THE PLAN

Get the slides & cheat sheets @ http://j.mp/owningrails-kit

Part I - The Architecture

- Ruby: The Language
- The design patterns behind Rails
  - Active Record
  - Front Controller & Rack
  - Intercepting Filter
  - Template View
  - ~~Conquer the Internets, TWICE~~

Part II - The Rails source

- Browsing Gems code
- Browsing Rails code
  - ActiveSupport
  - ActiveRecord
  - ActionPack
  - Railties

Exercises / Breaks / Q&A

~1/hour, 10-20min each

JOIN THE IRC CHANNEL

#codedclasses on freenode

http://webchat.freenode.net/

To ask a question:

macournoyer: ZOMGBBQ, why is this class so awesome?

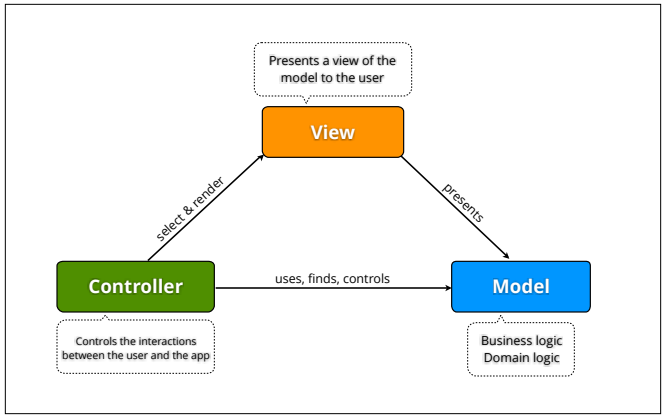
PART I

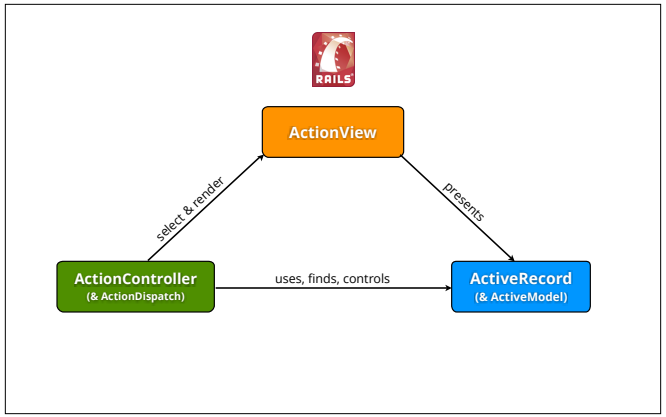
Rails Architecture

Why  ?

MVC

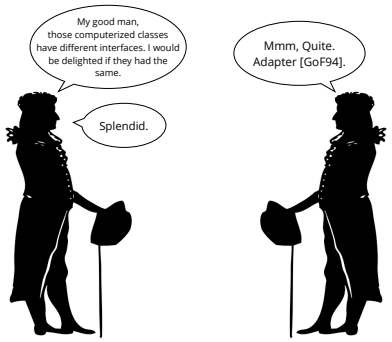
Model - View - Controller



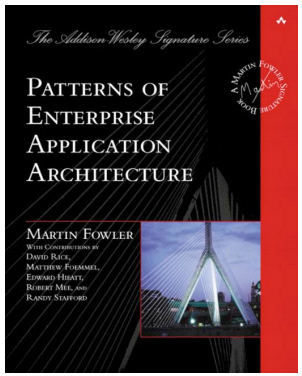


DESIGN PATTERNS

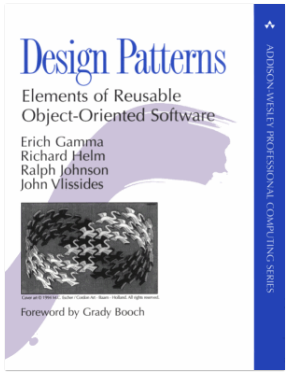
Most software problems have been solved



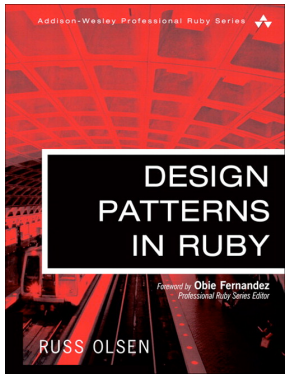
http://clipshay.com/wp/stock-man-silhouette-shadow-33439/



http://www.amazon.com/gp/product/0321127420



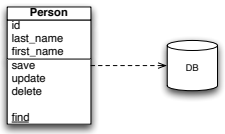
http://www.amazon.com/Design-Patterns-Elements-Reusable-Object-Oriented/dp/0201633612



http://amzn.com/0321490452

Active Record

From PEAA, Fowler 2002



Let's code

EXERCISE

Implement User.all

```
$ git clone git://github.com/owningrails/patterns.git
$ cd patterns
$ git checkout -b user-all
$ edit lib/active_record.rb
$ edit test/user_test.rb # Add test for User.all

# To test
$ ruby -I. test/user_test.rb
```

