@ddemaree

I like pie.

typekit

The web's best fonts.

Adobe

We make Photoshop™.

REFACTORING

vs

PREFACTORING

Don't
Repeat
Yourself

vs

Too much
abstraction

GitHub, Inc.

Reader

github

Search...

Explore Gist Blog Help

ddemaree

PRIVATE

typekit / typekit

Pull Request | Unwatch | Unstar 3 | Fork 0

Code | Network | Pull Requests 10 | Issues 41 | Wiki | Graphs

Closed

1,308 #1308

Discussion | Commits 74 | Diff 89

Localize Email Templates

Edit

No one is assigned

No milestone

This branch is just a localization branch.

One of the initial ideas was to replace PostOffice with ActionMailer, but it seems like this task is orthogonal to the I18n project and while it has some benefits, it's more of a scope creep than a must to work on now.

4 participants

a month ago

	POC to localize email templates	eaa7f59
	Placeholder for MailLocale::User to be used for localized emails	b1affa2
	module rename	13ace82

Closed

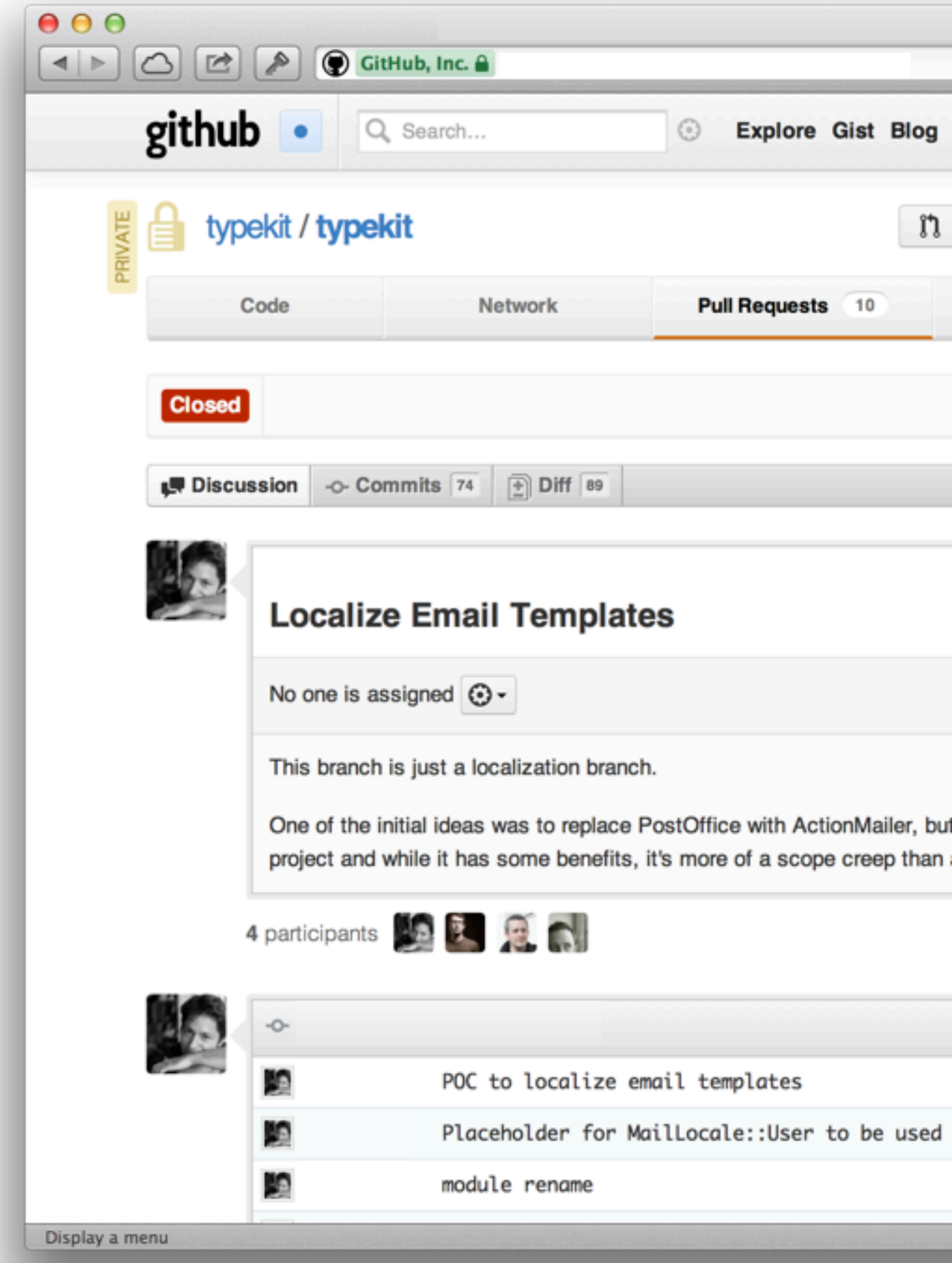
+ 1,006 additions

- 302 deletions

Display a menu

THE PROBLEM

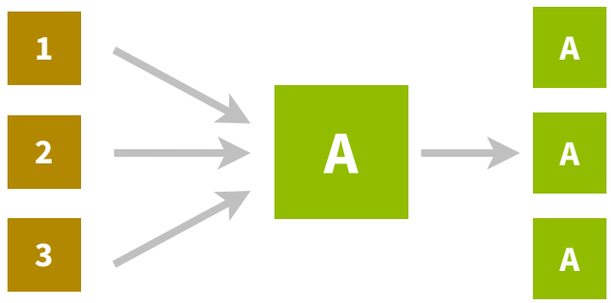
It's really easy to convince yourself that today's code will be more complex in the future, and design for that.



