



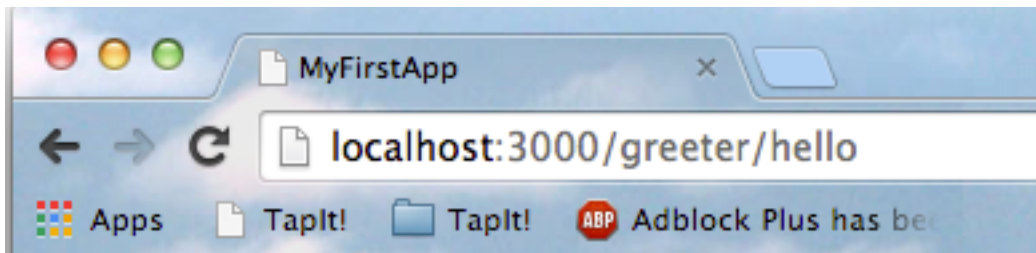
Introduction to Ruby on Rails

Part III

Action methods inside Controller

- If the action method is not really doing anything (i.e. empty) - we can remove it
- As long as there is a proper route defined and there is a properly named view file/template - the action method does not have to be there and Rails will use a convention to find the correct template

Controller - new look



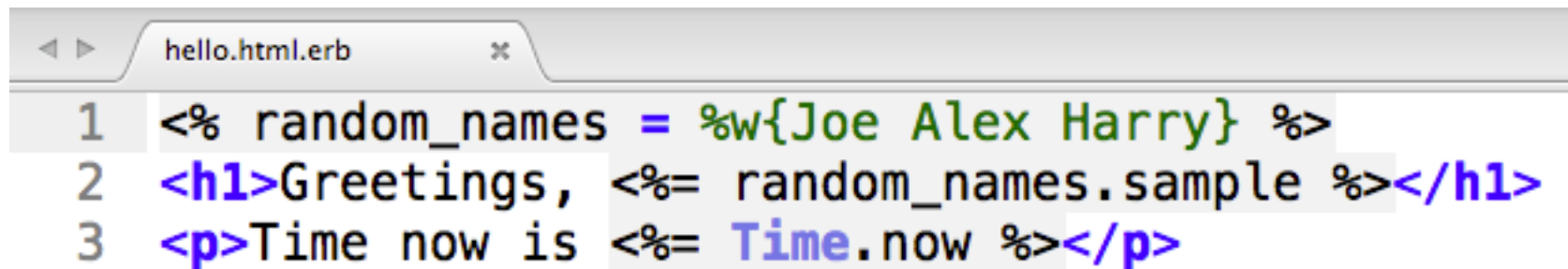
Greetings, Joe

Time now is 2014-06-01 10:00:31 -0400

You probably still want to leave the actions (methods) in the Controller (to make it easier on another person looking for them), but the point of this slide is that you don't **have** to have the methods if they are empty as long as the routes are properly defined and the views exist

Moving business logic out

- Our app “works”, but business logic does not belong in the view. The view should have as little Ruby code as possible