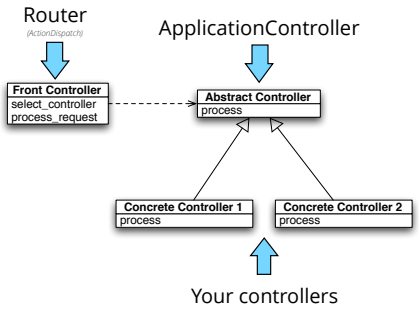
SAVE YOUR WORK

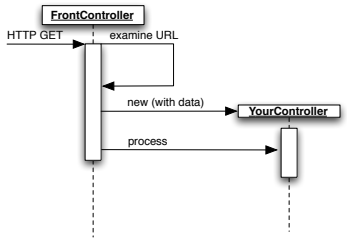
```
$ git add .
$ git commit -m "Implement AR.all"

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

Front Controller

From PEAA, Fowler 2002





Let’s code

(

Web Server

Thin

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

Request info Hash

Status code

Header

Body

```
POST / HTTP/1.1
Host: localhost
Connection: close
Content-Length: 6

ohaie!
```

```ruby
env = {
 "REQUEST_METHOD"  => "POST",
 "PATH_INFO"       => "/",
 "HTTP_VERSION"    => "1.1",
 "HTTP_HOST"       => "localhost",
 "HTTP_CONNECTION" => "close",
 "CONTENT_LENGTH"  => "6"
}
```

```ruby
[
  200,
  {
    "Content-Length" => "34",
    "Content-Type" => "text/html"
  },
  [
    "<html>",
    "  <h1>kthxbaie</h1>",
    "</html>"
  ]
]
```

```
HTTP/1.1 200 OK
Content-Length: 34
Content-Type: text/html

<html>
 <h1>kthxbaie</h1>
</html>
```

```
class _____

  # Read data sent by browser
1 data = socket.read

  # Parse the data to a Hash
2 env = parse(data)

  # Call the app
3 status, headers, body = app.call(env)

  # Send the response to the browser
4 socket.write "HTTP/1.1 #{status} ..."

end
```

config.ru

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

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

EXERCISE

Create a Rack app that outputs the requested path.
(Hint: inspect env variable)

```
$ git pull
$ gem install shotgun
$ git checkout -b rack_app
$ edit config.ru
$ shotgun
# Open your browser at
# http://localhost:9393/hi
# Should show: “/hi”
```

Bonus: Implement a tiny subset of Sinatra as a Rack app.

Just enough to run this:

```
get "/hi" do
  "Owning!"
end
```

Hint: use PATH_INFO and REQUEST_METHOD

Windows users: use rackup instead of shotgun and restart it (CTRL+C) on each change

SAVE YOUR WORK

```
$ git add .
$ git commit -m "Create a Rack app"

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

)

Front Controller

From PEAA, Fowler 2002

Let's code

EXERCISE

Create a UsersController, displaying a list of users (index).

```
# in patterns
$ git pull
$ git checkout -b users_controller
$ shotgun
# open http://localhost:9393/users
```

Bonus: build a Rails-like routing DSL to match the routes we have now.

config/routes.rb

```
Routes = Router.new do
  match '/' => 'home#index'
  match '/home/index' => 'home#index'
  match '/users' => 'users#index'
end
```

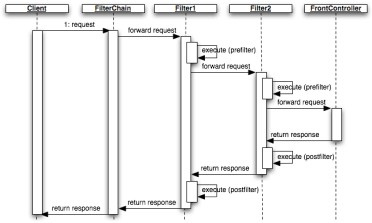
SAVE YOUR WORK

```
$ git add .
$ git commit -m "Create a controller"

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

Intercepting Filter

From Alur et al.



http://www.bonframework.org/j2ifilter-1

```ruby
class UserController < ApplicationController
  before_action :login_required
end
```

```ruby
class User < ActiveRecord::Base
  before_save :activate!
end
```

Let's code

EXERCISE

Create an after action to append a footer

```
$ git pull
$ git checkout -b after_action
$ shotgun
# Run tests
ruby -Itest test/filters_test.rb
```

app/controllers/home_controller.rb

```ruby
class HomeController < ApplicationController
  after_action :footer

  # ...

  def footer
    response.write "&copy; me"
  end
end
```

SAVE YOUR WORK

```
$ git add .
$ git commit -m "Add filters to controllers"

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

Template View

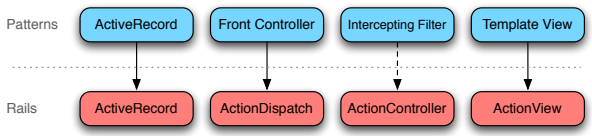
From PEAA, Fowler 2002

```
<h1><%= @title %></h1>
<ul>
<% @users.each do |user| %>
  <li><%= user.name %></li>
<% end %>
</ul>
```

Let's code

PART I

Recap



Get ready for tomorrow

Clone my Rails app

```
$ cd .. # if still in patterns dir
$ git clone git://github.com/owningrails/blog.git
$ cd blog
$ bundle install
$ bundle exec rake db:setup
```