Maintainable Rails View

文字

xdite@RubyConf China 2013

http://bit.ly/rails-view-bp

self introduction

Rails Developer since 2007

Speaker of RubyTaiwan Conf 2010,2011
Speaker of RubyChina Conf 2012, 2013
Speaker of Reddot RubyConf Singapore 2013
Grand Prize of Facebook World Hack (12 countries) 2012

Entrepreneur

Rocodev, top Rails consultancy in Taiwan
Logdown, leading Markdown Blogging Platform

rubyconfchina2013

- 20 USD

Rails View Best Practices

Agenda

- Concept: What's good view?
- Helper best practices
- Partial best practices
- Beyond Helper & Partial
- Object-oriented View

Warning

You should have tests before modifying codes

Best Practice Lesson 0

Concepts

Why best practices?

- Large & complicated application
- Team & different coding style

Your View

- Complex UI with logic　　UI 與 邏輯糾纏
- Too long to maintain　　冗長難以維護
- Low performance　　效能低落
- Security issues　　容易產生安全性問題

We need good code

What's Good code?

- Readability 易讀，容易了解
- Flexibility 彈性，容易擴充
- Effective 效率，撰碼快速
- Maintainability 維護性，容易找到問題
- Consistency 一致性，循慣例無須死背
- Testability 可測性，元件獨立容易測試

Good => Better => Best

So, What we can do?

Best Practice Lesson #1

Move logic to helper

Move logic to helper

Before

```
<% if current_user && current_user == post.user %>
  <%= link_to("Edit", edit_post_path(post))%>
<% end %>
```

Move logic to helper

After

```
<% if editable?(post) %>
  <%= link_to("Edit", edit_post_path(post))%>
<% end %>
```

Best Practice Lesson #2

Pre-decorate with Helper

常用欄位預先使用 Helper 整理

Pre-decorate with Helper

```
<%= @topic.content %>
```

Pre-decorate with Helper

always become this

```
<%= auto_link(truncate
        (simple_format
            (topic.content)),
                :length => 100) %>
```

Pre-decorate with Helper

```
<%= render_topic_content(@topic) %>
```

Do it at beginning

Common Case

- render_post_author
- render_post_published_date
- render_post_title
- render_post_content

Best Practice Lesson #3

Use Ruby in Helper ALL THE TIME

全程在 Helper 裡面使用 Ruby

Use Ruby in Helper ALL THE TIME

Before

```ruby
def render_post_taglist(post, opts = {})
  # ....
    raw tags.collect { |tag|  "<a href=\"#{posts_path(:tag
=> tag)}\" class=\"tag\">#{tag}</a>" }.join(", ")
end
```

double quote issue

Use Ruby in Helper ALL THE TIME

Before

```
def render_post_taglist(post, opts = {})
  # ....
  raw tags.collect { |tag| "<a href='#{posts_path(:tag =>
tag)}' class='tag'>#{tag}</a>" }.join(", ")
end
```

single quote issue

Use Ruby in Helper ALL THE TIME

After

```ruby
def render_post_taglist(post, opts = {})
# ...
  raw tags.collect { |tag| link_to(tag,posts_path(:tag =>
tag)) }.join(", ")
end
```

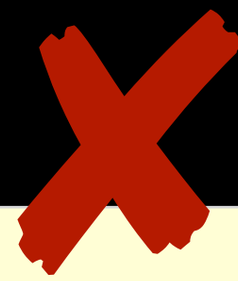
Best Practice Lesson #4

mix Helper & Partial

混合使用 Helper 與 Partial

mix Helper & Partial

Before



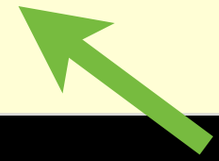
```
def render_post_title(post)
  str = “”
  str += “<li>”
  str += link_to(post.title, post_path(post))
  str += “</li>”
  return raw(str)
end
```

XSS vulnerability

mix Helper & Partial

After

```
def render_post_title(post)
  render :partial => "posts/title_for_helper", :locals =>
{ :title => post.title }
end
```



