```

```
    end
```

```
  end
```

```
end
```

```
RSpec::Matchers.define :be_created do
  match do |response|
    response.code.created?
  end

  failure_message_for_should do |response|
    FailureMessages.expected_response_code_for(:created).got(response.code)
  end
end

RSpec::Matchers.define :have_created_resource do |resource|
  match do |response|
    resource && response.code.created?
  end

  failure_message_for_should do |response|
    if !response.code.created?
      FailureMessages.expected_response_code_for(:created).got(response.code)
    else
      "expected the resource not to be nil"
    end
  end
end
```

**code is easier to  
understand when it  
operates at a  
consistent level of  
abstraction**

```
RSpec::Matchers.define :be_created do
  match do |response|
    response.code.created?
  end

  failure_message_for_should do |response|
    FailureMessages.expected_response_code_for(:created).got(response.code)
  end
end

RSpec::Matchers.define :have_created_resource do |resource|
  match do |response|
    resource && response.code.created?
  end

  failure_message_for_should do |response|
    if !response.code.created?
      FailureMessages.expected_response_code_for(:created).got(response.code)
    else
      "expected the resource not to be nil"
    end
  end
end
end
```

```
RSpec::Matchers.define :be_created do
  match do |response|
    response.code.created?
  end

  failure_message_for_should do |response|
    FailureMessages.expected_response_code_for(:created).got(response.code)
  end
end

RSpec::Matchers.define :have_created_resource do |resource|
  match do |response|
    resource && response.code.created?
  end

  failure_message_for_should do |response|
    if !response.code.created?
      FailureMessages.expected_response_code_for(:created).got(response.code)
    else
      FailureMessages.expected_non_nil_resource
    end
  end
end
```

```
RSpec::Matchers.define :be_created do
  match do |response|
    response.code.created?
  end

  failure_message_for_should do |response|
    FailureMessages.expected_response_code_for(:created).got(response.code)
  end
end

RSpec::Matchers.define :have_created_resource do |resource|
  match do |response|
    resource && response.code.created?
  end

