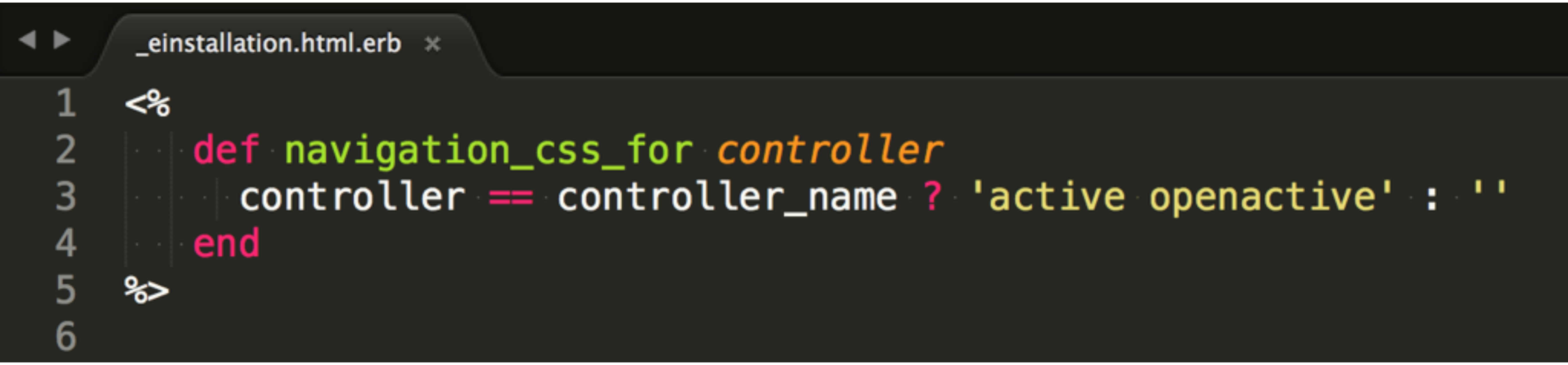
Rails: presenters

Przemysław Dąbek

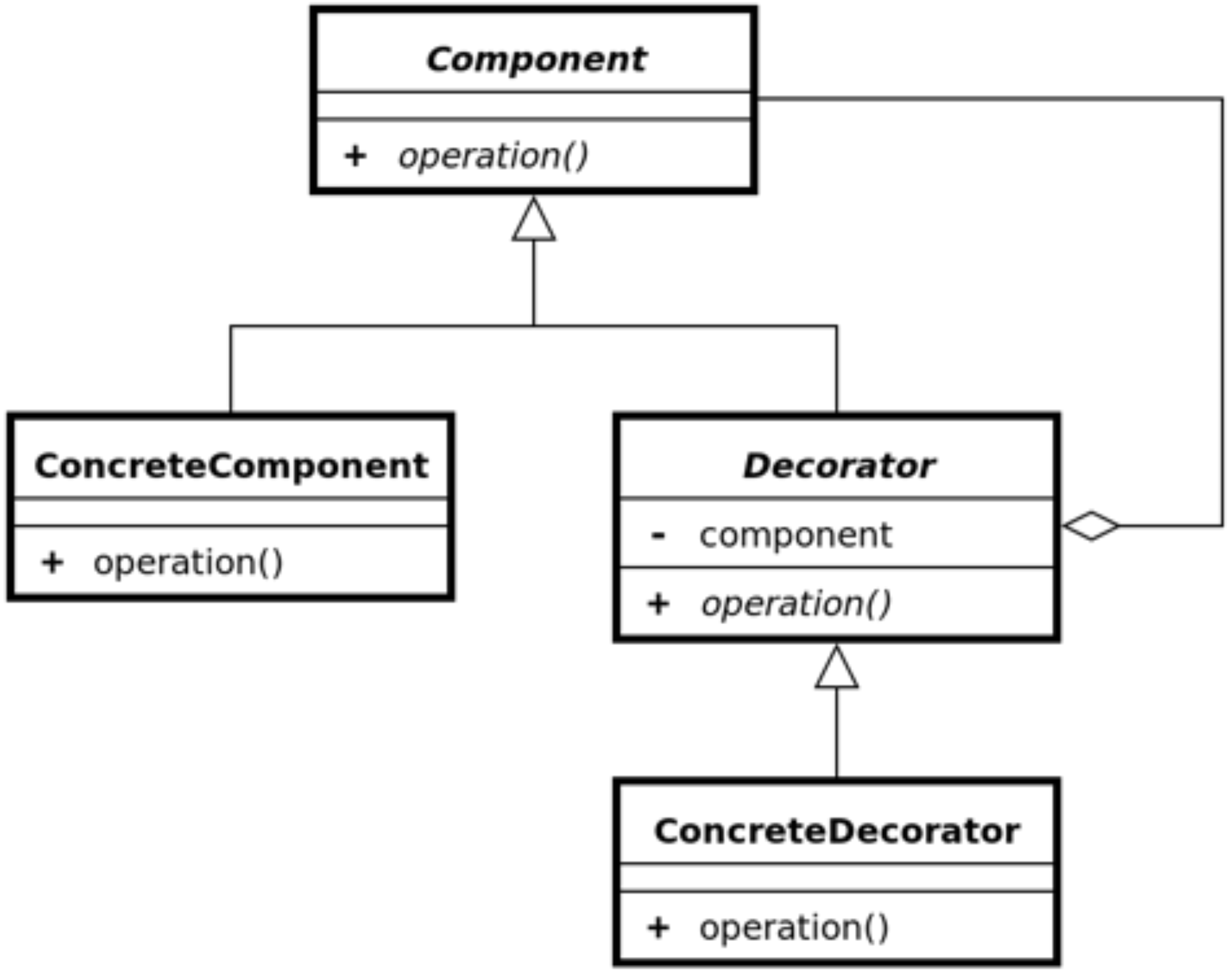
Quiz time



_einstallation.html.erb