Best Practice Lesson #5

Tell, Don’t ask

先 Query 再傳入 Helper

Tell, Don't ask

Before

```
def render_post_taglist(post, opts = {})
  tags = post.tags
  tags.collect { |tag| link_to(tag,posts_path(:tag =>
tag)) }.join(", ")
end
```

```
<% @posts.each do |post| %>
  <%= render_post_taglist(post) %>
<% end %>
```

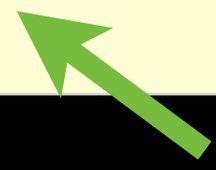
Performance issue

Tell, Don't ask

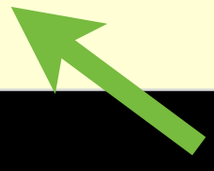
After

```
def render_post_taglist(tags, opts = {})
  tags.collect { |tag| link_to(tag,posts_path(:tag =>
tag)) }.join(", ")
end
```

```
<% @posts.each do |post| %>
  <%= render_post_taglist(post.tags) %>
<% end %>
```



```
def index
  @posts = Post.recent.includes(:tags)
end
```



Best Practice Lesson #6

Wrap into a method

資料儘量包裝成 method 而非放在 Helper

Wrap into a method

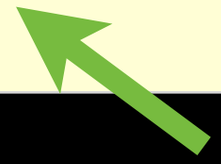
Before

```
def render_comment_author(comment)
  if comment.user.present?
    comment.user.name
  else
    comment.custom_name
  end
end
```

Wrap into a method

After

```
def render_comment_author(comment)
  comment.author_name
end
```



```
class Comment < ActiveRecord::Base
  def author_name
    if user.present?
      user.name
    else
      custom_name
    end
  end
end
```

Partial

Best Practice Lesson #7

Move code to Partial

view code 超過兩頁請注意

Move Code to Partial

- highly duplicated　　內容高度重複
- independent blocks　可獨立作為功能區塊

Common Case

- nav/user_info
- nav/admin_menu
- vendor_js/google_analytics
- vendor_js/disqus_js
- global/footer

Best Practice Lesson #8

Use presenter to clean the view

使用 Presenter 解決 login in view 問題

Use presenter to clean the view

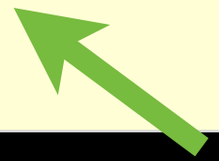
Before

```
<%= if user_rofile.has_experience? && user_rofile.experience_public? %>
  <p><strong>Experience:</strong> <%= user_profile.experience %></p>
<% end %>
```

Use presenter to clean the view

After

```
<% user_profile.with_experience do %>
  <p><strong>Experience:</strong> <%= user_profile.experience %></p>
<% end %>

<% user_profile.with_hobbies do %>
  <p><strong>Hobbies:<strong> <%= user_profile.hobbies %></p>
<% end %>
```



Use presenter to clean the view

After

```
class ProfilePresenter < ::Presenter
  def with_experience(&block)
    if profile.has_experience? && profile.experience_public?
      block.call(view)
    end
  end
end
```

Best Practice Lesson #9

Cache Digest

default since Rails 4.0+

Cache Digest

```
<% @project do %>
  aaa
  <% @todo do %>
    bbb
    <% @todolist do %>
      ccc
    <% end %>
  <% end %>
<% end %>
```

David Heinemeier Hansson (37signals)

Basecamp Projects Calendar Daily Progress Everyone Me

Jump to a project, person, or search...

v180/projects/67-20120217171705

BCX: Building BCX

The catch-all BCX project.