  failure_message_for_should do |response|
    if !response.code.created?
      FailureMessages.expected_response_code_for(:created).got(response.code)
    else
      FailureMessages.expected_non_nil_resource
    end
  end
end
```

# The **DRY** principle

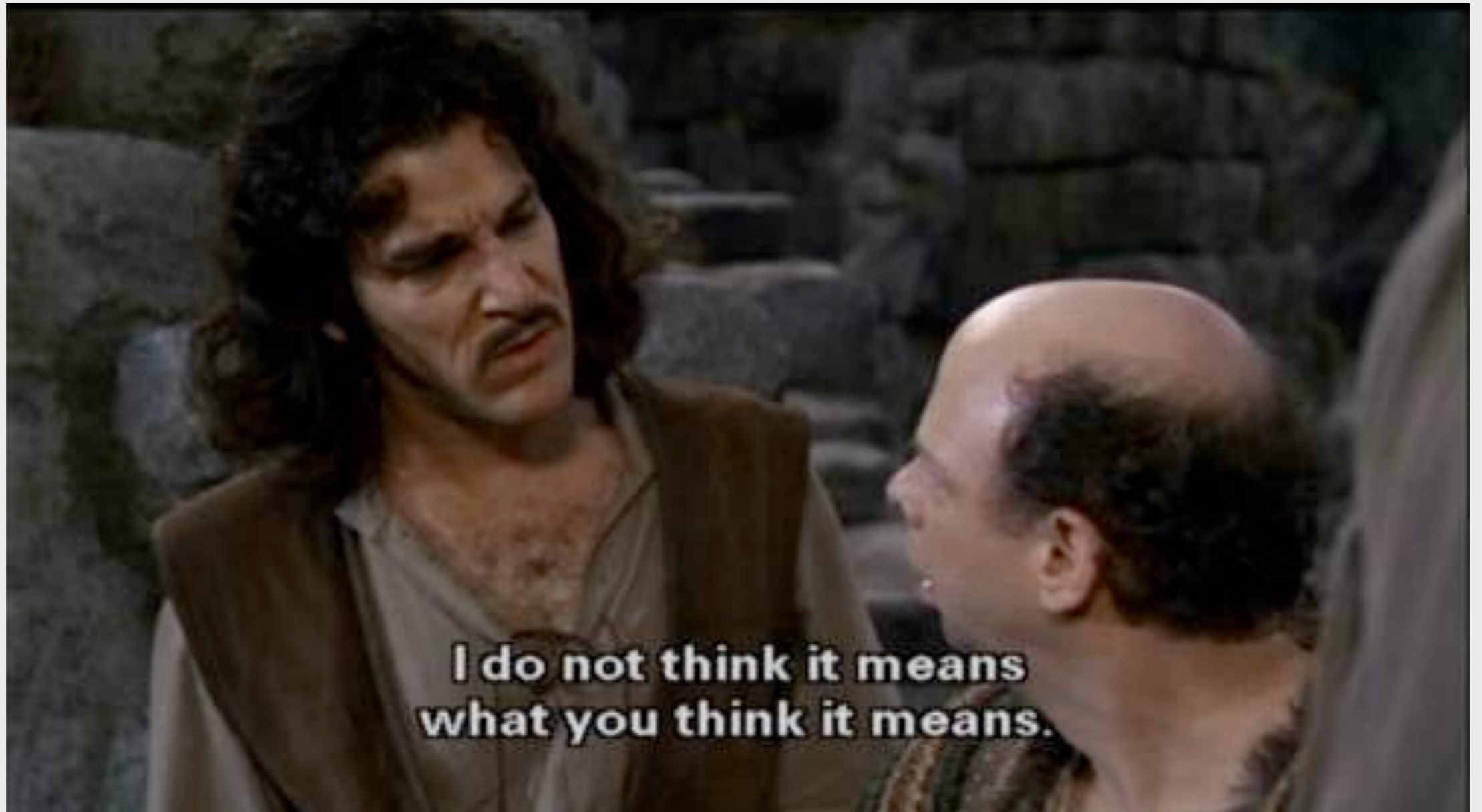
**Don't**

**Repeat**

**Yourself**



# You keep using that word ...



**Every piece of knowledge  
must have a  
single,  
unambiguous,  
authoritative representation  
within a system**

Every **piece of knowledge**

must have a

single,

unambiguous,

authoritative representation

within a system

Every **piece of knowledge**  
must have a  
**single,**  
**unambiguous,**  
**authoritative representation**  
within a system

ALLOCATIONS\_ON\_DATE = "allocations/{sourceType}/on/{year}/{month}"  
CREATE\_ALLOCATION = "allocations/{sourceType}/{sourceId}/{targetType}/{targetId}/{year}/{month}/{percentage}"  
DELETE\_ALLOCATION = "allocations/{sourceType}/{sourceId}/{targetType}/{targetId}/{year}/{month}"



"allocations/{sourceType}/on/{year}/{month}"

"allocations/{sourceType}/{sourceId}/{targetType}/{targetId}/{year}/{month}/{percentage}"

"allocations/{sourceType}/{sourceId}/{targetType}/{targetId}/{year}/{month}"

```
ALLOCATIONS                = "allocations"
ALLOCATIONS_ON_DATE        = ALLOCATIONS + "{sourceType}/on/{year}/{month}"
CREATE_ALLOCATION            = ALLOCATIONS +
    "{sourceType}/{sourceId}/{targetType}/{targetId}/{year}/{month}/{percentage}"
DELETE_ALLOCATION            = ALLOCATIONS +
    "{sourceType}/{sourceId}/{targetType}/{targetId}/{year}/{month}"
```

```
ALLOCATIONS           = "allocations"
SOURCE_TYPE           =("/{sourceType}"
ALLOCATIONS_ON_DATE  = ALLOCATIONS + SOURCE_TYPE + "/on/{year}/{month}"
CREATE_ALLOCATION      = ALLOCATIONS + SOURCE_TYPE +
"/{sourceId}/{targetType}/{targetId}/{year}/{month}/{percentage}"
DELETE_ALLOCATION      = ALLOCATIONS + SOURCE_TYPE +
"/{sourceId}/{targetType}/{targetId}/{year}/{month}"
```

```
ALLOCATIONS           = "allocations"
SOURCE_TYPE           =("/{sourceType}"
SOURCE_ID             =("/{sourceId}"
TARGET_TYPE           =("/{targetType}"
TARGET_ID             =("/{targetId}"
EFFECTIVE_DATE        =("/{year}/{month}"
ALLOCATIONS_ON_DATE   = ALLOCATIONS + SOURCE_TYPE + "/on" + EFFECTIVE_DATE
CREATE_ALLOCATION       = ALLOCATIONS + SOURCE_TYPE + SOURCE_ID +
    TARGET_TYPE + TARGET_ID + EFFECTIVE_DATE +("/{percentage}"
DELETE_ALLOCATION       = ALLOCATIONS + SOURCE_TYPE + SOURCE_ID +
    TARGET_TYPE + TARGET_ID + EFFECTIVE_DATE
```