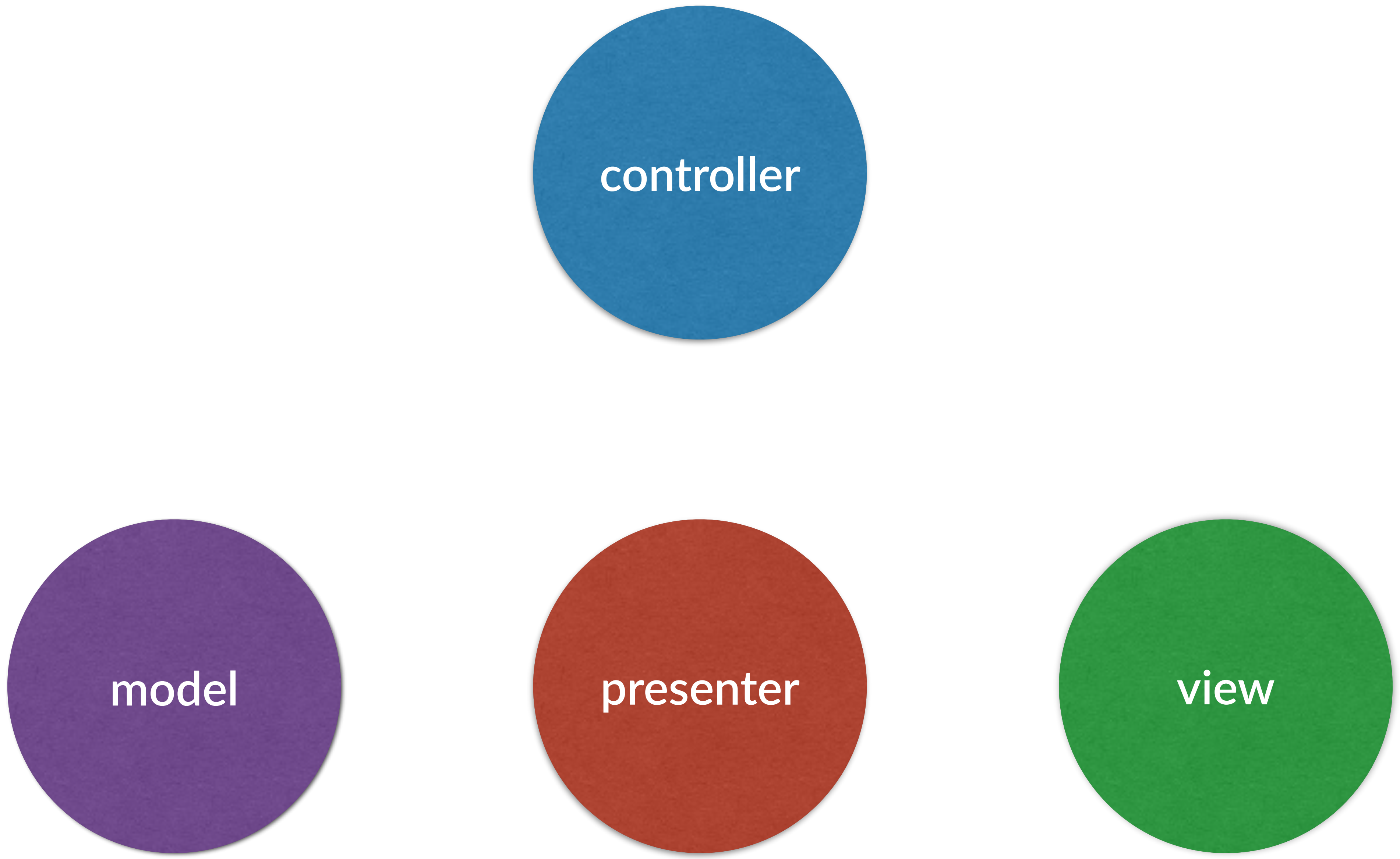
```
<%
    def navigation_css_for controller
      controller == controller_name ? 'active openactive' : ''
    end
%>
```