Invite more people
31 people on this project

Catch up
on recent changes

1132 Discussions 105 To-dos 1025 Files 17 Text documents Add the first: Date

Discussions

Start a discussion

Jonas D.	**Status update** - A little scattered today... • Finished and deployed attachments view cleanup • Investigated an autoresize issue on the new project dialog • Started looking at adding a starred group	6:51am	4
Noah L.	**Supporting BCX at launch** - I hooked up a branch of the help site that modifies the BCX help form to look more like the style SU hooked up for the Wufoo form (not perfect -- I can't quite seem to unwind	1:30am	6
Jeffrey H.	**Revisiting the http/https split for basecamp.com** - 123.basecamphq.com!	Feb 16	9
Sam S.	**With the to-do assignment dropdown focused, hittin** - (Never mind, just got the notifications. Our mail delivery must be backed up again.)	Feb 16	18
Noah L.	**Come early to Chicago and sprint on BCX** - Not sure how useful I am in person, but I could come a few days early as well.	Feb 16	17

1127 more discussions

v15/projects/67-20120217171705/todolists/project

To-do lists

Add a to-do list

Waiting for design

- "project gallery" link on projects/new doesn't link to anything 1 comment Jason Fried
- Review blank slates and ramp-up 2 comments
- Remove suite upgrade callouts from BC, HR, BP and CF account tabs
- Push claim as the primary flow on RSVP when we know there is a 37id with your email address already 1 comment Ryan Singer

Add a to-do

Design in progress

- Calendar

Add a to-do

v10/todolists/67-20120217171705

Ready for programmers

- "Branded" Launchpad for BC 12 comments — v45/todos/45-20120217151545
- Limits and upsells 33 comments
- Move events between calendars (aka "move between buckets" back-end) 1 comment Jeffrey Hardy
- Migration Jeremy Kemper
- Make sign-up respect new 37id unique-email and email-as-username policy
- To-do dates, sans calendar integration 9 comments David Heinemeier Hansson
- Automatically credit suite users 50% of the value of their suite against their BCX account
- Security: wipe localStorage when signing out or signing in with a different 37id; ensure HTTP cache doesn't leak permissions 3 comments

Add a to-do

Cache Digest

Difficult to invalid cache

```
<% cache @project do %>
  aaa
  <% cache @todo do %>
    bbb
    <% cache @todolist do %>
      ccc
    <% end %>
  <% end %>
<% end %>
```

Cache Digest

invalid cache manually

```
<% cache [v15,@project] do %>
  aaa
  <% cache [v10,@todo] do %>
    bbb
    <% cache [v45,@todolist] do %>
      zzz
    <% end %>
  <% end %>
<% end %>
```

Cache Digest

md5_of_this_view

```
<% cache @todolist do %>
  zzz
<% end %>
```

Cache Digest

Auto invalid

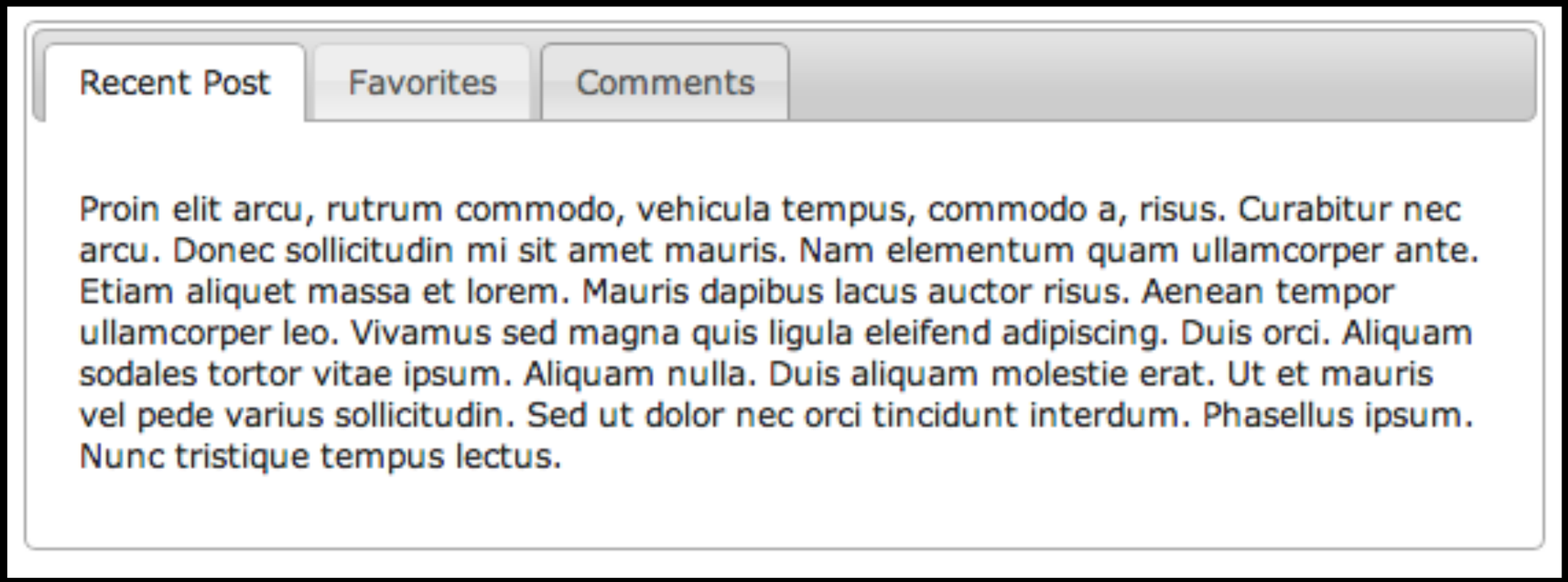
```
<% cache @project do %>
  aaa
  <% cache @todo do %>
    bbb
    <% cache @todolist do %>
      ccc
    <% end %>
  <% end %>
<% end %>
```

