Growing Your Rails App

ATLRUG

EST.

20 06

ATLANTA RUBY USERS GROUP

Growing A Rails App

Dan Cunning

February 8, 2017

Growing Your Rails App

Dan Cunning

Rails contractor since '08

Cunning Simplicity, Inc.

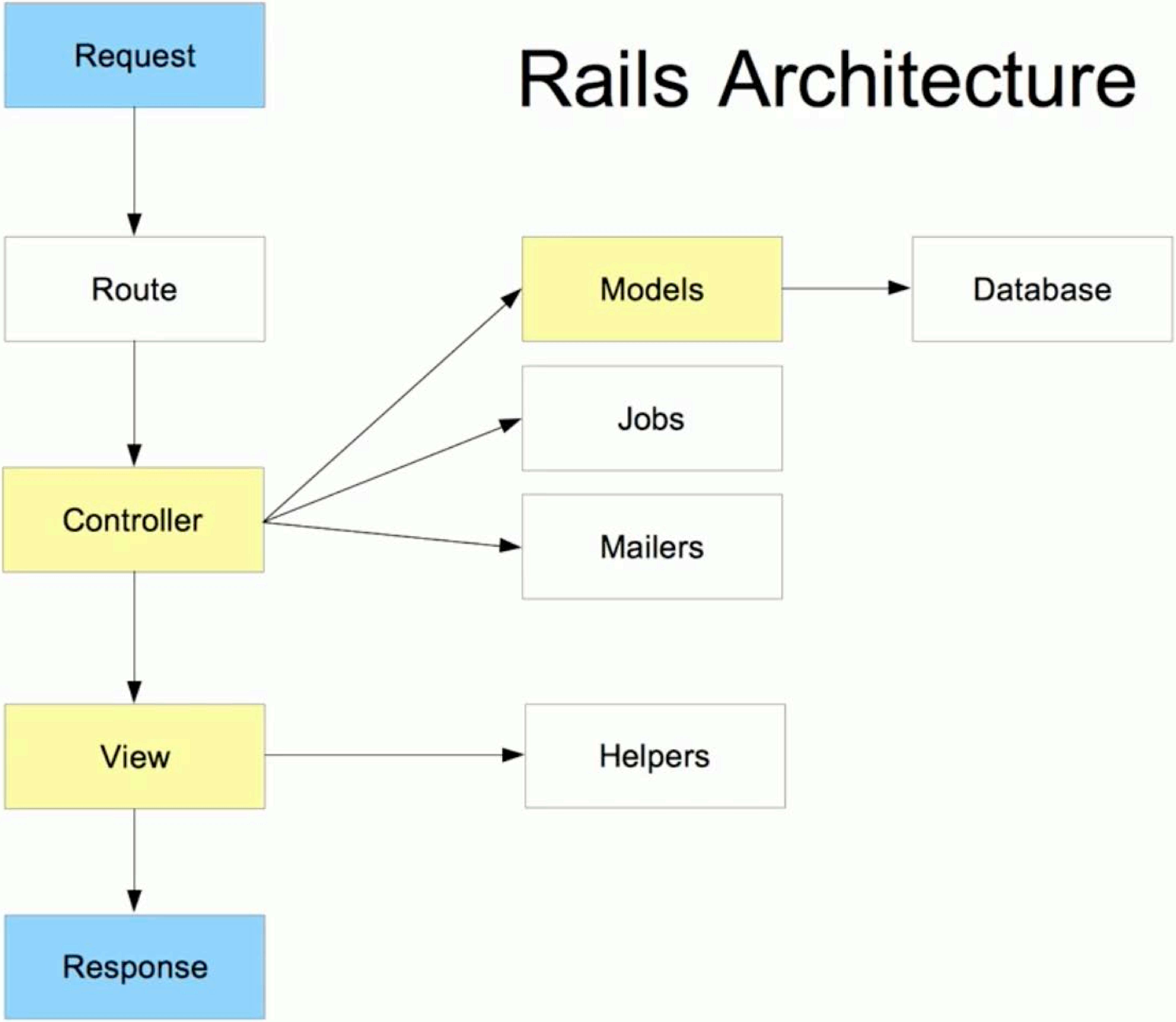
http://topdan.com

Growing Your Rails App

Photo by Ragesoss
GNU Free Documentation

Photo by Nate Steiner
Creative Commons Attribution

Rails Architecture



Controller Tip #1

Stick to the standard actions

```
index show new create edit update destroy
```

- Keeps your controllers small
- Builds RESTful Resources

Controller Tip #2

