

Model-View-Controller MVC

Web Programming Paradigm

Model

1. Classes

1. Objects/Data
2. Business Logic

2. Database

1. Interacts with Database

Keyword: Data

```
1  class Wiki < ActiveRecord::Base
2    belongs_to :user
3    has_many :users, through: :collaborators
4    has_many :collaborators
5
6    validates :title, length: { minimum: 5, maximum: 255 }, presence: true
7    validates :body, length: { minimum: 20 }, presence: true
8    validates :user, presence: true
9
10   extend FriendlyId
11   friendly_id :title, use: [:slugged, :history]
12
13   def user_collaborators
14     users = []
15     collaborators = Collaborator.includes(:user).where(wiki_id: self).all
16     collaborators.each do |collaborator|
17       users.push(collaborator.user)
18     end
19     users
20   end
21
22   def self.visible_to(user) # rubocop:disable Metrics/AbcSize
23     wikis = []
24     if user
25       if user.role == 'admin'
26         wikis = Wiki.all
27       else
28         wikis = Wiki.where('user_id=? OR private=?', user.id, false)
29         collaborators = Collaborator.includes(:wiki).where(user_id: user).all
30         collaborators.each do |collaborator|
31           unless wikis.include?(collaborator.wiki) # rubocop:disable Style/IfUnlessModifier
32             wikis.push(collaborator.wiki)
33           end
34         end
35       end
36     else
37       wikis = Wiki.where(private: false)
38     end
39     wikis
40     # wikis.uniq # uniq method removes private wiki when user is standard
41   end
end
```

View

1. Web Page
 1. HTML/CSS
 2. Embedded JavaScript
2. User Interaction
 1. Clicking Links
 2. Data Entry

Keyword: Presentation

12 lines (9 sloc) | 263 Bytes

```
1 .row
2   .col-md-4
3     %p Guidelines for wikis
4     %ul
5       %li Titles have to be at least 5 characters
6       %li Bodys have to be big, at least 20 characters
7       %li Nothing mean
8   .col-md-8
9     = render partial: 'wikis/form_new', locals: { wiki: @wiki }
10
11
```

14 lines (14 sloc) | 510 Bytes

```
1 = form_for wiki do |f|
2   .form-group
3     = f.label :title
4     = f.text_field :title, class: 'form-control', placeholder: 'Enter title'
5   .form-group
6     = f.label :body
7     = f.text_area :body, rows: 8, class: 'form-control', placeholder: 'Enter your wiki post'
8   - if current_user.role == 'admin' || (current_user.role == 'premium')
9     .form-group
10      = f.label :private, class: 'checkbox' do
11        = f.check_box :private
12        Private wiki
13  .form-group
14    = f.submit 'Save', class: 'btn btn-success'
```

Controller

1. Routes and processes
 1. Requests
 2. Responses
2. Takes/sends data from/to view
 1. CRUD operations/HTTP Verbs

Keyword: Routing

```
1 class WikisController < ApplicationController
2   before_action :authenticate_user!, except: [:index, :show]
3
4   def index
5     @wikis = Wiki.visible_to(current_user)
6     authorize @wikis
7   end
8
9   def show
10    @wiki = Wiki.friendly.find(params[:id])
11    unless request.path == wiki_path(@wiki)
12      redirect_to @wiki, status: :moved_permanently
13    end
14    authorize @wiki
15  end
16
17  def new
18    @wiki = Wiki.new
19    authorize @wiki
20  end
21
22  def create
23    @wiki = current_user.wikis.build(wiki_params)
24    authorize @wiki
25    if @wiki.save
26      flash[:notice] = 'Your wiki was saved'
27      redirect_to @wiki
28    else
29      flash[:error] = 'Your wiki failed to save'
30      render :new
31    end
32  end
33
34  def edit
35    @wiki = Wiki.friendly.find(params[:id])
36    authorize @wiki
37  end
38
39  def update
40    @wiki = Wiki.friendly.find(params[:id])
41    authorize @wiki
42    @wiki.slug = nil
43    if @wiki.update_attributes(wiki_params)
```

Separation of Concerns

1. Representation separate from data

1. Custom Interfaces -> Views

2. APIs -> Controllers

3. Stateless | Asynchronous | RESTful

Supports Convention

Models

Style

Follow Django's [defined conventions](#) for model code.