Decorator pattern



http://en.wikipedia.org/wiki/Decorator_pattern

Presenter



Draper

https://github.com/drapergem/draper

adds an object-oriented layer of presentation logic

Imaginary problem

- list of articles
- when article is published display publication date, otherwise display “Unpublished”

“the helper way”

```ruby
# app/helpers/articles_helper.rb
def publication_status(article)
  if article.published?
    "Published at #{article.published_at.strftime('%A, %B %e')}"
  else
    "Unpublished"
  end
end
```

Draper: structure

```
app
└── decorators
    └── article_decorator.rb
```

Draper: decorator class

```
# app/decorators/article_decorator.rb
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      "Unpublished"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
```

Draper: decorator class

```ruby
# app/decorators/article_decorator.rb
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      "Unpublished"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
```

Draper: decorator class

```ruby
# app/decorators/article_decorator.rb
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      "Unpublished"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
```

Draper: decorator class

```ruby
# app/decorators/article_decorator.rb
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
```

Draper: usage

Single object:

```
article = Article.find(params[:id]).decorate
article = ArticleDecorator.new(Article.find(params[:id]))
article = ArticleDecorator.decorate(Article.find(params[:id]))
```

Collection — decorating individual elements

```
articles = ArticleDecorator.decorate_collection(Article.all)
# Rails 4
articles = Article.all.decorate
```

Curly

https://github.com/zendesk/curly

template language allows separating your logic from the structure of your HTML templates

Curly: structure

```
app
├── presenters
│   └── articles
│       └── show_presenter.rb
└── views
    └── articles
        └── show.curly
```

Curly: presenter class

```ruby
# app/presenters/show_presenter.rb
class Articles::ShowPresenter < Curly::Presenter
  presents :article

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

  def article
    @article
  end
end
```

Curly: presenter class

```ruby
# app/presenters/show_presenter.rb
class Articles::ShowPresenter < Curly::Presenter
  presents :article

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

  def article
    @article
  end
end
```

Curly: presenter class

```ruby
# app/presenters/show_presenter.rb
class Articles::ShowPresenter < Curly::Presenter
  presents :article

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

  def article
    @article
  end
end
```

