Web Development with Ruby on Rails

Unit 8: Controllers and Scopes

Midterm due Monday (11/5)
at 12pm

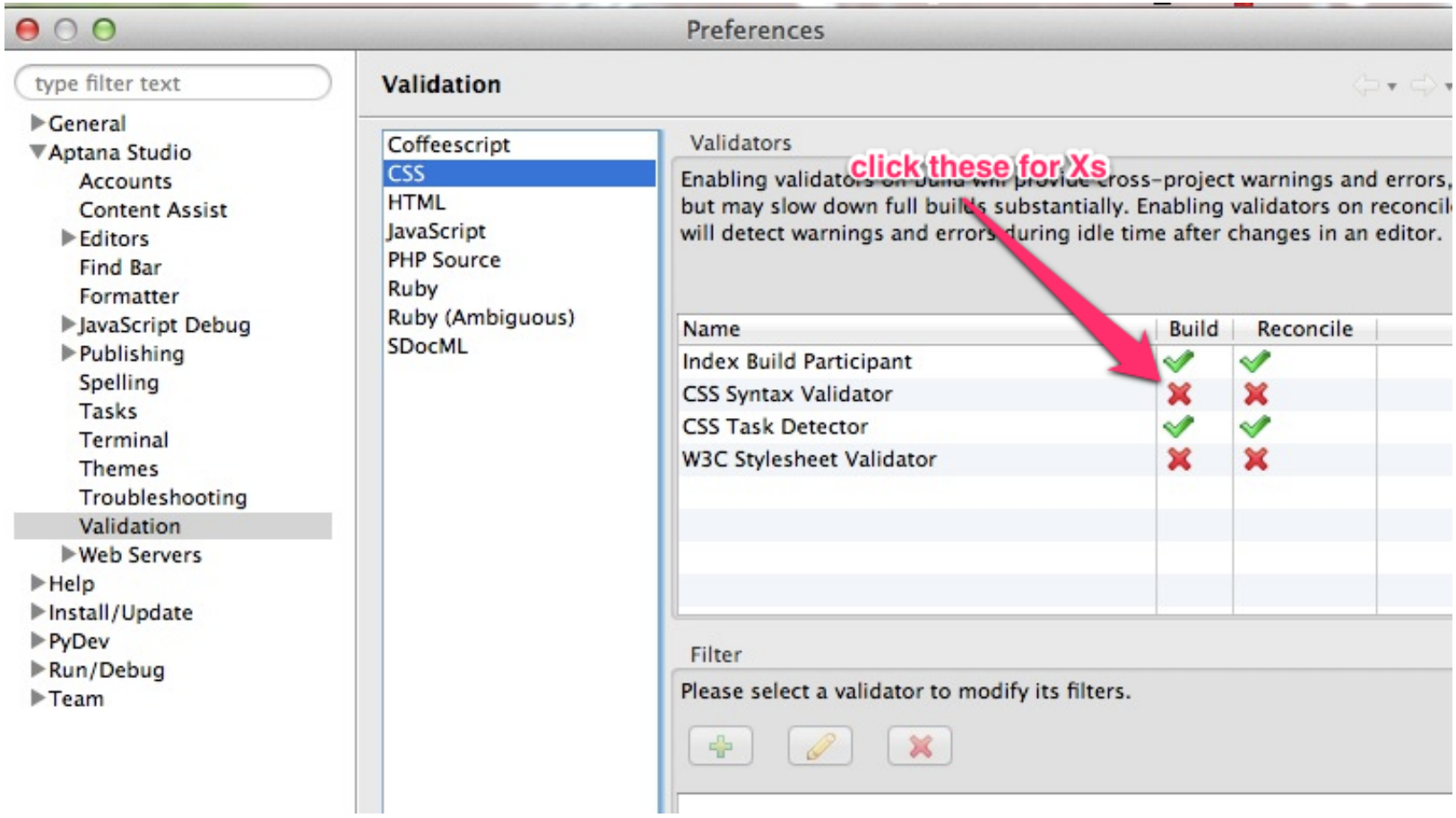
Philly.rb @ Drexel

Nov 13th

Free Pizza

phillyrb.org

FTFY!


Preferences
type filter text
General
Aptana Studio
Accounts
Content Assist
Editors
Find Bar
Formatter
JavaScript Debug
Publishing
Spelling
Tasks
Terminal
Themes
Troubleshooting
Validation
Web Servers
Help
Install/Update
PyDev
Run/Debug
Team
Validation
Coffeescript
CSS
HTML
JavaScript
PHP Source
Ruby
Ruby (Ambiguous)
SDocML
Validators
click these for Xs
Enabling validators on build will provide cross-project warnings and errors,
but may slow down full builds substantially. Enabling validators on reconcile
will detect warnings and errors during idle time after changes in an editor.
Name
Build
Reconcile
Index Build Participant
CSS Syntax Validator
CSS Task Detector
W3C Stylesheet Validator
Filter
Please select a validator to modify its filters.

