

PART II

Diving into Rails source

1

- Using the ~~force~~ source
- ActiveSupport
- ActiveRecord
- ActionPack
- Railties

2

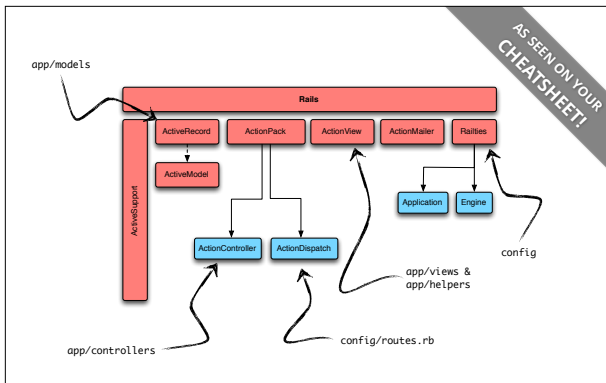
Using the
~~force~~
source

3

Gems

all the way down

4



5

```
or { $ export EDITOR="subl -n"  
    $ export EDITOR="vim"  
    $ set EDITOR=start Windows  
  
$ bundle open <GEM NAME>
```

6

Active Support

7

HOW TO BROWSE CODE

8

1. Start reading file.
2. Get overwhelmed.
3. Curse.
4. Claim you could do better *(optional)*.
5. Back to step 1 or give up.

HOW TO BROWSE CODE

OWNING TIP

9

1. Take a deep breath.
2. Start with an assumption.
3. Follow the flow. *(Don't read top to bottom.)*
4. Don't try to understand everything.

EXERCISE

Refactor our Filters module using ActiveSupport::Concern.

```
(in patterns)
$ git pull
$ git checkout -b concern
$ edit lib/filters.rb
# To test
$ ruby -Itest test/filters_test.rb
```

Bonus: use ActiveSupport::Callbacks too.

HINT

```
$ bundle open activesupport
```

10

SAVE YOUR WORK

```
$ git add .
$ git commit -m "Refactoring filters w/ AS"

# Get my changes
$ git checkout master
$ git pull
```

11

```
class User < ActiveRecord::Base
  validates :token, :uniqueness => true
  def generate_token
    # ...
  end
  def self.authenticate(login, password)
    # ...
  end
end

module TokenSupport
  extend ActiveSupport::Concern
  included do
    validates :token, :uniqueness => true
  end
  def generate_token
    # ...
  end
end

module Authentication
  extend ActiveSupport::Concern
  module ClassMethods
    def authenticate(login, password)
      # ...
    end
  end
end

class User < ActiveRecord::Base
  include Authentication
  include TokenSupport
end
```

OWNING TIP

12

13

The Basics

of browsing Rails code

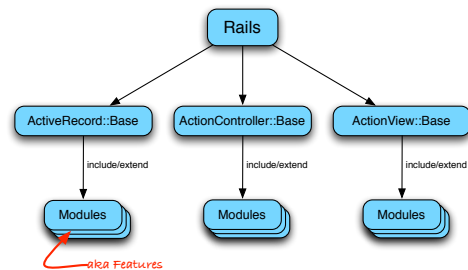
14

```
class User < ActiveRecord::Base
  # ...
end

class ApplicationController < ActionController::Base
  # ...
end

<div class="user">
  <%= self.class.superclass %> == ActionView::Base
</div>
```

15



Active Record

16

17

```
module ActiveRecord
  class Base
    include Persistence
    include Scoping
    include Sanitization
    include AttributeAssignment
    # ...
  end
end
```

Where's the code?

Model logic that requires DB access.	ActiveRecord	bundle open activerecord
Model logic that doesn't require DB access.	ActiveModel	bundle open activemodel

AS SEEN ON YOUR
CHEATSHEET!

18

EXERCISE

Locate the validation code.

```
$ git clone git://github.com/owningrails/blog.git
$ cd blog
$ bundle open activerecord
$ bundle open activemodel
```

Bonus: add validation to our ActiveRecord implementation.

```
(in patterns)
$ git pull
$ git checkout -b validation
$ edit app/models/user.rb
$ ruby -Itest test/user_test.rb
```

```
class User < ActiveRecord::Base
  validates :name, :presence => true
end

# Test
def test_valid
  user = User.new
  assert ! user.valid?
  assert_equal ["can't be blank"], user.errors[:name]
end
```

19

SAVE YOUR WORK

```
$ git add .
$ git commit -m "Implement validation"

# Get my changes
$ git checkout master
$ git pull
```

20

```
class Invitation
  include ActiveRecord::Validations

  attr_accessor :from, :to

  validates_presence_of :from, :to

  def initialize(from, to)
    @from = from
    @to = to
  end
end
```

OWNING TIP

21

Action Dispatch

22

The ~~magical~~ path of a request in Rails

GET /users/new



```
class UsersController < ApplicationController
  def new
    # What! How did you get here?
  end
end
```

23

```
class Rails::Application
  def call(env)
    [
      200,
      { 'Content-Type' => 'text/plain' },
      "you requested " + env['PATH_INFO']
    ]
  end
end
```

24

25

config.ru

```
# This file is used by Rack-based servers to start the application.

require ::File.expand_path('../config/environment', __FILE__)
run Rails.application
```