Make 'em Fat

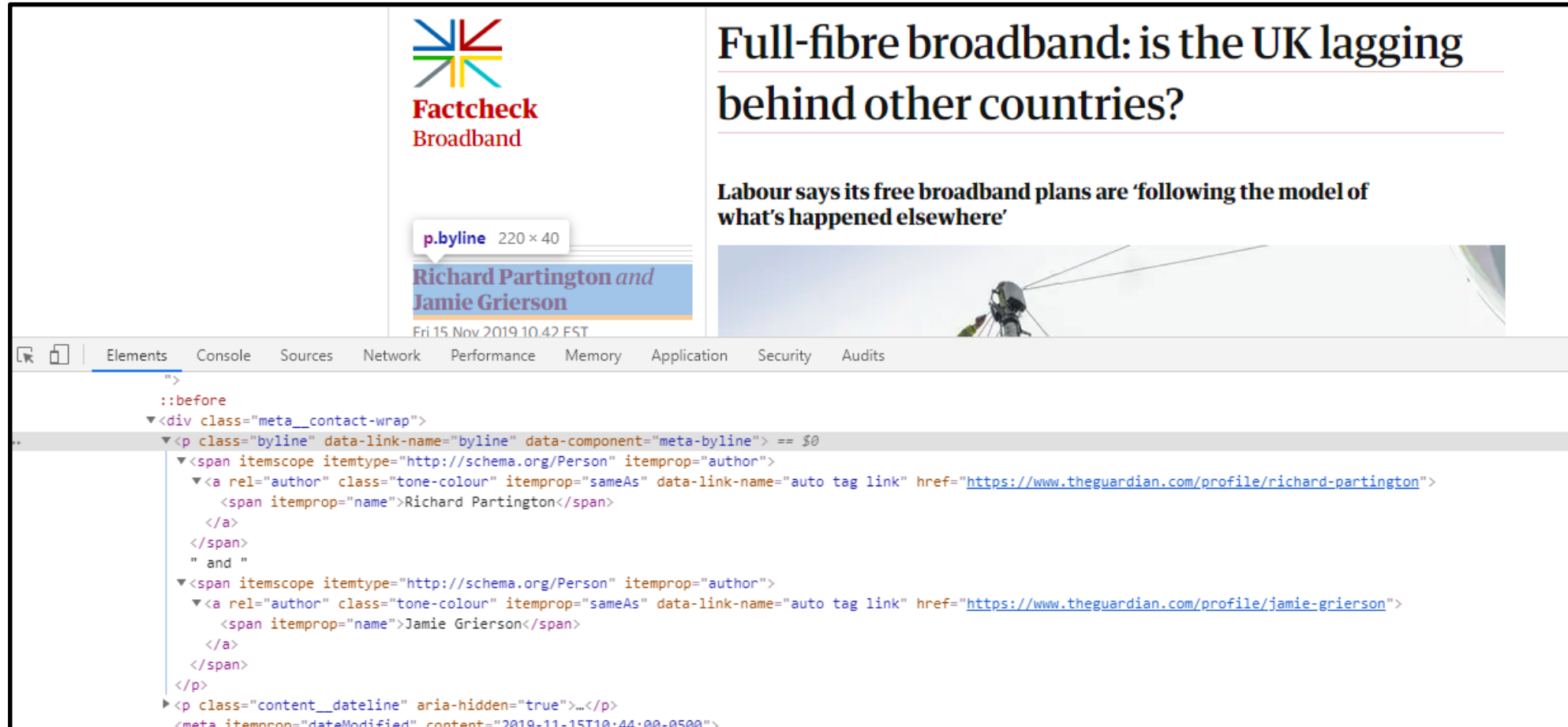
A common pattern in MVC-style programming is to build thick/fat models and **thin** controllers. For Django this translates to building models with lots of small methods attached to them and views which use those methods to keep their logic as minimal as possible. There are lots of benefits to this approach.

1. **DRY**: Rather than repeating the same logic in multiple views, it is defined once on the model.
2. **Testable**: Breaking up logic into small methods on the model makes your code easier to unit test.
3. **Readable**: By giving your methods friendly names, you can abstract ugly logic into something that is easily readable and understandable.