Pull logic into services

Services are “Plain Old Ruby Objects” (PORO) that consolidate logic, save models, queue background jobs and emails.

- Keeps your controller thin and secure

- Simplifies the controller's control flow

Models Tip #1

Treat models like API documentation for your database

ActiveRecord is just a really nice way to access the database.

- Keeps your models thin

- Builds a solid foundation for your app

Models Tip #2

Only use nouns, avoid verbs

Nouns / Properties:

```
user.name
user.receives_emails?

post.published?

order.total
```

Verbs / Methods:

```
user.unsubscribe!
user.send_welcome_email

post.publish!

order.calculate_total
```

Jobs Tip #1

Avoid nouns, only use verbs

Jobs and mailers act on models.

```
Users::Unsubscribe(user).perform_later

UsersMailer.welcome(user).deliver_later

Posts::Publish(post).perform_later

Orders::CalculateTotal(order).perform_now
```

Jobs Tip #2

Break jobs into smaller jobs

- Processes in parallel
- Avoids interruptions during restart
- Minimizes side-effects of exceptions

Mailers Tip #1

Never use deliver_now

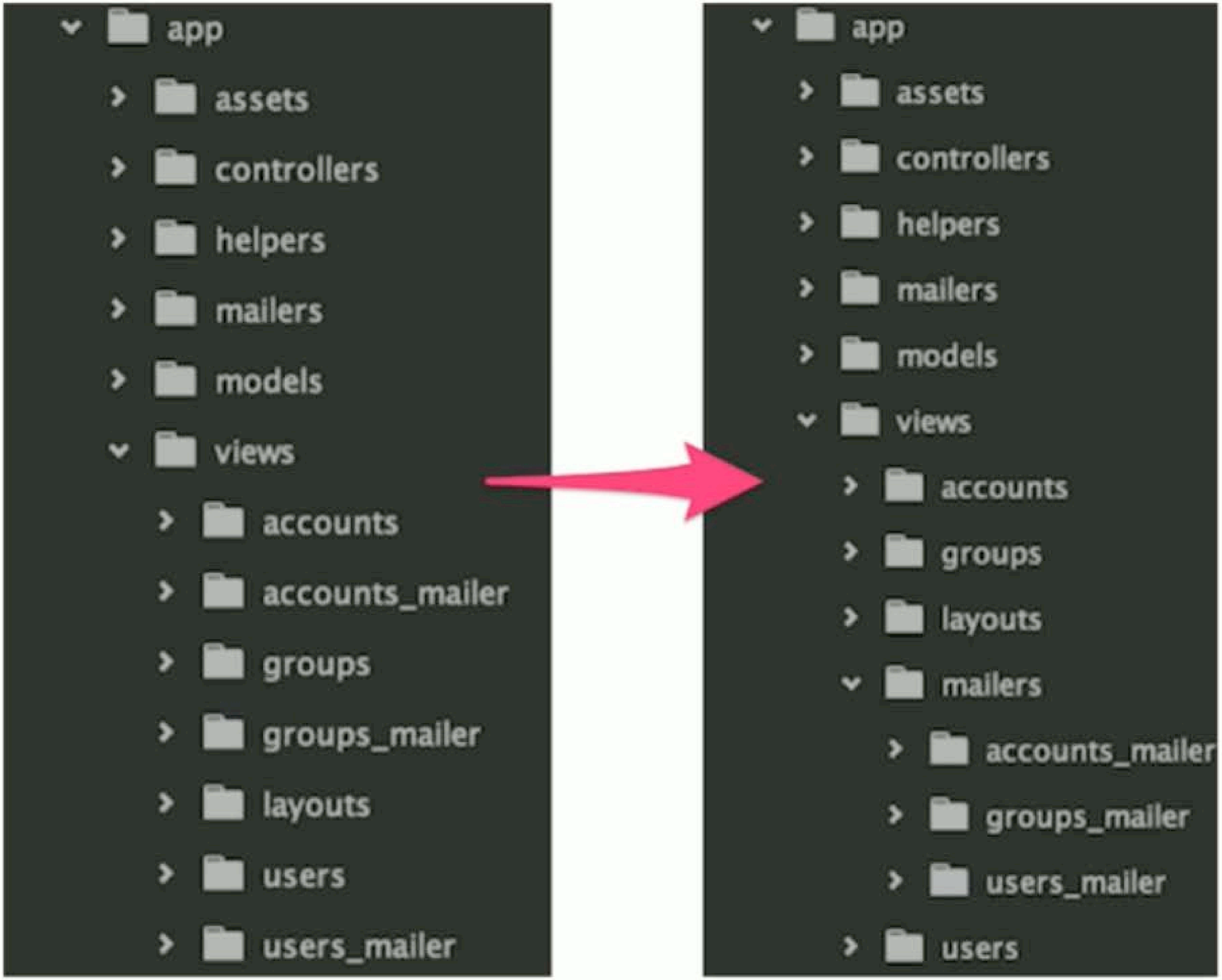
SMTP and HTTP requests will eventually fail