More models

Basic queries

```
Pledge.first
Pledge.last
Pledge.all
```

Ordering

```
# ascending
Pledge.order('amount')

# descending
Pledge.order('amount desc')
```

Conditions

```
# Exact field match
Pledge.where(completed: true)

# Inclusion
Pledge.where(user_id: [1,2,3])
```

Conditions

```
# SQL operators with ?
Pledge.where('amount > ?', 10)

# Matching
Pledge.where('issue_url like ?',
             '%bundler%')
```

Limits and offsets

```
# Note the chaining!
Pledge.offet(50).limit(10)
```

Chains all the way down

```
Pledge.where('amount > ?', 10).
       order('amount desc').
       limit(10)
```

Increase readability with scopes

```
class Pledge
  scope :pricey, where('amount > ?', 10)
  scope :top_ten, order('amount desc').
                  limit(10)
end

Pledge.pricey.top_ten
```

Scopes work on associations too

```
current_user.pledges.pricey.top_ten
```

Not sure? Ask it.

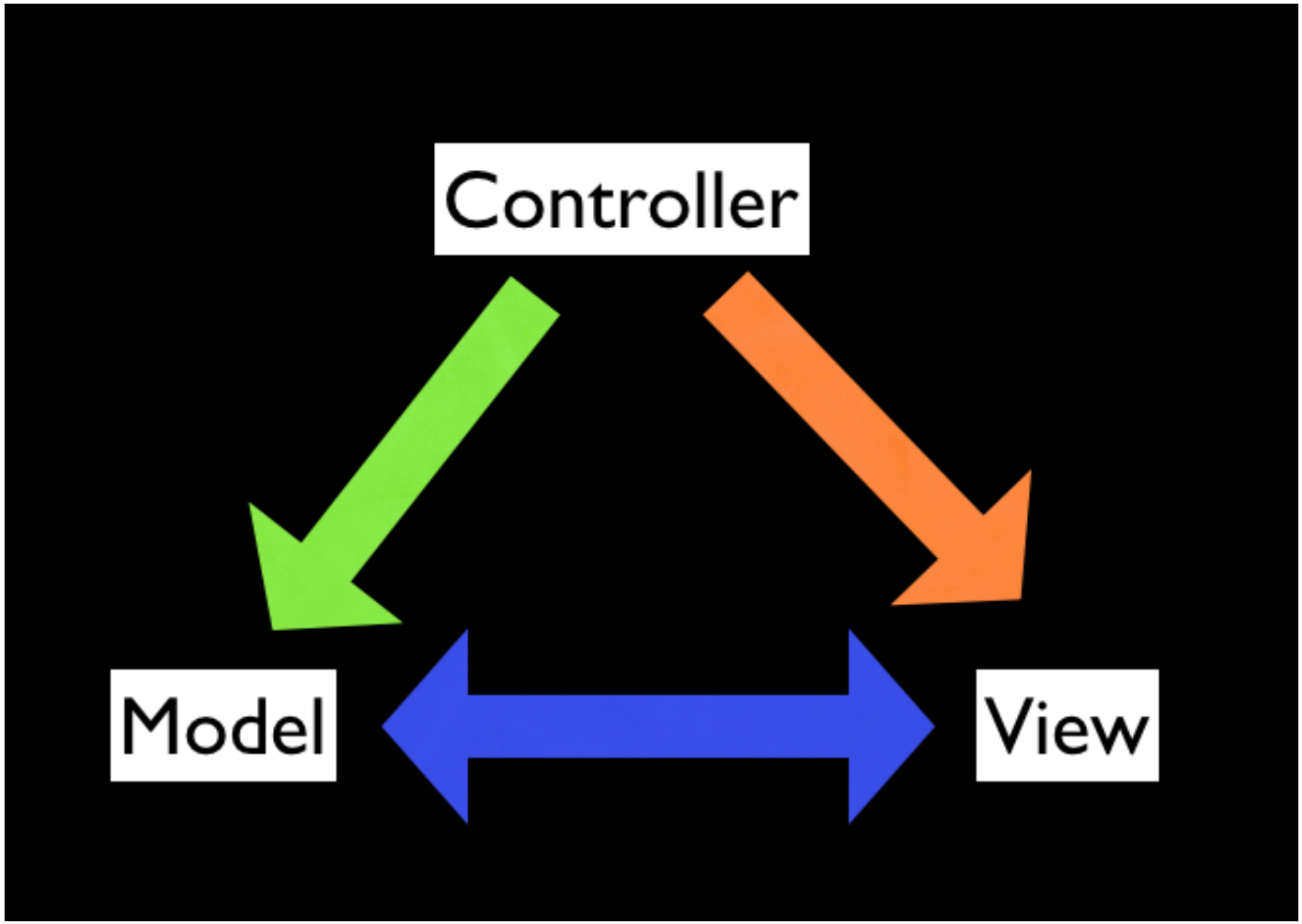
```
Pledge.pricey.top_ten.to_sql

# => "SELECT pledges.* FROM ...
```

And they're lazy

(Doesn't query DB until used)

So how can I use this to make sure people don't edit other people's pledges?