Best Practice Lesson #7

Cells

separate multiple logic view

Cells


Recent Post
Favorites
Comments
Proin elit arcu, rutrum commodo, vehicula tempus, commodo a, risus. Curabitur nec arcu. Donec sollicitudin mi sit amet mauris. Nam elementum quam ullamcorper ante. Etiam aliquet massa et lorem. Mauris dapibus lacus auctor risus. Aenean tempor ullamcorper leo. Vivamus sed magna quis ligula eleifend adipiscing. Duis orci. Aliquam sodales tortor vitae ipsum. Aliquam nulla. Duis aliquam molestie erat. Ut et mauris vel pede varius sollicitudin. Sed ut dolor nec orci tincidunt interdum. Phasellus ipsum. Nunc tristique tempus lectus.

Cells

Before

```
class UsersController < ApplicationController
  def show
    @user = User.find(params[:id])
    @recent_posts = @user.recent_posts.limit(5)
    @favorite_posts = @user.favorite_posts.limit(5)
    @recent_comments = @user.comments.limit(5)
  end
end
```

```
<%= render :partial => "users/recent_post", :collection => @recent_posts %>
<%= render :partial => "users/favorite_post", :collection => @favorite_posts %>
<%= render :partial => "users/recent_comment", :collection => @recent_comments %>
```

What if ....?

- each block cache expire in 3 / 5 / 7 hours?
- each block need to do different tweaks?

Huge Mess

- Long & Ugly Controller code
- Bad performance in controller & view
- Hard to cache

Cells

https://github.com/apotonick/cells

Cells

After

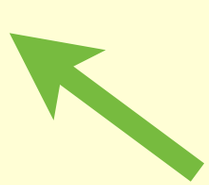
```
<%= render_cell :user, :rencent_posts, :user => @user %>
<%= render_cell :user, :favorite_posts, :user => @user %>
<%= render_cell :user, :recent_comments, :user => @user %>
```

```
class UsersController < ApplicationController
  def show
    @user = User.find(params[:id])
  end
end
```

Cells

After

```
class UserCell < Cell::Rails

  cache :recent_posts, :expires_in => 1.hours
  cache :favorite_posts, :expires_in => 3.hours
  cache :recent_comments, :expires_in => 5.hours

  def recent_posts(args)
    ...
  end

  def favorite_posts(args)
    ...
  end

  def recent_comments(args)
    ...
  end
end
```