26

patterns/lib/application.rb

```
class Application
  def call(env)
    request = Rack::Request.new(env)
    response = Rack::Response.new

    controller_name, action_name = route(request.path_info)

    controller_class = load_controller_class(controller_name)
    controller = controller_class.new # HomeController.new
    controller.request = request
    controller.response = response
    controller.process(action_name)

    response.finish
  end
end
```

+



rack middlewares
powers web applications

27

```
class Logger
  def initialize(app)
    @app = app
  end

  def call(env)
    puts "Calling " + env["PATH_INFO"]
    @app.call(env)
  end
end

use Logger

run App.new
```

28

```
use ActionDispatch::Static if config.serve_static_assets
use Rack::Lock if !config.allow_concurrency
use Rack::Runtime
use Rails::Rack::Logger
use ActionDispatch::ShowExceptions, config.consider_all_requests_local
use ActionDispatch::RemoteIp
use Rack::Sendfile, config.action_dispatch.x_sendfile_header
use ActionDispatch::Callbacks
use ActionDispatch::Cookies

use config.session_store, config.session_options
use ActionDispatch::Flash

use ActionDispatch::ParamsParser
use Rack::MethodOverride
use ActionDispatch::Head
use ActionDispatch::BestStandardsSupport

run ActionDispatch::Routing::RouteSet.new
```

29

EXERCISE

Replace Rails logger with a custom middleware.

```
(in blog)
$ bundle install
```

```
$ git checkout -b middleware
```

```
$ edit app/middlewares/custom_logger.rb
$ edit config/application.rb
$ rails server
```

Hint:

```
class DummyMiddleware
  def initialize(app)
    @app = app
  end

  def call(env)
    @app.call(env)
  end
end

# in config/application.rb
config.middleware.swap "Rails::Logger", "CustomLogger"
```

Bonus: hook a Sinatra app inside the Rails app under `/sinatra` using a middleware.

Hint: read & modify `PATH_INFO`.

```
class MySinatraApp < Sinatra::Base
  get "/" do
    "Hello from Sinatra"
  end
end
```

To call it:

```
MySinatraApp.call(env)
```

30

SAVE YOUR WORK

```
$ git add .
$ git commit -m "Add custom middlewares"

# Get my changes
$ git checkout master
$ git pull
```

OWNING TIP

```
match '/sinatra' => MySinatraApp, anchor: false
```

Remove /sinatra from
PATH_INFO & set
SCRIPT_NAME

31

Where's the code?

Code handling the processing & routing of a request	ActionDispatch	bundle open actionpack
Features you get access to inside your controllers	ApiController	bundle open actionpack
Controller code not related to web	AbstractController	bundle open actionpack

AS SEEN ON YOUR
CHEATSHEET!

32

Action
Controller

33

34

```
class ApplicationController < ActionController::Base
end
```

35

```
module ActionController
  class Metal
    attr_accessor :request, :response

    def process(action_name)
      send action_name
    end
  end

  class Base < Metal
    include Filters
    include Rendering
  end
end
```

36

```
module Filters
  extend ActiveSupport::Concern

  def process(action_name)
    # ...
    super
  end
end
```

HOW TO FIND A METHOD

OWNING TIP

37



1. Guess.
2. Search “*def render*”.
3. `method(:render).source_location`

EXERCISE

Locate the method behind `respond_with` that replaces the default scaffolding code.

```
$ bundle open actionpack
```

Bonus: override the code behind `respond_with` to display flash messages after successful update and create.

```
(in blog)
$ git pull
$ bundle exec rake test:prepare
$ git checkout -b respond
$ ruby -Itest test/controllers/
posts_controller_test.rb
```

```
# To replace the default responder
class ApplicationController < ActionController::Base
  self.responder = CustomResponder
end

class CustomResponder < ActionController::Responder
  # ...
end
```

38

SAVE YOUR WORK

```
$ git add .
$ git commit -m "Override responder code"

# Get my changes
$ git checkout master
$ git pull
```


39

Action View

40

HELPERS

```
module ActionView
  module Helpers
    # ...
    include DateHelper
    include FormHelper
    include NumberHelper
    include SanitizeHelper
    include TagHelper
    include TextHelper
    include UrlHelper
    # ...
  end
end
```

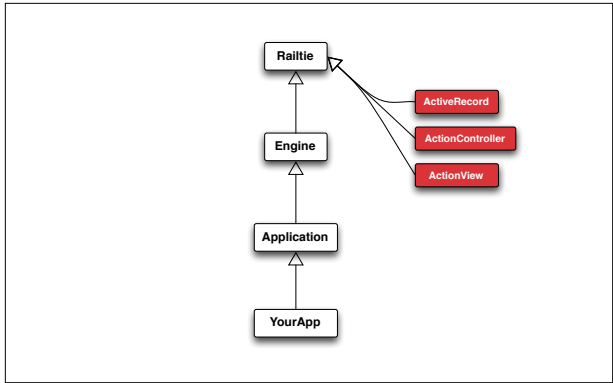


```
module ActionView
  class Base
    include Helpers
    # ...
  end
end
```

41

Railties

42



43

Where's the code?		
Model logic that requires DB access	ActiveRecord	bundle open activerecord
Model logic that doesn't require DB access	ActiveModel	bundle open activemodel
Code handling the processing & routing of a request	ActionDispatch	bundle open actionpack
Features you get access to inside your controllers	ActionController	bundle open actionpack
View and helpers code	ActionView	bundle open actionview
Framework code, holding all the components together	Rails	bundle open railties

44

Thanks!

... and final Q&A

<http://j.mp/owningrails-feedback>

45