```

DELIMITERS           = %w[/ { }]
ALLOCATIONS          = "allocations"
EFFECTIVE_DATE       = DELIMITERS[0..1].join + "year" + DELIMITERS[2] +
  DELIMITERS[0..1].join + "month" + DELIMITERS[2]
SOURCE               = "source"
TARGET               = "target"
ID                   = "Id"
TYPE                 = "Type"
SOURCE_TYPE          = DELIMITERS[0..1].join + SOURCE + TYPE + DELIMITERS[2]
SOURCE_ID            = DELIMITERS[0..1].join + SOURCE + ID + DELIMITERS[2]
TARGET_TYPE          = DELIMITERS[0..1].join + TARGET + TYPE + DELIMITERS[2]
TARGET_ID            = DELIMITERS[0..1].join + TARGET + ID + DELIMITERS[2]
ALLOCATION_SOURCE     = SOURCE_TYPE + SOURCE_ID
ALLOCATION_TARGET     = TARGET_TYPE + TARGET_ID
ALLOCATIONS_ON_DATE = ALLOCATIONS + SOURCE_TYPE + DELIMITERS[0] + "on" +
  EFFECTIVE_DATE
CREATE_ALLOCATION     = ALLOCATIONS + ALLOCATION_SOURCE + ALLOCATION_TARGET +
  EFFECTIVE_DATE + DELIMITERS[0..1].join + "percentage" + DELIMITERS[2]
DELETE_ALLOCATION     = ALLOCATIONS +
  ALLOCATION_SOURCE + ALLOCATION_TARGET + EFFECTIVE_DATE

```

```

def token(string)
 ("/{#{string}}")
end
def allocation_path(path)
  "allocations#{path}"
end
EFFECTIVE_DATE      = token("year") + token("month")
SOURCE               = "source"
TARGET               = "target"
ID                   = "Id"
TYPE                 = "Type"
SOURCE_TYPE          = token(SOURCE + TYPE)
SOURCE_ID            = token(SOURCE + ID)
TARGET_TYPE          = token(TARGET + TYPE)
TARGET_ID            = token(TARGET + ID)
ALLOCATION_SOURCE     = SOURCE_TYPE + SOURCE_ID
ALLOCATION_TARGET     = TARGET_TYPE + TARGET_ID
ALLOCATIONS_ON_DATE = allocation_path(SOURCE_TYPE + "/on" + EFFECTIVE_DATE)
CREATE_ALLOCATION     = allocation_path(
  ALLOCATION_SOURCE + ALLOCATION_TARGET + EFFECTIVE_DATE + token("percentage")
)
DELETE_ALLOCATION     = allocation_path(
  ALLOCATION_SOURCE + ALLOCATION_TARGET + EFFECTIVE_DATE
)

```

```
"/allocations/developer/37/project/2010/10/42"
```

```
def token(string)
```

```
  "/#{string}"
```

```
end
```

```
def allocation_path(path)
```

```
  "allocations#{path}"
```

```
end
```

```
EFFECTIVE_DATE      = token("year") + token("month")
```

```
SOURCE              = "source"
```

```
TARGET              = "target"
```

```
ID                  = "Id"
```

```
TYPE                = "Type"
```

```
SOURCE_TYPE         = token(SOURCE + TYPE)
```

```
SOURCE_ID           = token(SOURCE + ID)
```

```
TARGET_TYPE         = token(TARGET + TYPE)
```

```
TARGET_ID           = token(TARGET + ID)
```

```
ALLOCATION_SOURCE    = SOURCE_TYPE + SOURCE_ID
```

```
ALLOCATION_TARGET    = TARGET_TYPE + TARGET_ID
```

```
ALLOCATIONS_ON_DATE = allocation_path(SOURCE_TYPE + "/on" + EFFECTIVE_DATE)
```

```
CREATE_ALLOCATION    = allocation_path(  
  ALLOCATION_SOURCE + ALLOCATION_TARGET + EFFECTIVE_DATE + token("percentage")  
)
```

