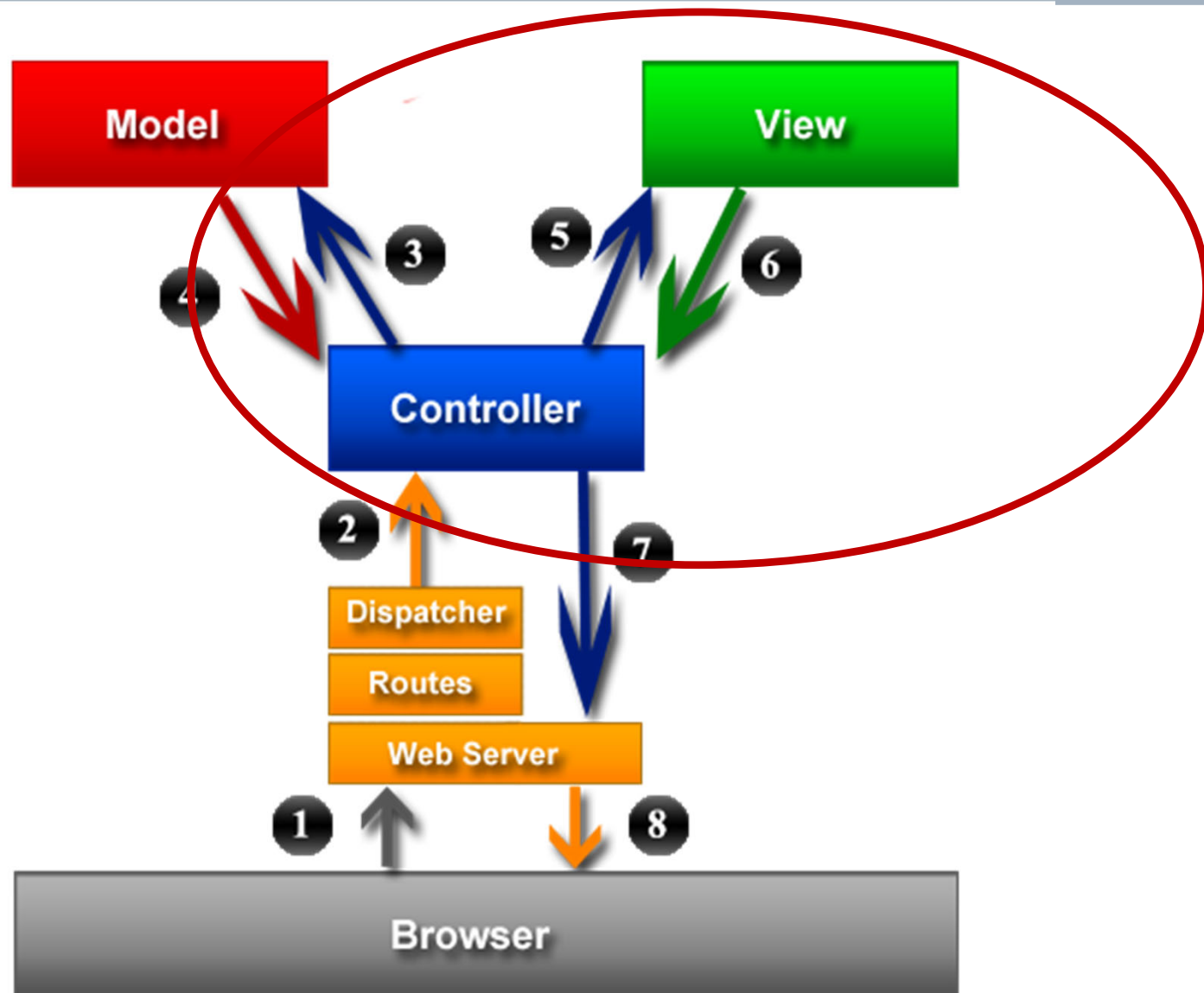


Rails: Views and Controllers

Computer Science and Engineering ■ College of Engineering ■ The Ohio State University

