Cleaning up views with Decorators & Helpers

Rebecca Miller-Webster
Game night at The Brooklyn Strategist, plus gourmet refreshments  
**FREE DATE** for members only  

Beverages and bacony treats at Baconery Cafe

Two tickets to Rattlestick Theatre’s *Buyer & Cellar*

Burgers, shakes, and live music at Sidewalk Cafe
Problem: Our Views are messy

- logic in views is harder to test
- views are difficult to reason about
Decorators & Helpers

WHEN SHIT GETS COMPLICATED
Decorators

Logic moves with the OBJECT

Helpers

Logic moves with the VIEW
This

```html
.pricing{ class: "#{offer.members_only ? 'membersOnly' : ''} #{additional_classes}" }
- if offer.members_only
  - if offer.free_for_member?
    .price.free
      %strong
        Free Date
      %em
        for members only
  - else
    %strong.details
      Members
      %br
      Only
    %dl.price
      %dt.title
        Member
      %dd.value
        = "#{offer_member_price(offer, {unit_tag: 'small'})}".html_safe
  - else
    %strong.details
      Save
      %br
      #{saving_percentage(offer)}
    %dl.price
      %dt.title
        Member
      %dd.value
        = "#{offer_member_price(offer, {unit_tag: 'small'})}".html_safe
    %dl.price
      %dt.title
        Standard
      %dd.value
        = "#{offer_non_member_price(offer, {unit_tag: 'small'})}".html_safe

%footer{ class: "#{'darkSeparatorTop' if add_separator}" }
- if offer_booked?(offer)
  %strong.booked
    Booked
- elsif offer.expired?
  %strong.expired
    = t('.expired')
- elsif offer.sold_out?
  %strong.sold_out
    = t('offers.offer.sold_out')
```
.pricing{ class: offer.price_classes }  
%strong.priceDetails= offer.price_details  
= offer.price_definition

%footer{ class: offer.price_classes }  
= offer.status_tag
How?
Generate classes

```
.pricing { class: offer.price_classes }
%strong.price_details = offer.price_details
    = offer.price_definition

%footer{ class: offer.price_classes }
    = offer.status_tag
```

def price_classes
    classes = []
    classes << 'membersOnly' if members_only
    classes << (free_for_member? ? 'free' : 'paid')
    classes << 'unavailable' if !bookable? || h.offer_booked?(self)
    classes << 'nonMembers' if !members_only && non_member_price > 0
    classes.join(' ')
end
def price_details
    str = ""
    if members_only && free_for_member?
        str = "Free Date <br />\<em\>for members only\</em\>"
    elsif members_only
        str = "Members <br />\Only\"
    else
        str = "Save <br />&{h.saving_percentage(self)}"
    end
    str.html_safe
end
Encapsulate repetition in partials

Sometimes zero
Sometimes one
Sometimes two
%strong.priceDetails= offer.price_details

= offer.price_definition

%footer{ class: offer.price_classes }
= offer.status_tag

def price_definition
  unless free_for_member? && members_only
    (text = '') << h.render(partial: 'shared/offer_pricing_dl',
      locals: { offer: self, member: true })
  end
  if !members_only
    text << h.render(partial: 'shared/offer_pricing_dl',
      locals: { offer: self, member: false })
  end
  text.html_safe
end
end
Generate html attributes

From

```ruby
- if current_user.has_booked_this_month?
  - html_config = { id: 'subscribed_new_booking', :class => 'track_submit', :data-event-name => "submit: upsell" }
- else
  - html_config = { id: 'subscribed_new_booking' }

= form_for([offer, Booking.new], url: offer_booking_path(offer, secure: true), html: html_config) do |f|
```

TO

```ruby
= form_for(@booking, offer.booking_form_attributes) do |f|
```

WITH

```ruby
def booking_form_attributes
  attrs = {
    url: h.offer_booking_path(self, secure: true),
    html: { id: 'new_booking' }

  if h.current_user.has_booked_this_month?
    attrs[:html][:class] = 'track_submit'
    attrs[:html][:data-event-name] = "submit: upsell"

  end

  attrs
end
```
Thank you!