```
DELETE_ALLOCATION    = allocation_path(  
  ALLOCATION_SOURCE + ALLOCATION_TARGET + EFFECTIVE_DATE
```

```
)
```

"/allocations/developer/37/project/2010/10/42"

"allocations/{sourceType}/on/{year}/{month}"

"allocations/{sourceType}/{sourceId}/{targetType}/{targetId}/{year}/{month}/{percentage}"

"allocations/{sourceType}/{sourceId}/{targetType}/{targetId}/{year}/{month}"

ALLOCATIONS\_ON\_DATE = "allocations/{sourceType}/on/{year}/{month}"

CREATE\_ALLOCATION = "allocations/{sourceType}/{sourceId}/{targetType}/{targetId}/{year}/{month}/{percentage}"

DELETE\_ALLOCATION = "allocations/{sourceType}/{sourceId}/{targetType}/{targetId}/{year}/{month}"

```
ALLOCATIONS_ON_DATE = "allocations/{sourceType}/on/{year}/{month}"
CREATE_ALLOCATION     = "allocations/{sourceType}/{sourceId}/{targetType}/{targetId}/{year}/{month}/{percentage}"
DELETE_ALLOCATION     = "allocations/{sourceType}/{sourceId}/{targetType}/{targetId}/{year}/{month}"
```



```
def token(string)
  "/#{string}"
end
def allocation_path(path)
  "allocations#{path}"
end
EFFECTIVE_DATE = token("year") + token("month")
SOURCE         = "source"
TARGET         = "target"
ID             = "Id"
TYPE           = "Type"
SOURCE_TYPE    = token(SOURCE + TYPE)
SOURCE_ID      = token(SOURCE + ID)
TARGET_TYPE    = token(TARGET + TYPE)
TARGET_ID      = token(TARGET + ID)
ALLOCATION_SOURCE = SOURCE_TYPE + SOURCE_ID
ALLOCATION_TARGET = TARGET_TYPE + TARGET_ID
ALLOCATIONS_ON_DATE = allocation_path(
  SOURCE_TYPE + "/on" + EFFECTIVE_DATE
)
CREATE_ALLOCATION = allocation_path(
  ALLOCATION_SOURCE + ALLOCATION_TARGET + EFFECTIVE_DATE + token("percentage")
)
DELETE_ALLOCATION = allocation_path(
  ALLOCATION_SOURCE + ALLOCATION_TARGET + EFFECTIVE_DATE
)
```

```
ALLOCATIONS_ON_DATE = "allocations/{sourceType}/on/{year}/{month}"
CREATE_ALLOCATION     = "allocations/{sourceType}/{sourceId}/{targetType}/{targetId}/{year}/{month}/{percentage}"
DELETE_ALLOCATION     = "allocations/{sourceType}/{sourceId}/{targetType}/{targetId}/{year}/{month}"
```

**some duplicate  
letters/words**



```
def token(string)
  "/#{string}"
end
def allocation_path(path)
  "allocations#{path}"
end
EFFECTIVE_DATE      = token("year") + token("month")
SOURCE              = "source"
TARGET              = "target"
ID                  = "Id"
TYPE                = "Type"
SOURCE_TYPE         = token(SOURCE + TYPE)
SOURCE_ID           = token(SOURCE + ID)
TARGET_TYPE         = token(TARGET + TYPE)
TARGET_ID           = token(TARGET + ID)
ALLOCATION_SOURCE    = SOURCE_TYPE + SOURCE_ID
ALLOCATION_TARGET    = TARGET_TYPE + TARGET_ID
ALLOCATIONS_ON_DATE = allocation_path(
  SOURCE_TYPE + "/on" + EFFECTIVE_DATE
)
CREATE_ALLOCATION    = allocation_path(
  ALLOCATION_SOURCE + ALLOCATION_TARGET + EFFECTIVE_DATE + token("percentage")
)
DELETE_ALLOCATION    = allocation_path(
  ALLOCATION_SOURCE + ALLOCATION_TARGET + EFFECTIVE_DATE
)
```

```
ALLOCATIONS_ON_DATE = "allocations/{sourceType}/on/{year}/{month}"
CREATE_ALLOCATION     = "allocations/{sourceType}/{sourceId}/{targetType}/{targetId}/{year}/{month}/{percentage}"
DELETE_ALLOCATION     = "allocations/{sourceType}/{sourceId}/{targetType}/{targetId}/{year}/{month}"
```

**some duplicate  
letters/words**



**#!(\*&?\$@**

