SIMPLIFYING CODE

MONSTER TO ELEGANT IN N<5 STEPS

Tute Costa - @tutec

WE'LL LEARN HOW TO TRANSFORM THIS

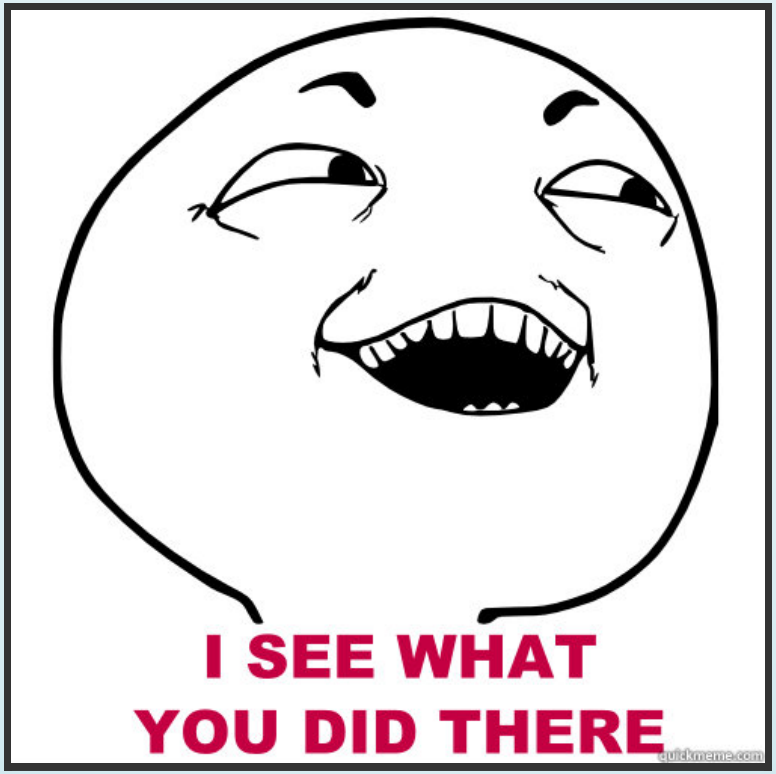
```
// fax
if (!empty($this->post['fax'])) {
  $site->fax = $this->post['fax'][0];
}

// Facebook & Twitter
if ($this->post['social_media']) {
  if (!empty($this->post['facebook'])) {
    $facebookURL = $this->post['facebook'];
    if (strpos($this->post['facebook'], 'http://')
      $facebookURL = "http://" . $this->post['faceb
    }
    $site->facebook_url = $facebookURL;
  }

  if (!empty($this->post['twitter'])) {
```