When is it time to refactor?

When is it ***not*** time to refactor?

Rules of thumb



The Rule of Three

“Code can be copied once, but that when the same code is used three times, it should be extracted into a new procedure.”

http://en.wikipedia.org/wiki/Rule_of_three_(computer_programming)

```ruby
def foo
  # Does some stuff ...
  if @user.status == :active
    @user.status = :inactive
    @user.save!
  end
end
```

```ruby
def foo
  # Does some stuff ...
  if @user.status == :active
    @user.status = :inactive
    @user.save!
  end
end
```


```ruby
def bar
  # Does some other stuff ...
  if @user.status == :active
    @user.status = :inactive
    @user.save!
  end
end
```

```ruby
def foo
  # Does some stuff ...
  if @user.status == :active
    @user.status = :inactive
    @user.save!
  end
end

def bar
  # Does some other stuff ...
  if @user.status == :active
    @user.status = :inactive
    @user.save!
  end
end
```


```ruby
def baz
  # Does yet other stuff ...
  if @user.status == :active
    @user.status = :inactive
    @user.save!
  end
end
```

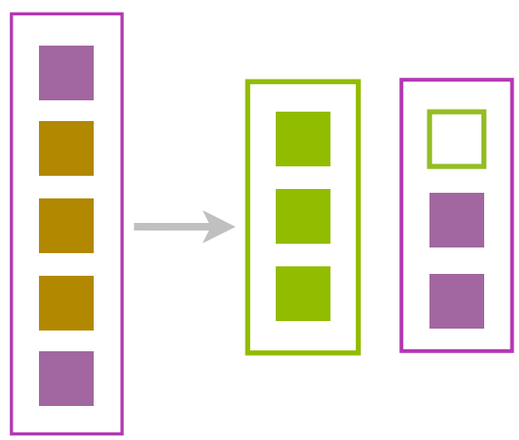
```ruby
def deactivate_user(user)
  if user.status == :active
    user.status = :inactive
    user.save!
  end
end

def foo
  # Does some stuff ...
  deactivate_user(@user)
end

def bar
  # Does some other stuff ...
  deactivate_user(@user)
end

def baz
  # Does yet other stuff ...
  deactivate_user(@user)
end
```

```ruby
def deactivate_user(user)
  if user.status == :active
    user.status = :inactive
    user.save!
  end
end

def foo
  # Does some stuff ...
  deactivate_user(@user)
end

def bar
  # Does some other stuff ...
  deactivate_user(@user)
end

def baz
  # Does yet other stuff ...
  deactivate_user(@user)
end
```

This is better code

BUT

Resist the temptation to do this until you really have repeated yourself three times



Three of a Kind

Three or more related methods/symbols
can be meaningfully grouped together

```ruby
class User < ActiveRecord::Base

  def adobe_profile
    AdobeId::Profile.new(self.adobe_profile_id)
  end

  def connected_to_adobe_profile?
    !self.adobe_profile_id.nil?
  end

end
```

```ruby
class User < ActiveRecord::Base

  def adobe_profile
    AdobeId::Profile.new(self.adobe_profile_id)
  end

  def connected_to_adobe_profile?
    !self.adobe_profile_id.nil?
  end

  def connect_to_adobe_profile(profile_obj)
    self.adobe_profile_id = profile_obj.id
    self.email_unique = false
    save
  end



end
```

```ruby
class User < ActiveRecord::Base

  def adobe_profile
    AdobeId::Profile.new(self.adobe_profile_id)
  end

  def connected_to_adobe_profile?
    !self.adobe_profile_id.nil?
  end

  def connect_to_adobe_profile(profile_obj)
    self.adobe_profile_id = profile_obj.id
    self.email_unique = false
    save
  end

  def disconnect_adobe_profile
    self.adobe_profile_id = nil
    self.email_unique = true
    save
  end

end
```

```ruby
module UserAdobeProfile

  def adobe_profile
    AdobeId::Profile.new(self.adobe_profile_id)
  end

  def connected_to_adobe_profile?
    !self.adobe_profile_id.nil?
  end

  # Anything else pertaining to users'
  # Adobe profiles …

end

class User < ActiveRecord::Base
  include UserAdobeProfile
end
```

No Assumptions

You don’t know what you know
until you know it.

JEFF ATWOOD:

> **I believe writing a truly reusable class is an order of magnitude harder than writing a single use class.** Sometimes the right thing to do is resist the urge to write "general purpose" solutions.

http://www.codinghorror.com/blog/2004/09/the-delusion-of-reuse.html

It's awesome.

ddemaree@adobe.com

@ddemaree

log.demaree.me

WindyCityRails

typekit

Adobe

It will not surprise you to learn that

WE ARE HIRING

`http://bit.ly/typekitrailsjob`

or just come talk to me

WindyCityRails

typekit

Adobe