```
def token(string)
  "/#{string}"
end
def allocation_path(path)
  "allocations#{path}"
end
EFFECTIVE_DATE      = token("year") + token("month")
SOURCE               = "source"
TARGET               = "target"
ID                   = "Id"
TYPE                 = "Type"
SOURCE_TYPE          = token(SOURCE + TYPE)
SOURCE_ID            = token(SOURCE + ID)
TARGET_TYPE          = token(TARGET + TYPE)
TARGET_ID            = token(TARGET + ID)
ALLOCATION_SOURCE     = SOURCE_TYPE + SOURCE_ID
ALLOCATION_TARGET     = TARGET_TYPE + TARGET_ID
ALLOCATIONS_ON_DATE = allocation_path(
  SOURCE_TYPE + "/on" + EFFECTIVE_DATE
)
CREATE_ALLOCATION     = allocation_path(
  ALLOCATION_SOURCE + ALLOCATION_TARGET + EFFECTIVE_DATE + token("percentage")
)
DELETE_ALLOCATION     = allocation_path(
  ALLOCATION_SOURCE + ALLOCATION_TARGET + EFFECTIVE_DATE
)
```

**DRY** does **not** mean  
“don’t type the same  
characters twice”

**DRY** is about  
duplicated **concepts**

**DRY is about**  
**isolation of change**

**How do I know if this  
particular duplication  
is problematic?**

# Purpose/Intention

**Risk**

Have to **change** in  
**more than one** place?

**Might **forget** to  
change in other place!**

**Duplication** is not  
always obvious

**Same name,  
different meaning**

# browser

Percentages:

50.0

30.0

20.0

# report

percentage
------------

0.5
-----

0.3
-----

0.2
-----

percentage = 50.0 # in browser

percentage = 0.5 # in reports

```
class Target < ActiveRecord::Base  
end
```

```
Target.create!(:percentage => percentage / 100.0)
```

```
<%= target.percentage * 100.0 %>
```

percentage = 50.0 # in browser

ratio = 0.5 # in reports/database

```
class Target < ActiveRecord::Base
  def percentage=(new_value)
    self.ratio = new_value/100.0 if new_value
  end

  def percentage
    ratio * 100.0
  end
end
```

```
Target.create!(:percentage => percentage)
```

```
<%= target.percentage %>
```

```
class Target < ActiveRecord::Base
  def percentage=(new_value)
    self.ratio = new_value/100.0 if new_value
  end

  def percentage
    ratio * 100.0
  end
end
```

```
Target.create!(:percentage => percentage)
```

```
<%= target.percentage %>
```

# browser

Percentages:

50.0

30.0

20.0

# report

	<b>ratio</b>
	0.5
	0.3
	0.2

**Same meaning,  
different API**

```
class Line
  attr_accessor :start_point, :end_point, :length

  def initialize(start_point, end_point, length)
    @start_point, @end_point, @length = start_point, end_point, length
  end
end
```

```
class Line
  attr_accessor :start_point, :end_point, :length

  def initialize(start_point, end_point, length)
    @start_point, @end_point, @length = start_point, end_point, length
  end
end
```

```
line = Line.new(Point.new(0,0), Point.new(37,42), 55.97)
puts line.length
# => 55.97
```

```
class Line
  attr_reader :start_point, :end_point

  def initialize(start_point, end_point)
    @start_point, @end_point = start_point, end_point
  end

  def length
    Math.hypot(end_point.x - start_point.x, end_point.y - start_point.y)
  end
end
```

# Structural/Functional

```
class Person
  def primary_zip_code
    primary = addresses.detect { |a| a.primary? }
    primary ||= addresses.first
    primary.zip_code if primary
  end
end
```

```
class Person
  def primary_zip_code
    primary = addresses.detect { |a| a.primary? }
    primary ||= addresses.first
    primary.zip_code if primary
  end

  def primary_phone_number
  end
end
```

```
class Person
  def primary_zip_code
    primary = addresses.detect {|a| a.primary?}
    primary ||= addresses.first
    primary.zip_code if primary
  end

  def primary_phone_number
    primary = addresses.detect {|a| a.primary?}
    primary ||= addresses.first
    primary.phone_number if primary
  end
end
```

```
class Person
  def primary_zip_code
    main = addresses.detect {|a| a.primary?}
    main ||= addresses.first
    main if main
  end

  def primary_phone_number
    primary = addresses.detect {|a| a.primary?}
    primary ||= addresses.first
    primary.phone_number if primary
  end
end
```

```
class Person
  def primary_zip_code
    main = addresses.detect {|a| a.primary?} || addresses.first
    main.zip_code if main
  end

  def primary_phone_number
    primary = addresses.detect {|a| a.primary?}
    primary ||= addresses.first
    primary.phone_number if primary
  end
end
```

# Refactoring: Extract Method

```
class Person
  def primary_zip_code
    primary_address.zip_code
  end

  def primary_phone_number
    primary_address.phone_number
  end

  def primary_address
    addresses.detect {|a| a.primary?} ||
      addresses.first ||
      Address.new
  end
end
```

# Code smell: Feature Envy