文字

Best Practice Lesson #11

content_for / yield

jump to right location

<%= yield %> ?

yield

```
<%= stylesheet_link_tag "application" %>
  <%= yield %>
<%= javascript_include_tag "application" %>
```

put javascript at bottom

Best Practices for Speeding Up Your Web Site

yield

undefined javascript

```
main content here

<script type= "text/javascript">
  your script here
</script>
```

yield

```
<%= stylesheet_link_tag "application" %>
  <%= yield %>
<%= javascript_include_tag "application" %>
```

T_T

```
<%= stylesheet_link_tag "application" %>
<%= javascript_include_tag "application" %>
<%= yield %>
```

yield

```
<%= stylesheet_link_tag "application" %>
  <%= yield %>
<%= javascript_include_tag "application" %>
```

^_^

```
<%= stylesheet_link_tag "application" %>
<%= yield %>
<%= javascript_include_tag "application" %>
<%= yield :page_specific_javascript %>
```

yield

```
main content here

<%= content_for :page_specific_javascript do %>
  <script type= "text/javascript">
    your script here
  </script>
<% end %>
```

apply on sidebar

content_for / yield

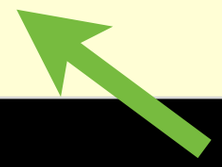
Before

```
<div class="main">
  main content
</div>

<div class="sidebar">
  <% case @ad_type %>
  <% when foo %>
    <%= render "ad/foo"%>
  <% when bar %>
    <%= render "ad/bar"%>
  <% else %>
    <%= render "ad/default"%>
  <% end %>
</div>
```

content_for / yield

After

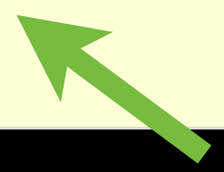
```
<div class="main">
  <%= yield %>
</div>

<div class="sidebar">
  <%= yield :sidebar %>
</div>
```



content_for / yield

After

```
main content

<%= content_for :sidebar do %>
  <%= render "ad/foo"%>
<% end %>
```



Best Practice Lesson #12

Decoration in Controller

Decoration in Controller

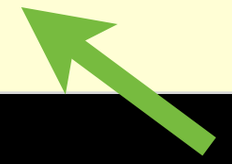
Before

```
<%= content_for :meta do %>
  <meta content="xdite's blog" name="description">
  <meta content="Blog.XDite.net" property="og:title">
<% end %>
```

Decoration in Controller

After

```
def show
  @blog = current_blog
  drop_blog_title @blog.name
  drop_blog_descption @blog.description
end
```



```
<%= stylesheet_tag "application" %>
<%= render_page_title %>
<%= render_page_descrption %>
```

Best Practice Lesson #13

Decoration using I18n

Decoration in I18n

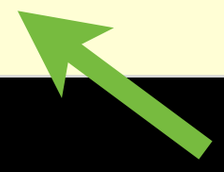
Before

```
def render_user_geneder(user)
  if user.gender == "male"
    "男 (Male)"
  else
    "女 (Female)"
  end
end
```

Decoration in I18n

After

```
def render_user_gender(user)
  I18n.t("users.gender_desc.#{user.geneder}")
end
```



Decoration in I18n

```
def render_book_purchase_option(book)
  if book.available_for_purchase?
    "Yes"
  else
    "No"
  end
end
```

Best Practice Lesson #14

Decoration using Decorator

don’t put everything in model

Decoration using Decorator Before

```
def render_article_publish_status(article)
  if article.published?
    "Published at #{article.published_at.strftime('%A, %B %e')}"
  else
    "Unpublished"
  end
end
```

Decoration using Decorator Before

```
class Article < ActiveRecord::Base
  def human_publish_status
    if published?
      "Published at #{article.published_at.strftime('%A, %B %e')}"
    else
      "Unpublished"
    end
  end
end
```

Decoration using Decorator Before

```
class Article < ActiveRecord::Base

  def human_publish_status
  end

  def human_publish_time
  end

  def human_author_name
  end

  ........
end
```