Into THIS

```
current_user
  .public_profiles
  .update(params)
```



ABOUT YOU

ABOUT TUTE

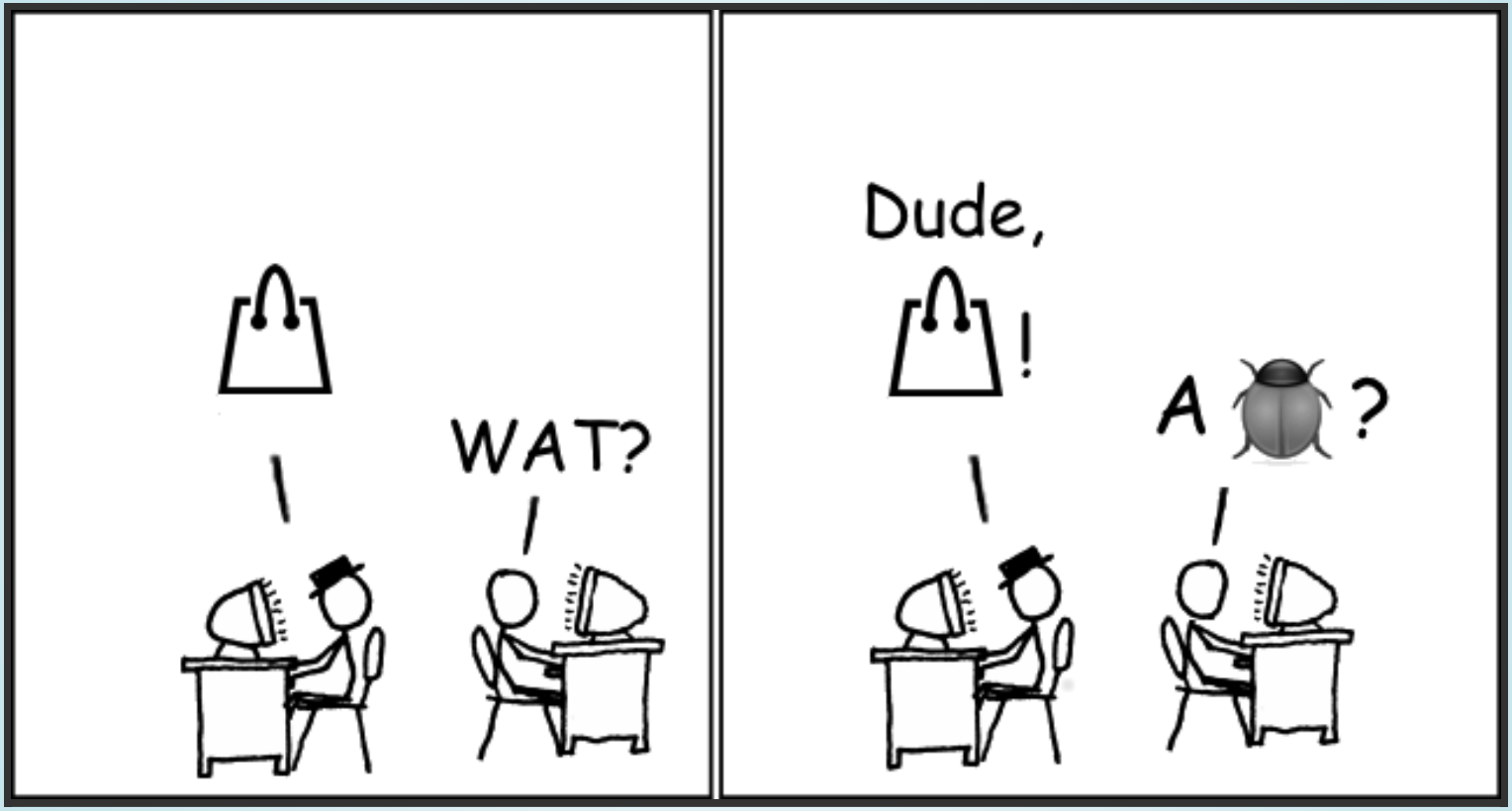
ABOUT TUTE


eugenio costa
Web
Imágenes
Vídeos
Más
Herramientas de búsqueda
Imágenes de eugenio costa - Informar sobre las imágenes
SS Eugenio C - Wikipedia, the free encyclopedia

ABOUT TUTE

ABOUT TUTE

ABOUT TUTE



Alright if we'll go all technical I'll say "bug".

2011/2012: CHEFSURFING

We wrote so much code.

Every 2 months complexity would bite us.

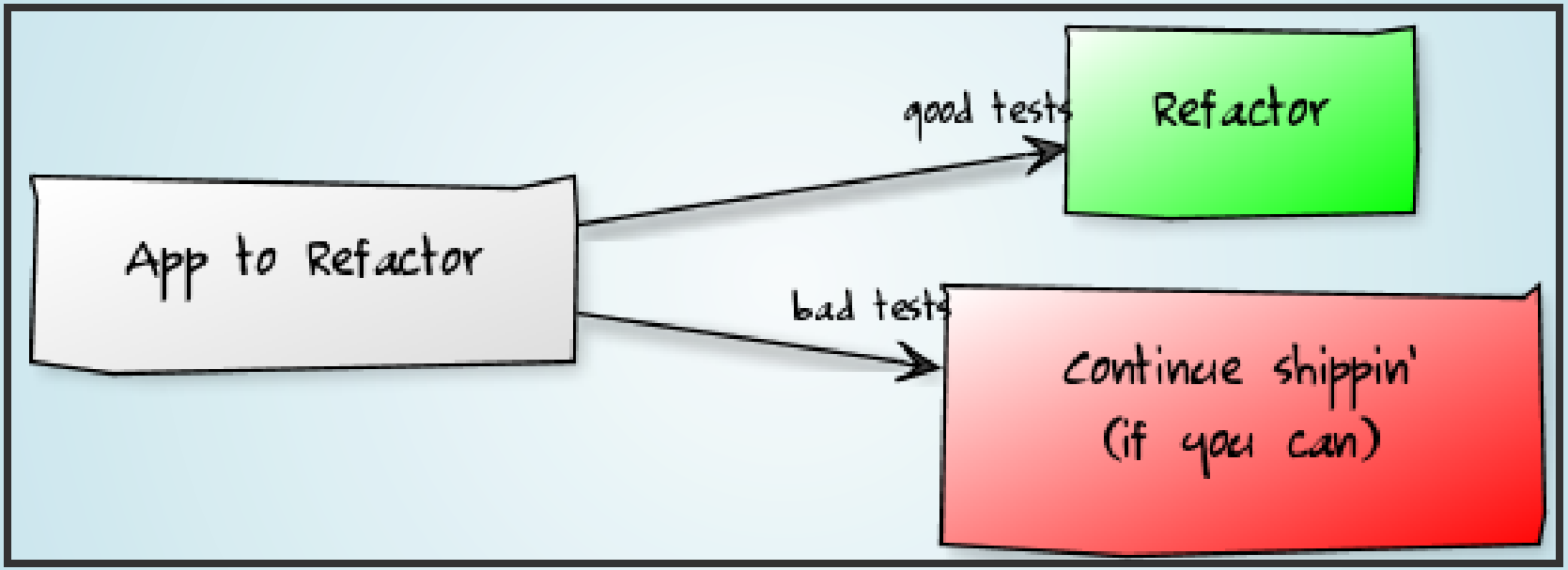
Stop-the-world. Test && Refactor.