Curly: view

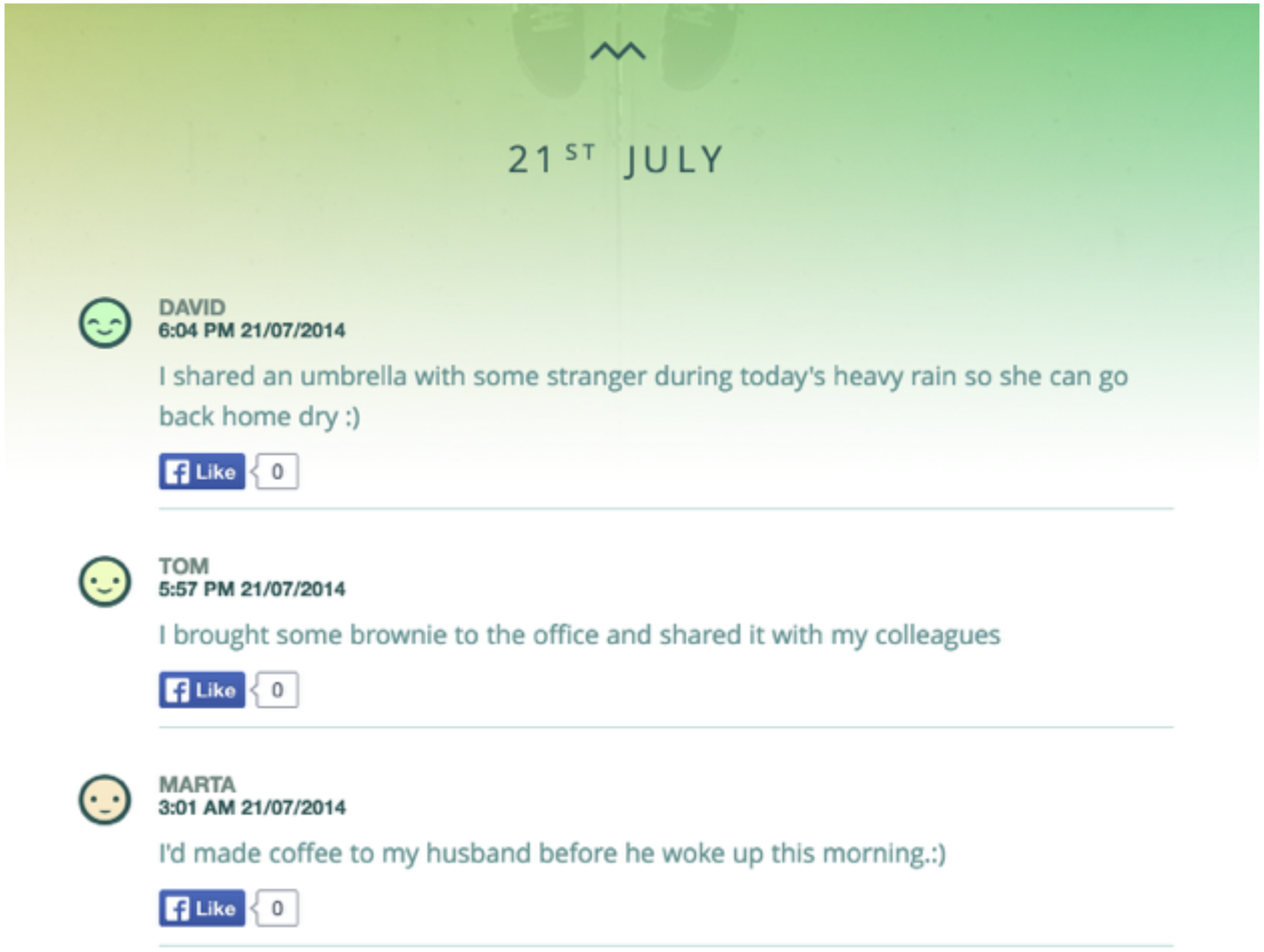
```
# app/views/articles/show.curly
...
{{publication_status}}
...
```

Cells

https://github.com/apotonick/cells

allow you to encapsulate parts of your page into separate MVC components (view models)

Cells: case study



21 ST JULY

DAVID
6:04 PM 21/07/2014
I shared an umbrella with some stranger during today's heavy rain so she can go back home dry :)

TOM
5:57 PM 21/07/2014
I brought some brownie to the office and shared it with my colleagues

MARTA
3:01 AM 21/07/2014
I'd made coffee to my husband before he woke up this morning.:)

Cells: structure

```
app
└── cells
    ├── deed
    │   └── show.html.haml
    └── deed_cell.rb
```

Cells: cell class

```ruby
# app/cells/deed_cell.rb
class DeedCell < Cell::ViewModel
  property :person
  property :action

  def show
    render
  end

  private

  def time
    model.created_at.strftime("%l:%M %P")
  end
end
```

Cells: cell class

```ruby
# app/cells/deed_cell.rb
class DeedCell < Cell::ViewModel
  property :person
  property :action

  def show
    render
  end

  private

  def time
    model.created_at.strftime("%l:%M %P")
  end
end
```

Cells: cell class

```ruby
# app/cells/deed_cell.rb
class DeedCell < Cell::ViewModel
  property :person
  property :action

  def show
    render
  end

  private

  def time
    model.created_at.strftime("%l:%M %P")
  end
end
```

Cells: cell class

```ruby
# app/cells/deed_cell.rb
class DeedCell < Cell::ViewModel
  property :person
  property :action

  def show
    render
  end

  private

  def time
    model.created_at.strftime("%l:%M %P")
  end
end
```

Cells: cell view

```haml
# app/cells/deeds/show.html.haml
%h2= person
%h3= time
%p= action
```

Cells: rendering cell

```
# app/views/landing/_deeds.html.haml

# ...
      - deeds.each do |deed|
        .deed= cell(:deed, deed).show
# ...
```

Presenters: summary

- format data for displaying
- extract/remove logic
- loosen coupling between views & models

Gems

https://www.ruby-toolbox.com/categories/rails_presenters

https://github.com/drapergem/draper

https://github.com/apotonick/cells

https://github.com/zendesk/curly

More to read...

http://robots.thoughtbot.com/evaluating-alternative-decorator-implementations-in

http://robertomurray.co.uk/blog/2014/decorators-presenters-delegators-rails/

http://nicksda.apotomo.de/2011/10/rails-misapprehensions-helpers-are-shit/

http://mikepackdev.com/blog_posts/31-exhibit-vs-presenter

Thanks!