```
class AddressCollection < Array
  def primary
    detect {|a| a.primary?} || first || Address.new
  end
end
```

```
class Person
  attr_reader :addresses

  def initialize
    @addresses = AddressCollection.new
  end

  def primary_zip_code
    primary_address.zip_code
  end

  def primary_phone_number
    primary_address.phone_number
  end

  def primary_address
    addresses.primary
  end
end
```

# Specs

```
describe Person do
  describe "#distance_to_event" do
    context "with one address" do
      context "with no zipcode" do
        it "returns nil"
      end
    end
  end
end
end
```

```
describe Person do
  describe "#distance_to_event" do
    context "with one address" do
      context "with no zipcode" do
        it "returns nil" do
          person = Person.new
          person.addresses << Address.new(:zipcode => nil)
          @person.distance_to_event(Event.new).should be(nil)
        end
      end
    end
  end
end
end
end
```

```
describe Person do
  describe "#distance_to_event" do
    context "with one address" do
      context "with no zipcode" do
        before do
          @person = Person.new.tap do |person|
            person.addresses << Address.new(:zipcode => nil)
          end
        end
        it "returns nil" do
          @person.distance_to_event(Event.new).should be(nil)
        end
      end
    end
  end
end
end
end
```

```
describe Person do
  describe "#distance_to_event" do
    context "with one address" do
      before do
        @person = Person.new.tap do |person|
          person.addresses << Address.new
        end
      end
    end
    context "with no zipcode" do
      before do
        @person.addresses.first.zipcode = nil
      end
      it "returns nil" do
        @person.distance_to_event(Event.new).should be(nil)
      end
    end
  end
end
end
end
```

```
describe Person do
  before do
    @person = Person.new
  end
  describe "#distance_to_event" do
    context "with one address" do
      before do
        @person.addresses << Address.new
      end
      context "with no zipcode" do
        before do
          @person.addresses.first.zipcode = nil
        end
        it "returns nil" do
          @person.distance_to_event(Event.new).should be(nil)
        end
      end
    end
  end
end
end
end
```

**Less duplication can  
mean more indirection**

**this can make it  
harder to reason  
about code**

```
describe Person do
  describe "when first created" do
    it "has no friends" do
      Person.new.should have(:no).friends
    end
  end
end
```

```
Person
  when first created
    has no friends
```

```
describe Person do
  describe "when first created" do
    it { should have(:no).friends }
  end
end
```

```
Person
  when first created
    should have 0 friends
```

```
describe Calculator do
  describe "#add" do
    it "returns the sum of the operands provided" do
      calculator = Calculator.new
      calculator.add(2,3).should eq(5)
    end
  end
end
```

```
Calculator
  #add
    returns the sum of the operands provided
```

```
describe Calculator do
  subject { Calculator.new.add(2,3) }
  it { should eq(5) }
end
```

```
Calculator
  should eq 5
```

# Metaprogramming

```
class Person
  def initialize(first_name, last_name, full_name)
    @first_name = first_name
    @last_name = last_name
    @full_name = full_name
  end

  def first_name
    @first_name
  end

  def last_name
    @last_name
  end

  def full_name
    @full_name
  end
end
```

```
class Person
  def initialize(first_name, last_name, full_name)
    @first_name = first_name
    @last_name = last_name
    @full_name = full_name
  end

  %w[first last full].each do |prefix|
    define_method "#{prefix}_name" do
      instance_variable_get("@#{prefix}_name")
    end
  end
end
```

**Metaprogramming** is  
**helpful** when you can't  
know the **runtime**  
**conditions**

**Metaprogramming**  
**can also make it**  
**harder to reason**  
**about code**

# Rails Controllers

```
class WidgetsController < ApplicationController
  def index
    @widgets = Widget.all
  end

  def show
    @widget = Widget.find(params[:id])
  end

  def new
    @widget = Widget.new
  end

  def edit
    @widget = Widget.find(params[:id])
  end

  def create
    @widget = Widget.new(params[:widget])
    if @widget.save
      redirect_to(@widget, :notice => 'Widget was successfully created.')
    else
      render :action => "new"
    end
  end
end
```

```
def update
  @widget = Widget.find(params[:id])

  if @widget.update_attributes(params[:widget])
    redirect_to(@widget, :notice => 'Widget was successfully updated.')
  else
    render :action => "edit"
  end
end

def destroy
  @widget = Widget.find(params[:id])
  @widget.destroy
  redirect_to(widgets_url)
end
end
```

# Extract method