Not predictable. Not sustainable.

2013: GENERAL ASSEMBLY

- Rails app in BAD shape. But well tested.
- I was told *“refactor allthethings!“*
- Improved productivity week after week
- I want to repeat (and improve!) this story.

PRECONDITION: THERE'S TESTS



REFACTORING PATTERNS

1. Intention Revealing Method
2. Special Case Object
3. Replace Method with Method Object
4. Service Object

INTENTION REVEALING METHOD

Why it's called is more important than how/what it does.

```
# Remove duplicates?
if hash[row[1]][date] != row[0]
  # ...
```

```
if remove_duplicates?(row, date)
  # ...

def remove_duplicates?(data, date)
```

INTENTION REVEALING METHOD

1. Add comments if code needs it
2. Transform comments into methods

   Transform them syntactically, then create the method.

3. Comments are now code.

   Code describes itself.

INTENTION REVEALING METHOD

git clone
http://github.com/tute/
refactoring-workshop

INTENTION REVEALING METHOD

<5 lines per method

No problem!

INTENTION REVEALING METHOD

It's arguably the easiest pattern.

But the hardest as well.

INTENTION REVEALING METHOD RESOURCES

- http://ntcoding.blogspot.com.ar/2011/05/cle code-tricks-intention-revealing.html
- http://www.codinghorror.com/blog/2008/07/ without-comments.html
- http://c2.com/cgi/wiki?ExtractMethod

2. SPECIAL CASE OBJECTS

No more `if`s or `try`s.

2. SPECIAL CASE OBJECTS

`nil` is a troublemaker.

Source of hard to trace exceptions:

```
Undefined method `email' for
nil:NilClass
```

```
session[:current_user]  # => nil
if (false) then 1 end   # => nil
empty_method()          # => nil
```

2. SPECIAL CASE OBJECTS

A symbol is better than `nil`:

```
def current_user
  User.find_by_id(params[:id]) ||
  :guest_user
end
```

```
current_user.email
```

```
undefined method `email' for
    :guest_user:Symbol
```

2. SPECIAL CASE OBJECTS

If there may be `nil` we need to enclose it with an `if`:

```
if current_user
  "Ohai, #{current_user.email}!"
else
  'Ohai, guest!'
end
```

2. SPECIAL CASE OBJECTS

Instead of `nil`, return a new object

```
class NullUser
  def email
    'guest'
  end
end
```

```
def current_user
  User.find(session[:user_id]) ||
  NullUser.new
end
```

```
"Ohai, #{current_user.email}!"
```

2. SPECIAL CASE OBJECTS

RESOURCES

- RubyTapas #112
- http://devblog.avdi.org/2011/05/30/null-obje
  falsiness/
- http://martinfowler.com/eaaCatalog/specialC
- http://www.cs.oberlin.edu/~jwalker/nullObjP
- http://nickknowlson.com/blog/2013/04/16/w
  maybe-is-better-than-null/