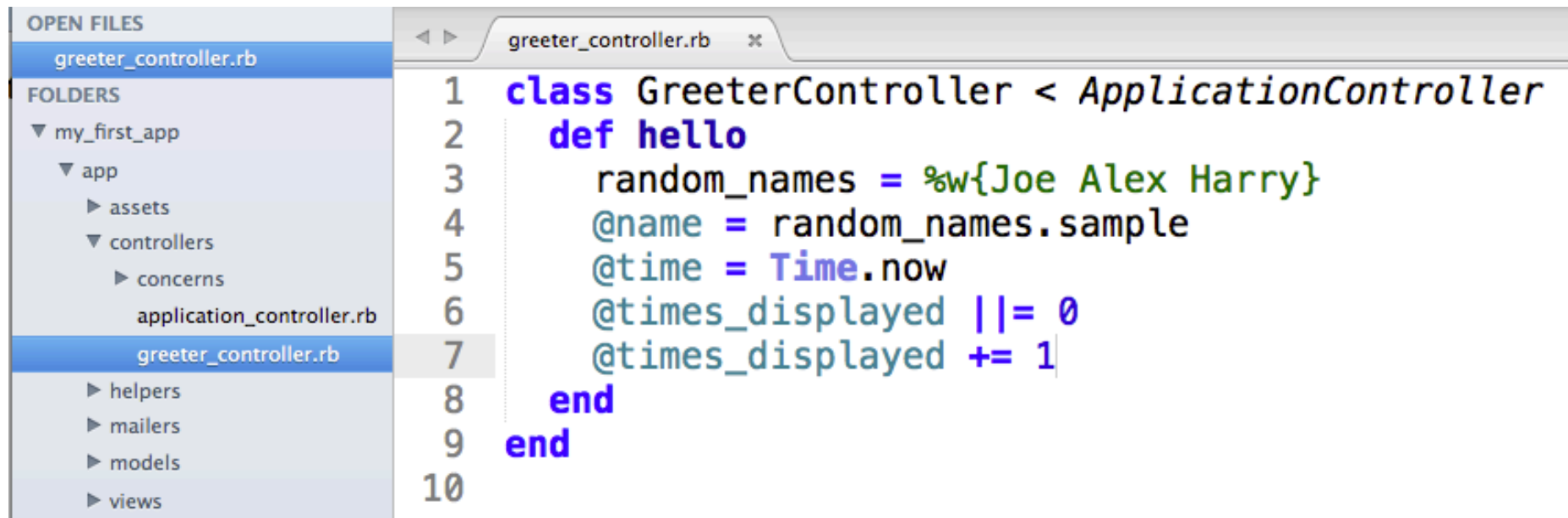
A screenshot of a code editor window titled 'hello.html.erb'. The editor shows three lines of code: line 1 is a Ruby assignment `<% random_names = %w{Joe Alex Harry} %>`; line 2 is an HTML `<h1>` tag containing a Ruby interpolation `<%= random_names.sample %>`; and line 3 is an HTML `<p>` tag containing a Ruby interpolation `<%= Time.now %>`. The code is color-coded: Ruby strings are green, HTML tags are blue, and Ruby symbols/keywords are purple.

```
1 <% random_names = %w{Joe Alex Harry} %>
2 <h1>Greetings, <%= random_names.sample %></h1>
3 <p>Time now is <%= Time.now %></p>
```

- Let's move some code to the controller!

Moving business logic out (Cont.)

- Instance variables from the controller are available inside the view



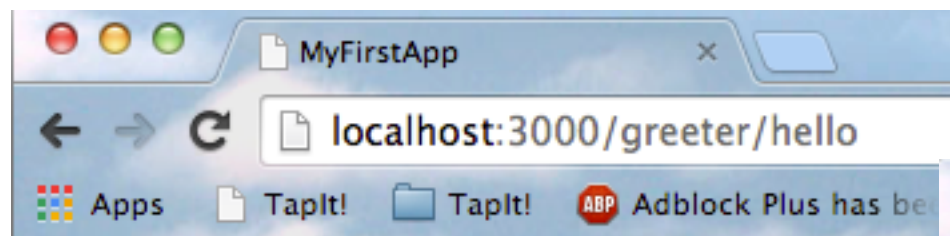
```
OPEN FILES
greeter_controller.rb

FOLDERS
▼ my_first_app
  ▼ app
    ► assets
    ▼ controllers
      ► concerns
        application_controller.rb
      greeter_controller.rb
    ► helpers
    ► mailers
    ► models
    ► views

1 class GreeterController < ApplicationController
2   def hello
3     random_names = %w{Joe Alex Harry}
4     @name = random_names.sample
5     @time = Time.now
6     @times_displayed ||= 0
7     @times_displayed += 1
8   end
9 end
10
```