Lecture 31

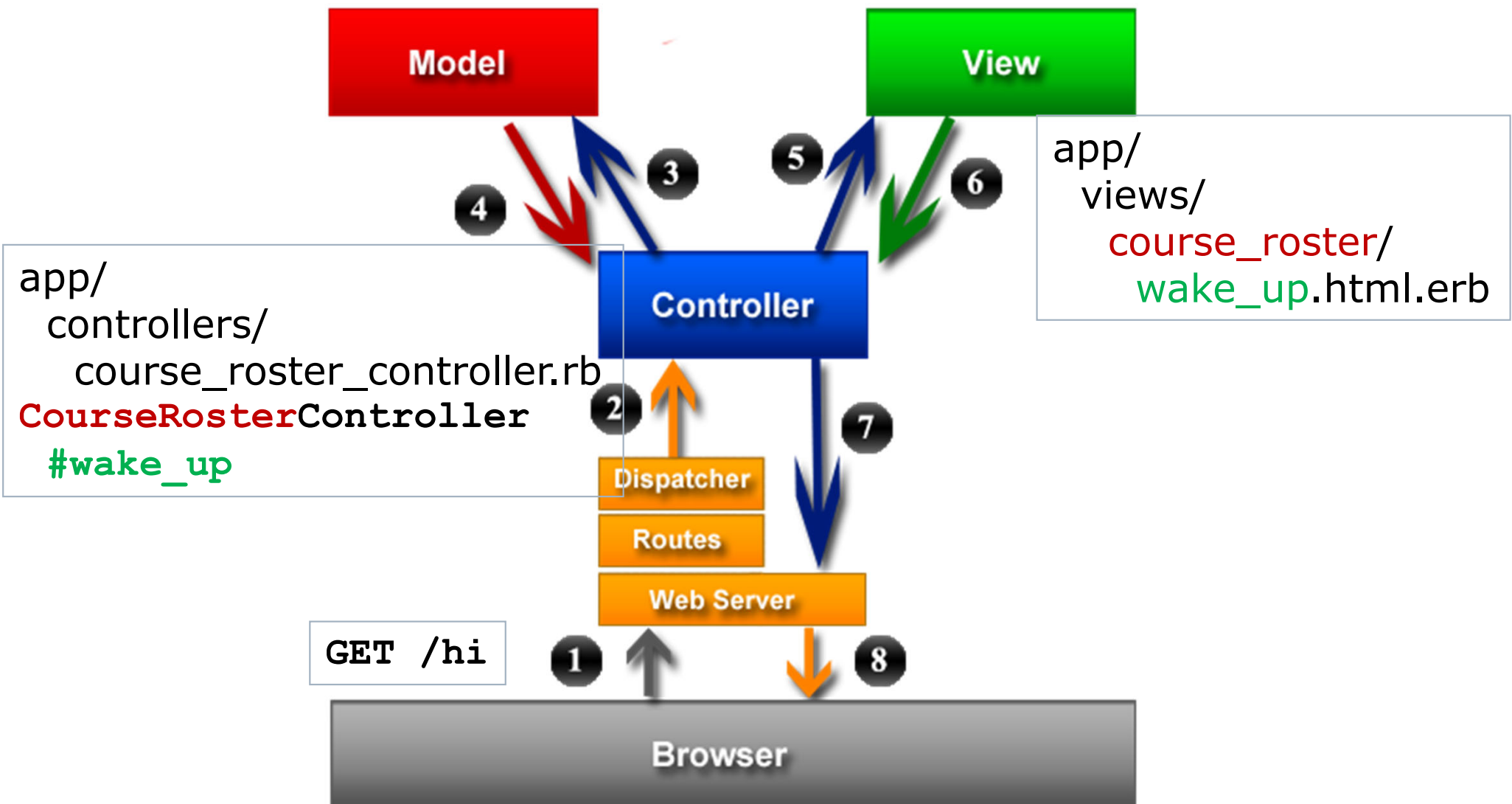
Recall: Rails Architecture



Wiring Views and Controllers

- A controller is just an ordinary Ruby class
 - Extends `ApplicationController`
`class CourseRosterController <`
 `ApplicationController`
 - Location: `app/controllers/`
 - Filename: `course_roster_controller.rb`
- Actions are methods in that class
 `def wake_up`
 `...`
 `end`
- A view is an HTML page (kind of) that corresponds to that action
 - Location: `app/views/course_roster/`
 - Filename: `wake_up.html.erb`
 - Has access to *instance* variables (e.g., `@student`) of corresponding controller!

Recall: Rails Architecture



Demo: Building From Scratch

```
$ rails new demo
```

□ Create CourseRosterController

■ Location: app/controllers

```
class CourseRosterController <
    ApplicationController
```

■ Create (empty) method `wake_up`

□ Add route to config/routes.rb

```
get 'hi', to: 'course_roster#wake_up'
```

□ Create view (wake_up.html.erb)

■ Location: app/views/course_roster

```
$ rails server
```

Example: Controller

```
# in app/controllers/  
# filename course_roster_controller.rb  
  
class CourseRosterController <  
  ApplicationController  
  def wake_up  
    # for this simple eg, no code needed  
  end  
end
```

Example: Route Definition

```
# in config/  
# filename routes.rb
```

```
Rails.application.routes.draw do  
  get 'hi', to: 'course_roster#wake_up'  
  # equivalent to (but shorter than):  
  #   match 'hi', to: 'course_roster#wake_up',  
  #       via: [:get]  
end
```