3. REPLACE METHOD WITH METHOD OBJECT

How to refactor a GIGANTIC method without getting lost in the cold night.

```
def row_per_day_format(file_name)
  file = File.open file_name, 'r:ISO-8859-1'
  # hash[NivelConsistencia][date] = [[value, status]]
  hash = { '1' => {}, '2' => {} }
  dates = []
  str = ''

  CSV.parse(file, col_sep: ';').each do |row|
    next if row.empty?
    next if row[0] =~ /^\/\//
    date = Date.parse(row[2])
    (13..43).each do |i|
      measurement_date = date + (i-13)

      # If NumDiasDeChuva is empty it means no data
      value  = row[7].nil? ? -99.9 : row[i]
      status = row[i + 31]
      hash_value = [value, status]

      dates << measurement_date
```

3. REPLACE METHOD WITH METHOD OBJECT

1/5. Look carefully at the code. Get scared.

```
def row_per_day_format(file_name)
  file = File.open file_name, 'r:ISO-8859-1'
  # hash[NivelConsistencia][date] = [[value, status]]
  hash = { '1' => {}, '2' => {} }
  dates = []
  str = ''

  CSV.parse(file, col_sep: ';').each do |row|
    next if row.empty?
    next if row[0] =~ /^\/\//
    date = Date.parse(row[2])
    (13..43).each do |i|
      measurement_date = date + (i-13)

      # If NumDiasDeChuva is empty it means no data
      value  = row[7].nil? ? -99.9 : row[i]
      status = row[i + 31]
      hash_value = [value, status]

      dates << measurement_date
```

3. REPLACE METHOD WITH METHOD OBJECT

1/4. Create a class with same initialization arguments as BIG method

```
class FormatAtoB
  def initialize(file_name)
    @file_name = file_name
  end
end
```

3. REPLACE METHOD WITH METHOD OBJECT

2/4. Copy & Paste the method's body in the new class, with no arguments

```
class FormatAtoB
  def initialize(file_name)
    @file_name = file_name
  end

  def row_per_day_format
    file = File.open file_name, 'r:ISO-8859-1'
    # hash[NivelConsistencia][date] = [[value, stat
    hash = { '1' => {}, '2' => {} }
    dates = []
    str = ''

    CSV.parse(file, col_sep: ';').each do |row|
      next if row.empty?
      next if row[0] =~ /^\/\//
      date = Date.parse(row[2])
```

3. REPLACE METHOD WITH METHOD OBJECT

3/4. Replace original method with a call to the new class

```
def row_per_day_format(file_name)
  FormatAtoB.new(file_name).row_per_day_format
end
```

3. REPLACE METHOD WITH METHOD OBJECT

4/4. Apply "Intention Revealing Method" to the class. Voilà.

```
class FormatAtoB
  def initialize(file_name)
    @file_name = file_name
  end

  def row_per_day_format
    load_file_a
    format_data
  end

  private

  def load_file_a
    # [...]
  end
end
```

3. REPLACE METHOD WITH METHOD OBJECT

RESOURCES

- http://www.refactoring.com/catalog/replaceM
- http://confreaks.com/videos/1071-cascadiaru refactoring

4. SERVICE OBJECTS

Decoupling different concerns
from chubby classes

4. SERVICE OBJECTS

IF WE ADD NEW FUNCTIONALITY TO AN OBJECT:

- It couples to a new dependency
- It loses cohesion
- Testing gets harder and slower
- We can't describe the model without connectors "and"/"or". SRP.

4. SERVICE OBJECTS

If it's a domain concept, it's an Object.

If it's only an algorithm (no state) we call it a Service.

4. SERVICE OBJECTS

RESOURCES

- http://blog.codeclimate.com/blog/2012/10/17-ways-to-decompose-fat-activerecord-models/#service-objects
- http://railscasts.com/episodes/398-service-objects

NEXT STEPS

Send Pull Requests to GitHub.

It's a gold mine:

sferik/rails_admin

Code Climate

TracksApp/tracks

Code Climate

NEXT STEPS: "4 RULES"

1. Classes of at most 100 lines of code
2. Methods of at most 5 lines of code
3. A method accepts at most 4 arguments
4. A controller instantiates only one object

WHY REFACTORING

- Not only about aesthetics, but shared understanding, performance.
- We work with the tools with which we work. We are users and creators.
- If I have a bias I choose "over-engineering". "Under-engineering" is risky, expensive, and over-crowded.

EL FIN

@tutec - github.com/tute