Why is this code a support nightmare?

```
User.receives_newsletter.find_each do |user|
  UsersMailer.newsletter(user).deliver_now
end
```

Mailers Tip #2

Group views inside app/views/mailers



```
v app
  > assets
  > controllers
  > helpers
  > mailers
  > models
  v views
    > accounts
    > accounts_mailer
    > groups
    > groups_mailer
    > layouts
    > users
    > users_mailer
```

```
v app
  > assets
  > controllers
  > helpers
  > mailers
  > models
  v views
    > accounts
    > groups
    > layouts
    v mailers
      > accounts_mailer
      > groups_mailer
      > users_mailer
    > users
```

Views Tip #1

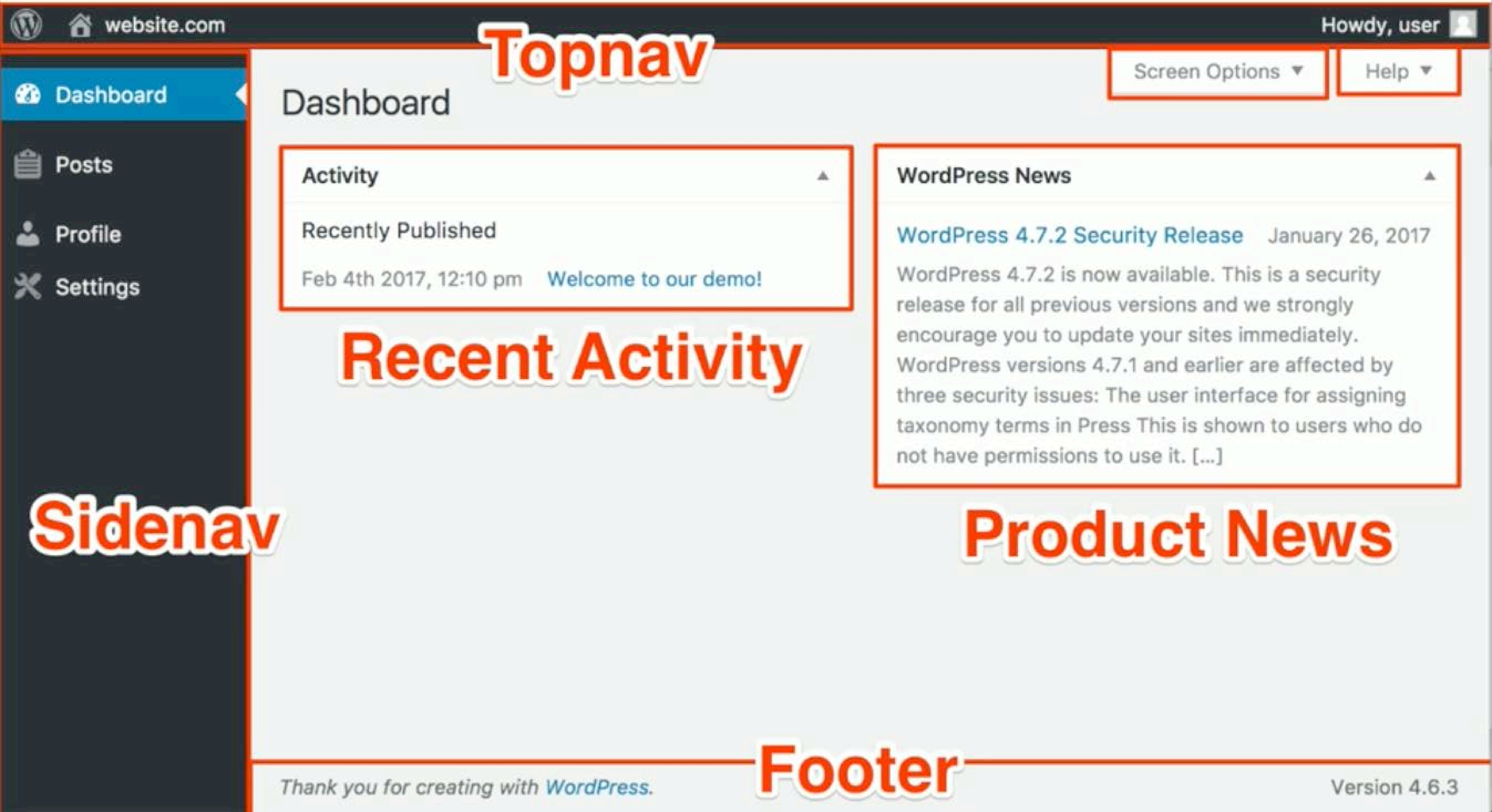
Avoid most logic

- Limit yourself to simple loops and if statements

- Place all other logic inside helpers

Views Tip #2

Break them into visual pieces



Recently Published

Feb 4th 2017, 12:10 pm Welcome to our demo!

WordPress 4.7.2 Security Release January 26, 2017

WordPress 4.7.2 is now available. This is a security release for all previous versions and we strongly encourage you to update your sites immediately. WordPress versions 4.7.1 and earlier are affected by three security issues: The user interface for assigning taxonomy terms in Press This is shown to users who do not have permissions to use it. […]

Thank you for creating with WordPress.

Version 4.6.3

Helpers Tip #1

Break helpers into view components

View components are “Plain Old Ruby Objects” (PORO) that fill in the details of rendering a visual element.