```
class WidgetsController < ApplicationController
  def show
    assign_widget
  end

  def edit
    assign_widget
  end

  def update
    assign_widget
    if @widget.update_attributes(params[:widget])
      redirect_to(@widget, :notice => 'Widget was successfully updated.')
    else
      render :action => "edit"
    end
  end

  def destroy
    assign_widget
    @widget.destroy
    redirect_to(widgets_url)
  end

private

  def assign_widget
    @widget = Widget.find(params[:id])
  end
end
```

**Before filters!**

```
class WidgetsController < ApplicationController
  before_filter :assign_widget, :only => [:show, :edit, :update, :destroy]

  def show; end

  def edit; end

  def update
    if @widget.update_attributes(params[:widget])
      redirect_to(@widget, :notice => 'Widget was successfully updated.')
    else
      render :action => "edit"
    end
  end

  def destroy
    @widget.destroy
    redirect_to(widgets_url)
  end

private

  def assign_widget
    @widget = Widget.find(params[:id])
  end

end
```

```
def index
  @widgets = Widget.all
end

def show; end

def new
  @widget = Widget.new
end

def edit; end

def create
  @widget = Widget.new(params[:widget])

  if @widget.save
    redirect_to(@widget, :notice => 'Widget was successfully created.')
  else
    render :action => "new"
  end
end

def update
  if @widget.update_attributes(params[:widget])
    redirect_to(@widget, :notice => 'Widget was successfully updated.')
  else
    render :action => "edit"
  end
end

def destroy
  @widget.destroy
  redirect_to(widgets_url)
end
```

```
def index
  @widgets = Widget.all
end

def show; end

def new
  @widget = Widget.new
end

def edit; end

def create
  @widget = Widget.new(params[:widget])

  if @widget.save
    redirect_to(@widget, :notice => 'Widget was successfully created.')
  else
    render :action => "new"
  end
end

def update
  if @widget.update_attributes(params[:widget])
    redirect_to(@widget, :notice => 'Widget was successfully updated.')
  else
    render :action => "edit"
  end
end

def destroy
  @widget.destroy
  redirect_to(widgets_url)
end
```

```
def new
  @widget = Widget.new
end
```

```
def create
  @widget = Widget.new(params[:widget])
  if @widget.save
    redirect_to(@widget, :notice => 'Widget was successfully created.')
  else
    render :action => "new"
  end
end
```

```
private
```

```
def assign_widget
  @widget = Widget.find(params[:id])
end
```

```
def new

end

def create

  if @widget.save
    redirect_to(@widget, :notice => 'Widget was successfully created.')
  else
    render :action => "new"
  end
end
```

private

```
def assign_widget
  @widget = params[:id] ? Widget.find(params[:id]) :
                    Widget.new(params[:widget])
end
```

```
class WidgetsController < ApplicationController
  before_filter :assign_widget, :only => [:show, :new, :edit, :create:, :update, :destroy]

  def show; end
  def new; end
  def edit; end

  def create
    if @widget.save
      redirect_to(@widget, :notice => 'Widget was successfully created.')
    else
      render :action => "new"
    end
  end

  def update
    if @widget.update_attributes(params[:widget])
      redirect_to(@widget, :notice => 'Widget was successfully updated.')
    else
      render :action => "edit"
    end
  end

  def destroy
    @widget.destroy
    redirect_to(widgets_url)
  end

private

  def assign_widget
    @widget = params[:id] ? Widget.find(params[:id]) : Widget.new(params[:widget])
  end

end
```

```
class WidgetsController < ApplicationController
  before_filter :assign_widget, :except => :index

  def show; end
  def new; end
  def edit; end

  def create
    if @widget.save
      redirect_to(@widget, :notice => 'Widget was successfully created.')
    else
      render :action => "new"
    end
  end

  def update
    if @widget.update_attributes(params[:widget])
      redirect_to(@widget, :notice => 'Widget was successfully updated.')
    else
      render :action => "edit"
    end
  end

  def destroy
    @widget.destroy
    redirect_to(widgets_url)
  end

private

  def assign_widget
    @widget = params[:id] ? Widget.find(params[:id]) : Widget.new(params[:widget])
  end

end
```

```
class WidgetsController < ApplicationController
  before_filter :assign_widget, :except => index

  def index
    @widgets = Widget.all
  end

  def show; end
  def new; end
  def edit; end

  def create
    if @widget.save
      redirect_to(@widget, :notice => 'Widget was successfully created.')
    else
      render :action => "new"
    end
  end