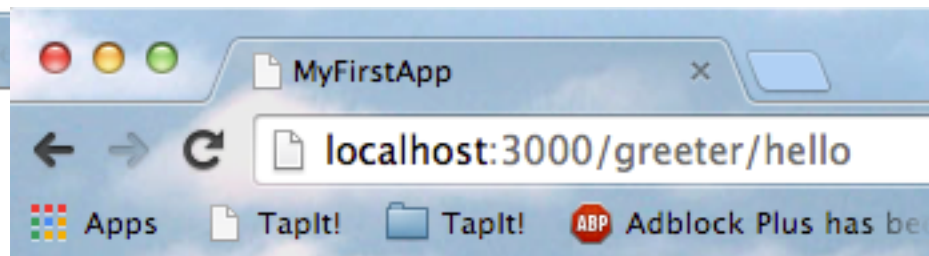
New views/greeter/hello.html.erb

```
1 <h1>Greetings, <%= @name %></h1>  
2 Time now is <%= @time %><br/>  
3 This page has been viewed <%= @times_displayed %> time(s)
```



Greetings, Joe

Time now is 2014-06-01 10:21:36 -0400
This page has been viewed 1 time(s)



Greetings, Alex

Time now is 2014-06-01 10:22:39 -0400
This page has been viewed 1 time(s)

Instance variables in Rails

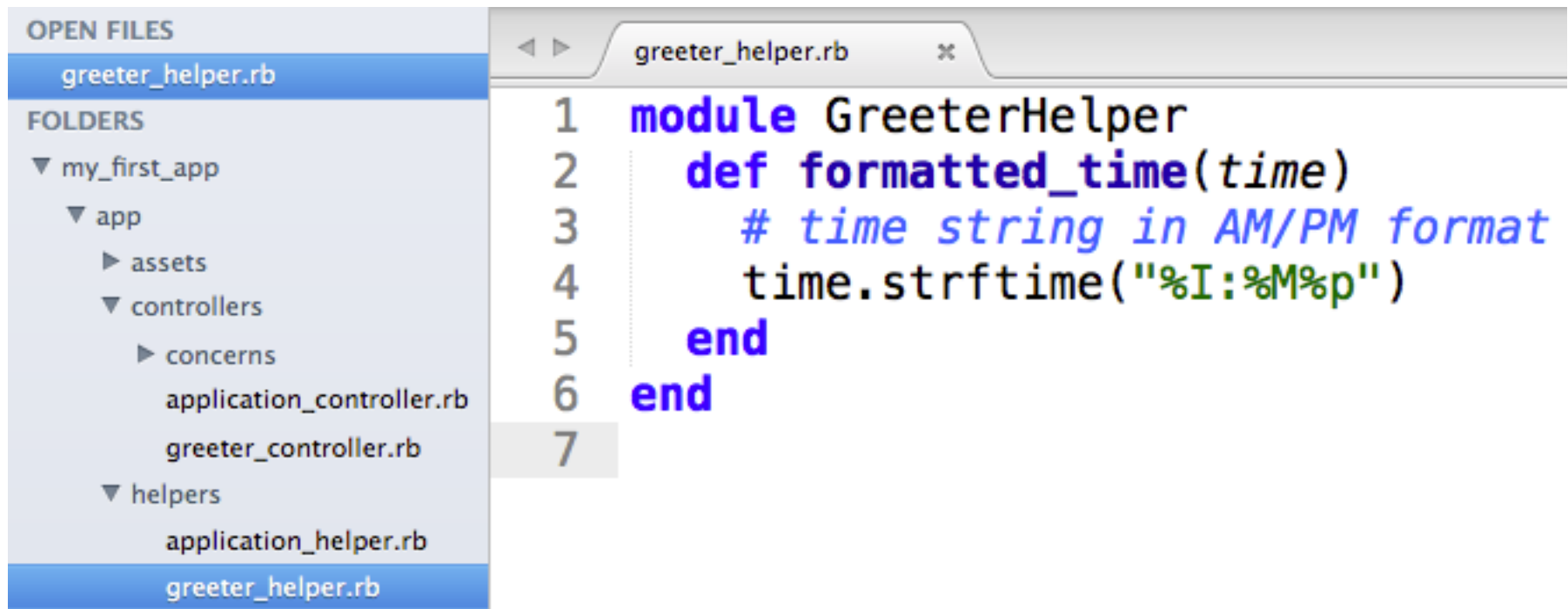
- Unlike Servlets/JSP – a new instance of the Controller is created **every time a request is made**
- Therefore, you can't “store” values in the Controller's instance variables in between requests
- Alternative?
 - Session
 - Database

Helpers

- We've made the current time available through `@time` instance variable (controller)
- But what if we want to format how the time looks? Should that code go in the View? Then, we can't reuse it. So... maybe the Controller? That doesn't seem correct either – since the Controller should be “view format” agnostic
- Helpers to the rescue!