Controllers


Controller
Model
View

A controller's job

- Handle security (mostly)
- Pull a model
- Determine a view

Handling security

```
class ApplicationController
  protect_from_forgery
  before_filter :authenticate_user!
end
```

Determine Model & View

```
class PledgeController
  def index
    @pledges = current_user.pledges
    # render :index
  end
end
```

Model can have many Controllers

Controller for a user's organization

```
resources :organizations
resources :users do
  member do
    resource :organization,
      controller: 'UserOrganization'
  end
end
```

```
class UserOrganization
  def new
    @user = User.find(params[:user_id]
    @organization = @user.organizatior
                                build
  end
end
```

Filters

- before, after, around
- exit if you call `render` or `redirect_to`

Use symbols to reference method

```
before_filter :make_awesome

def make_awesome
  Awesome.create
end
```

Or use a block

```
# make some awesome
before_filter do
  Awesome.create
end
```

Rendering

```
def index
  render text: 'hello world'
end
```

Rendering

- Normally automatic
- Lots of little options

```
render :edit
render :action => :edit
render 'edit'
render 'edit.html.erb'
render :action => 'edit'
render :action => 'edit.html.erb'
render 'books/edit'
render 'books/edit.html.erb'
render :template => 'books/edit'
render :template => 'books/edit.html.erb'
render '/path/to/rails/app/views/books/edit'
render '/path/to/rails/app/views/books/edit.h
render :file => '/path/to/rails/app/views/b
render :file => '/path/to/rails/app/views/b
```

Rendering

I mainly use

```
# A quick test
render text: 'hello world'
# For re-rendering a form
render action: 'new'
# For a JSON api
render json: @some_object
```

Rendering

There's a whole guide on it

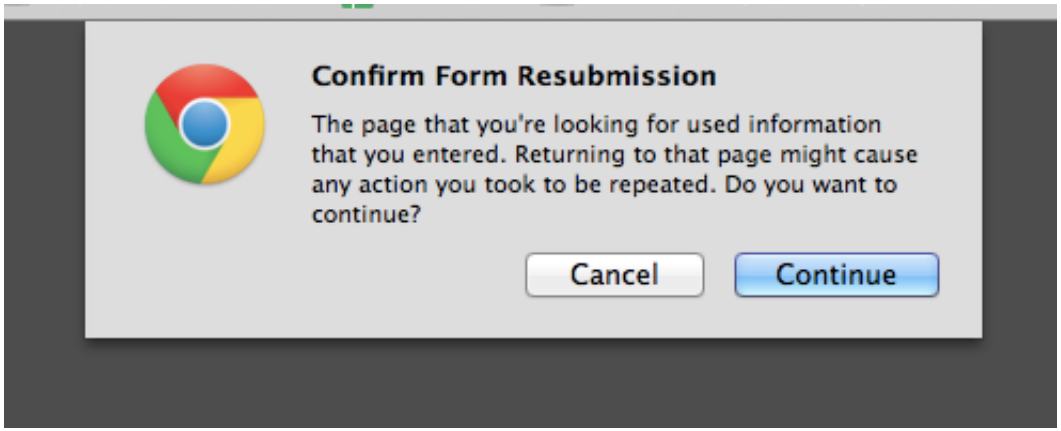
http://guides.rubyonrails.org/layouts_and_rendering.html

Redirects

```
redirect_to user_path(@user)
```

Redirects

Definitely a good idea after a form post to avoid this.