  def update
    if @widget.update_attributes(params[:widget])
      redirect_to(@widget, :notice => 'Widget was successfully updated.')
    else
      render :action => "edit"
    end
  end

  def destroy
    @widget.destroy
    redirect_to(widgets_url)
  end

private

  def assign_widget
    @widget = params[:id] ? Widget.find(params[:id]) : Widget.new(params[:widget])
  end

end
```

```
class WidgetsController < ApplicationController
  before_filter :assign_widget, :except => :index

  def index
    @widgets = Widget.all
  end

  def show; end
  def new; end
  def edit; end

  def create
    if @widget.save
      redirect_to(@widget, :notice => 'Widget was successfully created.')
    else
      render :action => "new"
    end
  end

  def update
    if @widget.update_attributes(params[:widget])
      redirect_to(@widget, :notice => 'Widget was successfully updated.')
    else
      render :action => "edit"
    end
  end

  def destroy
    @widget.destroy
    redirect_to(widgets_url)
  end

private

  def assign_widget
    @widget = params[:id] ? Widget.find(params[:id]) : Widget.new(params[:widget])
  end

end
```

```
class WidgetsController < ApplicationController
  before_filter :assign_widget, :except => :index

  def index
    @widgets = Widget.all
  end

  def show; end
  def new; end
  def edit; end

  def create
    if @widget.save
      redirect_to(@widget, :notice => 'Widget was successfully created.')
    else
      render :action => "new"
    end
  end

  def update
    if @widget.update_attributes(params[:widget])
      redirect_to(@widget, :notice => 'Widget was successfully updated.')
    else
      render :action => "edit"
    end
  end

  def destroy
    @widget.destroy
    redirect_to(widgets_url)
  end

private

  def assign_widget
    @widget = params[:id] ? Widget.find(params[:id]) : Widget.new(params[:widget])
  end

end
```

```
class WidgetsController < ApplicationController
  before_filter :assign_widget, :except => :index

  def index
    @widgets = Widget.all
  end

  def show; end
  def new; end
  def edit; end

  def create
    if @widget.save
      redirect_to(@widget, :notice => 'Widget was successfully created.')
    else
      render :action => "new"
    end
  end

  def update
    if @widget.update_attributes(params[:widget])
      redirect_to(@widget, :notice => 'Widget was successfully updated.')
    else
      render :action => "edit"
    end
  end

  def destroy
    @widget.destroy
    redirect_to(widgets_url)
  end

  private

  def assign_widget
    @widget = params[:id] ? Widget.find(params[:id]) : Widget.new(params[:widget])
  end

end
```

**Filters are great  
for  
filtering!**

**Filters are great for  
orthogonal or cross-  
cutting concerns,  
like logging and  
authorization**

**Filters** can also make  
it **harder to reason**  
**about code**

# In Summary

**Many** problems in  
software **can** be  
traced back to  
**duplication**, but ...

... **not all** duplication  
is inherently **evil**

**Duplication** comes in  
many forms

**Duplication** appears  
for **several reasons**

Every **piece of knowledge**  
must have a  
**single,**  
**unambiguous,**  
**authoritative representation**  
within a system

**This is not really about  
duplicated **strings** of  
characters**

**DRY is about  
duplicated concepts**

**every time you**  
**reduce duplication**  
**you**  
**increase coupling**  
**by introducing**  
**new dependencies**

**depend on things  
which are  
less likely to change**

# learn about code smells

**Duplicated Code**

**Feature Envy**

**Inappropriate Intimacy**

**Large Class**

**Long Method**

**learn more principles!**

# SOLID Principles

- Single Responsibility
- Open/Closed
- Liskov Substitution
- Interface Segregation
- Dependency Inversion

**OO in one sentence**

**keep it shy, DRY, and tell  
the other guy!**

# Questions?

@dchelimsky

<http://blog.davidchelimsky.net/>

<http://relishapp.com/rspec/>

<http://www.drwtrading.com/>

<http://www.pragprog.com/the-pragmatic-programmer>

<http://pragprog.com/titles/achbd/the-rspec-book>

<http://www.amazon.com/>

[Refactoring-Improving-Design-Existing-Code/  
dp/0201485672](http://www.amazon.com/Refactoring-Improving-Design-Existing-Code/dp/0201485672)

<http://www.amazon.com/>

[Software-Development-Principles-Patterns-Practices/  
dp/0135974445](http://www.amazon.com/Software-Development-Principles-Patterns-Practices/dp/0135974445)