Decoration using Decorator Before

Decoration using Decorator After

```
<%= @article.publication_status %>
```

Draper

Decorators/View-Models for Rails Applications

https://github.com/drapergem/draper

Decoration using Decorator After

```ruby
class ArticleDecorator < Draper::Decorator
  delegate_all

  def publication_status
    if published?
      "Published at #{published_at}"
    else
      "Unpublished"
    end
  end

  def published_at
    object.published_at.strftime("%A, %B %e")
  end
end
```

```ruby
def show
  @article = Article.find(params[:id]).decorate
end
```

Best Practice Lesson #15

Decoration using View Object

Decoration using View Object

Before

```
<dl class="event-detail">
  <dt>Event Host</dt>
  <dd>
    <% if @event.host == current_user %>
      You
    <% else %>
      <%= @event.host.name %>
    <% end %>
  </dd>
  <dt>Participants</dt>
  <dd>
  <%= @event.participants.reject { |p|
     p == current_user }.map(&:name).join(", ") %>
    </dd>
</dl>
```

Decoration using View Object

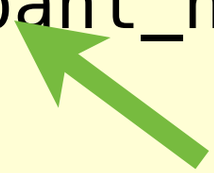
Before

```
class EventDetailView
  def initialize(template, event, current_user)
    @template = template
    @event = event
    @current_user = current_user
  end

  def host
    if @event.host == @current_user
      "You"
    else
      @event.host.name
    end
  end

  def participant_names
    participants.map(&:name).join(", ")
  end


  private

  def participants
    @event.participants.reject { |p| p == @current_user }
  end
end
```

Decoration using View Object

After

```
<dl class="event-detail">
  <dt>Host</dt>
  <dd><%= event_detail.host %></dd>
  <dt>Participants</dt>
  <dd><%= event_detail.participant_names %></dd>
</dl>
```



Best Practice Lesson #16

Form builder

simplify complex form

Form Builder

Before

```
<%= form_for @user do |form| %>
  <div class="field">
    <%= form.label :name %>
    <%= form.text_field :name %>
  </div>

  <div class="field">
    <%= form.label :email %>
    <%= form.text_field :email %>
  </div>
<% end %>
```

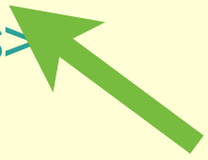
Form Builder

```
class HandcraftBuilder < ActionView::Helpers::FormBuilder
  def custom_text_field(attribute, options = {})
    @template.content_tag(:div, class: "field") do
      label(attribute) + text_field(attribute, options)
    end
  end
end
```

Form Builder

After

```
<%= form_for @user, :builder => HandcraftBuilder do |form| %>

  <%= form.custom_text_field :name %>
  <%= form.custom_text_field :email %>

<% end %>
```



popular form builders

- simple_form
- bootstrap_form

Best Practice Lesson #17

Form Object

wrap logic in FORM, not in model nor in controller

Form Object

Before

```
<%= simple_form_for @registration, :url =>
registrations_path, :as => :registration do |f| %>

  <%= f.input :name %>
  <%= f.input :email %>

  <label class="checkbox">
    <%= check_box_tag :terms_of_service %>
    I accept the <%= link_to("Terms of Service ", "/
pages/tos") %>
  </label>
  <%= f.submit %>
<% end %>
```

Form Object

Before

```
def create
  if params[:agree_term]
    if @registration.save
      redirect_to root_path
    else
      render :new
    end
  else
    render :new
  end
end
```

Form Object

After

```
<%= simple_form_for @form :url =>
registrations_path, :as => :registration do |f| %>

  <%= f.input :name %>
  <%= f.input :email %>

  <label class="checkbox">
    <%= f.input :terms_of_service %>
    I accept the <%= link_to("Terms of Service ", "/
pages/tos") %>
  </label>
  <%= f.submit %>
<% end %>
```