Session

A persistent hash for each browser session

app/controllers/pledges_controller.rb

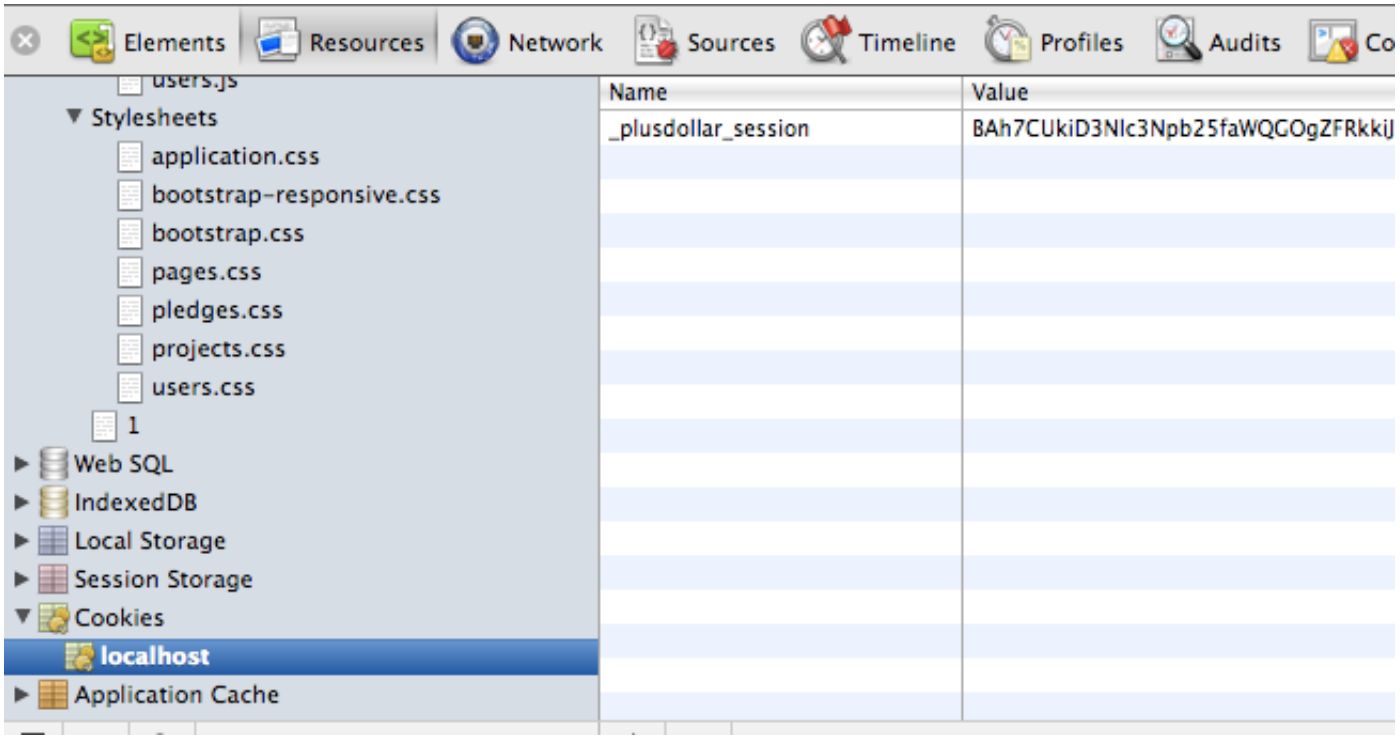
```
def index
  ...
  session['greeting'] = 'hello!'
end
```

app/views/pledges/show.html.erb

```
<p>
  Greeting: <%= session['greeting'] %>
</p>
```

Mainly for storing `user_id` or a return URL after a multi-page process.

Lives here


Elements
Resources
Network
Sources
Timeline
Profiles
Audits
Co
users.js
Stylesheets
application.css
bootstrap-responsive.css
bootstrap.css
pages.css
pledges.css
projects.css
users.css
1
Web SQL
IndexedDB
Local Storage
Session Storage
Cookies
localhost
Application Cache
Name
Value
_plusdollar_session
BAh7CUkiD3Nlc3Npb25faWQGOgZFRkkiJ

Params

- A per-request hash of user-provided data
- Uses GET query params or POST data
- Can be nested: `params[:pledge][:name]`

```ruby
def create
  @pledge = Pledge.new(params[:pledge]
  ...
end
```

Thin Controllers

(Displaying Recent Pledges)

Don't do this

```
class ApplicationController
  before_filter :set_recent_pledges
  def set_recent_pledges
    @recent = user.pledges.
        where("created_at >= ?",
              14.days.ago)
  end
end
```

Do this

```
class Pledge
  def self.recent
    where("pledges.created_at >= ?",
          14.days.ago)
  end
end
```

Then the controller can stay thin

```
class ApplicationController
  before_filter :set_recent_pledges
  def set_recent_pledges
    @recent = current_user.pledges.rec
  end
end
```

Or empty, actually

```
class ApplicationController
end

class ApplicationHelper
  def recent_pledges
    current_user.pledges.recent if
      current_user
  end
end
```

Now just use the helper where we need it

```
<%= render recent_pledges %>
```

More on controllers

http://guides.rubyonrails.org/actioncontrolleroverview.html

Next Topic: Hosted Services

Homework:

Add a scope on your models, use it in your controllers/views