Example: View

```
<!-- in app/views/course_roster/  
      filename wake_up.html.erb -->
```

```
<h1>Yo!!</h1>
```

```
<p>Are you awake?</p>
```


Single Point of Control

- Notice the duplication in names
- Controller name (**course_roster**) used in:
 - Name of the controller class
 - Filename of controller class implementation
 - Route
 - Directory name containing views
- Action name (**wake_up**) used in:
 - Name of the method within controller class
 - Route
 - Filename of view source
- "Solution": generate all these parts

```
$ rails g controller course_roster  
wake_up
```

Demo: Generating A Controller

```
$ rails generate controller prof ask  
visit_office
```

□ Results in:

- Addition of new routes to config/routes.rb
get '**prof**/**ask**'
- Creation of **ProfController** class
app/controllers/**prof**_controller.rb
- Definition of methods in **ProfController**
def **ask** ... end
def visit_office ... end
- Creation of 2 views (*i.e.* one per action)
app/views/**prof**/**ask**.html.erb
app/views/**prof**/visit_office.html.erb

```
$ rails server
```

ERb: Embedded Ruby

- General templating mechanism
 - “Template” = a string (usually contents of some file)
 - Contains (escaped) bits of ruby
 - `<% code %>` execute ruby code (“scriptlet”)
 - `<%= expr %>` replace with result of ruby expr
 - `<%# text %>` ignore (a comment)
- Example: a text file

```
This is some text.  
<% 5.times do %>  
Current Time is <%= Time.now %>!  
<% end %>
```
- Process using erb tool to generate result

```
$ erb example.txt.erb > example.txt
```
- Naming convention: *filename.outputlang.erb*
 - Example `index.html.erb`
- Many alternatives, eg HAML

Example: prof/visit_office View

```
<h1>Office</h1>
```

```
<p>
```

```
Hello! Welcome to <%= @location %>.
```

```
</p>
```

```
<p>
```

```
If you have a question, ask it
```

```
  <a href="prof/ask">here</a>.
```

```
Current time: <%= Time.now %>
```

```
</p>
```

Example: prof/visit_office View

```
<h1>Office</h1>
```

```
<p>
```

```
  Hello! Welcome to <%= @location %>.
```

```
</p>
```

```
<p>
```

```
  If you have a question, ask it  
  <a href="/prof/ask">here</a>.
```

```
  Current time: <%= Time.now %>
```

```
</p>
```

Example: Prof Controller

```
def visit_office
  @location = 'Dreese 695'
end
```

Example: Helpers in Views

```
<h1>Office</h1>
```

```
<p>
```

```
  Hello! Welcome to <%= @location %>.
```

```
</p>
```

```
<p>
```

```
  If you have a question, ask it
```

```
  <%= link_to "here", controller: "prof",  
                                action: "ask" %> or
```

```
  <%= link_to "here", ask_path %>.
```

```
  Current time: <%= Time.now %>
```

```
</p>
```

Solution: Layouts

- HTML formed from: **Layout** + **Template**
 - Layout is the common structure of HTML pages
 - Layout uses `yield` to include (page-specific) template

- File: **layout.erb**

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title> ... etc
  </head>
  <body>
    <%= partial "navigation" %>
    <%= yield %>
    <%= partial "footer" %>
  </body>
</html>
```

- Layout is where you put site-wide styling
 - e.g., navigation bar, div's with CSS classes, footers

Defining and Choosing Layouts

- Default layout for responding to action in **ProfController**
 - `app/views/layouts/prof.html.erb`
 - If not found, then use `app/views/layouts/application.html.erb`
- Or controller can explicitly name layout

```
class ProfController < ApplicationController
  layout "people/snazzy"
  # layout "people/snazzy", except: [:show]
```
- There is an application-wide controller that can also specify a fall-back layout

```
class ApplicationController <
  ActionController::Base
  layout "main"
```

Parameterizing the Layout

- In the layout use yield with args

```
<head>
```

```
  <title><%= yield(:title) %></title>
```

```
  <meta charset="utf-8">
```

```
...
```

- In the view, pass in argument with provide

```
<% provide(:title, "Office") %>
```

```
<h1>The Office</h1>
```

Demo With HTTP Parameters

□ Pass argument to action ask

■ Add a segment to the route

```
get 'prof/ask/:msg', # or prof/ask/(:msg)  
  to: 'prof#ask'
```

■ Change ask to access params

```
def ask  
  @q = params[:msg]  
end
```

■ Use instance variable in view

```
<p>You said: <%= @q %>!</p>
```

Summary

- View/Controller coupling
 - Location of view from name of controller
 - Filename of view from name of action
 - Controller instance variables available in the view
 - HTTP parameters available in the controller via params hash
- ERb
 - Template for generating HTML
 - Parameterized with instance variables
 - Layouts