For a good example of a fat model in Django, look at [the definition of](#)

```
django.contrib.auth.models.User .
```

Representation



The image shows a browser window with a news article and its source code. The article is from Factcheck Broadband and is titled "Full-fibre broadband: is the UK lagging behind other countries?". The byline is "Richard Partington and Jamie Grierson". The article text is "Labour says its free broadband plans are 'following the model of what's happened elsewhere'". The browser's developer tools are open, showing the HTML source code for the byline. The code is as follows:

```
<div class="meta_contact-wrap">  
  <p class="byline" data-link-name="byline" data-component="meta-byline"> == $0  
    <span itemscope itemtype="http://schema.org/Person" itemprop="author">  
      <a rel="author" class="tone-colour" itemprop="sameAs" data-link-name="auto tag link" href="https://www.theguardian.com/profile/richard-partington">  
        <span itemprop="name">Richard Partington</span>  
      </a>  
    </span>  
    " and "  
    <span itemscope itemtype="http://schema.org/Person" itemprop="author">  
      <a rel="author" class="tone-colour" itemprop="sameAs" data-link-name="auto tag link" href="https://www.theguardian.com/profile/jamie-grierson">  
        <span itemprop="name">Jamie Grierson</span>  
      </a>  
    </span>  
  </p>  
<p class="content_dateline" aria-hidden="true">...</p>  
<meta itemprop="dateModified" content="2019-11-15T10:44:00-0500">
```

Data

```
▼ root: {} 1 key
  ▼ response: {} 4 keys
    status: "ok"
    userTier: "developer"
    total: 1
  ▼ content: {} 12 keys
    id: "politics/2019/nov/15/labour-full-fibre-broadband-is-the-uk-lagging-behind-other-countries"
    type: "article"
    sectionId: "politics"
    sectionName: "Politics"
    webPublicationDate: "2019-11-15T15:42:26Z"
    webTitle: "Full-fibre broadband: is the UK lagging behind other countries?"
    webUrl: "https://www.theguardian.com/politics/2019/nov/15/labour-full-fibre-broadband-is-the-uk-lagging-behind-other-countries"
    apiUrl: "https://content.guardianapis.com/politics/2019/nov/15/labour-full-fibre-broadband-is-the-uk-lagging-behind-other-countries"
  ▼ tags: [] 2 items
    ▼ 0: {} 12 keys
      id: "profile/richard-partington"
      type: "contributor"
      webTitle: "Richard Partington"
      webUrl: "https://www.theguardian.com/profile/richard-partington"
      apiUrl: "https://content.guardianapis.com/profile/richard-partington"
      references: [] 0 items
      bio: "<p>Richard Partington is the Guardian's economics correspondent. Twitter <a href='\"https://twitter.com/rjpartington\"'>@rjpartington&nb  
sp;</a></p>"
      bylineImageUrl: "https://uploads.guim.co.uk/2017/12/27/Richard-Partington.jpg"
      bylineLargeImageUrl: "https://uploads.guim.co.uk/2017/12/27/Richard_Partington,_R.png"
      firstName: "Richard"
      lastName: "Partington"
      twitterHandle: "RJPartington"
    ▼ 1: {} 12 keys
      id: "profile/jamie-grierson"
      type: "contributor"
      webTitle: "Jamie Grierson"
```

Maintainable

1. Separate updates to separate layers
 1. Representation (new HTML/CSS: mobile compatible)
 2. Controller (Authentication methods)
 3. Models (Changing Data Requirements: soft deletes, archiving)
2. Specialized attention
 1. Web Design
 2. UX
 3. Programming

Testable

1. Representation/Features -> Clicks, URLs
2. Controllers -> [Mock] Requests
3. Models -> Data Retrieval, Storage, Creation

Extensible

Search : Representation → Data

The screenshot displays the EBSCOhost search interface. At the top, there is a navigation bar with links for 'New Search', 'Publications', 'Subject Terms', 'Cited References', and 'More'. On the right side of the navigation bar, there are links for 'Sign Out', 'Folder', 'Preferences', and 'Language'. Below the navigation bar, the main search area includes the EBSCOhost logo, a search input field, a dropdown menu for 'Select a Field (optional)', and a green 'Search' button. There are also options for 'AND' and 'OR' search modes, a 'Clear' button, and '+' and '-' buttons to add or remove search criteria. Below the search area, there are links for 'Basic Search', 'Advanced Search', and 'Search History'. The 'Search History/Alerts' section is visible, showing a table of search results. A green arrow points from the 'Advanced Search' link to the 'Search with AND' button in the search history section.

MY
EBSCOhost

Searching: **Academic Search Complete** | [Choose Databases](#)

Select a Field (optional) ▼ Search

AND ▼ Select a Field (optional) ▼ Clear ?

AND ▼ Select a Field (optional) ▼ + -

[Basic Search](#) [Advanced Search](#) [Search History](#)

Search History/Alerts

[Print Search History](#) [Retrieve Searches](#) [Retrieve Alerts](#) [Save Searches / Alerts](#)

Select / deselect all

Search ID#	Search Terms	Search Options	Actions
<input type="checkbox"/> S1	AB higher W0 education W5 funding	Expanders - Apply equivalent subjects Search modes - Boolean/Phrase	Rerun View Details Edit