- Keeps your helper modules small
- Allows benefits of OO design

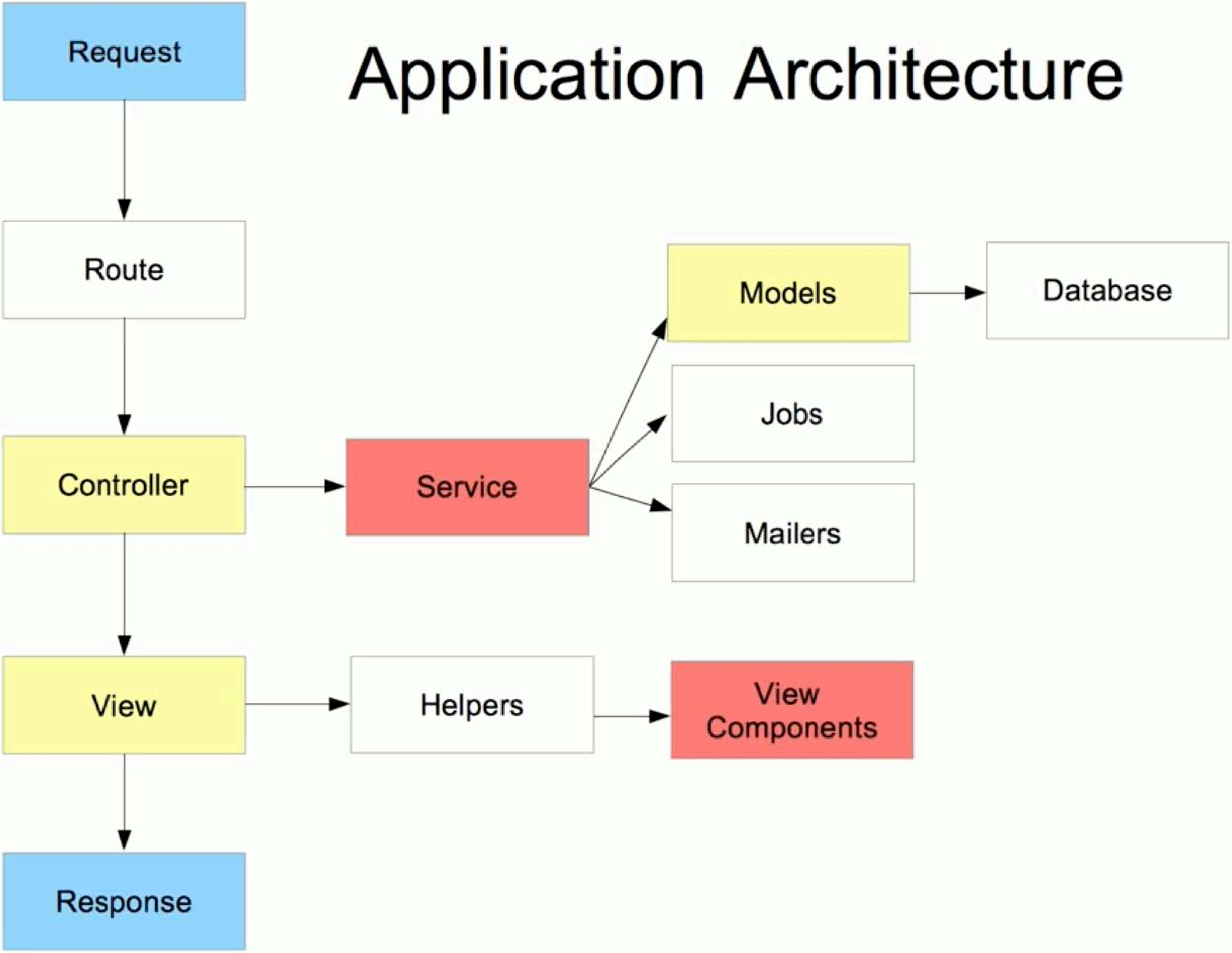
Helpers Tip #2

Do not include all helpers

```
config.action_controller.include_all_helpers = false
```

- Helps you create a solid foundation of shared modules
- Ensures most helpers only help certain views, so the method names can be shorter

Application Architecture



Request

Route

Controller

Service

Models

Database

Jobs

Mailers

View

Helpers

View Components

Response

Major Point

Plan your Rails architecture, set its convention, and never compromise

- Avoids the broken window theory
- Keeps your greenfield project feeling green

Growing Your Rails App

Photo by Ragesoss
GNU Free Documentation

Dan Cunning
http://topdan.com