Form Object

After

```
def new
  @form = RegistrationForm.new(Registration.new)
end
```

Reform

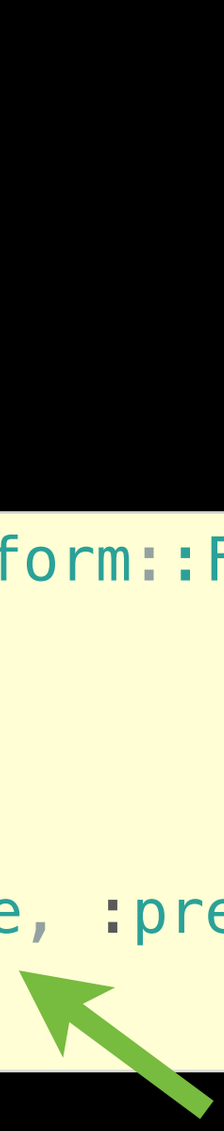
Decouples your models from form validation, presentation and workflows.

https://github.com/apotonick/reform

Form Object

After

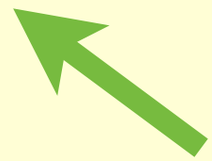
```
class RegistrationForm < Reform::Form
  property :name
  property :email
  property :term_of_service

  validates :term_of_service, :presence => true
end
```



Form Object

After

```
def create
  if @form.validate(params[:registration])
    @form.save
  else
    render :new
  end
end
```



Best Practice Lesson #18

Policy Object / Rule Engine

centralize permission control

last one!!

Policy Object / Rule Engine

Before

```
def render_post_edit_option(post)
  if post.user == current_user
    render :partial => "post/edit_bar"
  end
end
```

Policy Object / Rule Engine

Before

```
def render_post_edit_option(post)
  if post.user == current_user || current_user.admin?
    render :partial => "post/edit_bar"
  end
end
```

Policy Object / Rule Engine

Before

```
def render_post_edit_option(post)
  if post.user == current_user || current_user.admin? || current_user.moderator?
    render :partial => "post/edit_bar"
  end
end
```

Policy Object / Rule Engine

Before

```
class PostController < ApplicationController
  before_filter :check_permission, :only => [:edit]

  def edit
    @post = Post.find(params[:id])
  end
end
```

Pundit

Minimal authorization through OO design and pure Ruby classes

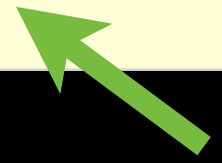
https://github.com/elabs/pundit

Policy Object (Pundit) After

```
class PostPolicy
  attr_reader :user, :post

  def initialize(user, post)
    @user = user
    @post = post
  end

  def edit?
    user.admin? || user.moderator?
  end
end
```

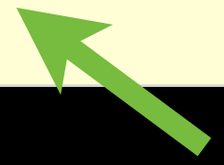
Policy Object (Pundit) After

```
<% if policy(@post).edit? %>
  <%= render :partial => "post/edit_bar" %>
<% end %>
```



Rule Engine (CanCan) After

```
<% if can? :update, @post %>
  <%= render :partial => "post/edit_bar" %>
<% end %>
```



Cancan

Authorization Gem for Ruby on Rails.

https://github.com/ryanb/cancan

Rule Engine (CanCan)

```
class Ability
  include CanCan::Ability

  def initialize(user)

    if user.blank?
      # not logged in
      cannot :manage, :all
    elsif user.has_role?(:admin)
      can :manage, :all
    elsif user.has_role?(:moderator)
      can :manage, Post
    else
      can :update, Post do |post|
        (post.user_id == user.id)
      end
    end
  end
```

Recap

- Always assume things need to be decorated
- Extract logic into methods / classes
- Avoid perform query in view/helper
- When things get complicated, build a new control center

Reference

- http://blog.xdite.net
- https://github.com/bloudermilk/maintainable_templates
- http://pivotallabs.com/form-backing-objects-for-fun-and-profit/
- http://saturnflyer.com/blog/jim/2013/10/21/how-to-make-your-code-imply-responsibilities/
- http://objectsonrails.com/

Thanks

xdite@rocodev.com