Helpers (Continued)

- (empty) `greeter_helper.rb` module generated alongside the controller
- Let's add a helper method

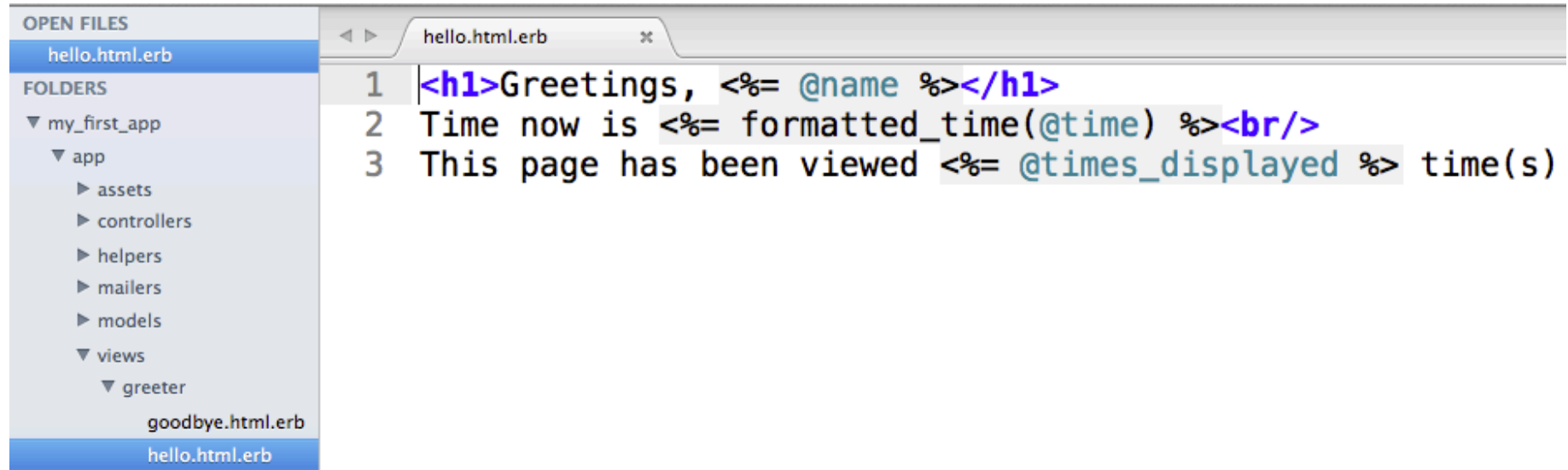


```
OPEN FILES
greeter_helper.rb

FOLDERS
▼ my_first_app
  ▼ app
    ► assets
    ▼ controllers
      ► concerns
        application_controller.rb
        greeter_controller.rb
      ▼ helpers
        application_helper.rb
        greeter_helper.rb

1 module GreeterHelper
2   def formatted_time(time)
3     # time string in AM/PM format
4     time.strftime("%I:%M%p")
5   end
6 end
7
```

Helpers (Continued) – New View



The screenshot shows a web application editor interface. On the left is a sidebar with 'OPEN FILES' and 'FOLDERS'. The 'OPEN FILES' list includes 'hello.html.erb' and 'goodbye.html.erb'. The 'FOLDERS' list shows a project structure with 'my_first_app' containing 'app', 'assets', 'controllers', 'helpers', 'mailers', 'models', 'views', and a 'greeter' sub-folder. The main editor area displays the code for 'hello.html.erb' with three lines: 1. `<h1>Greetings, <%= @name %></h1>`, 2. `Time now is <%= formatted_time(@time) %>
`, and 3. `This page has been viewed <%= @times_displayed %> time(s)`.

```
1 <h1>Greetings, <%= @name %></h1>
2 Time now is <%= formatted_time(@time) %><br/>
3 This page has been viewed <%= @times_displayed %> time(s)
```

Greetings, Joe

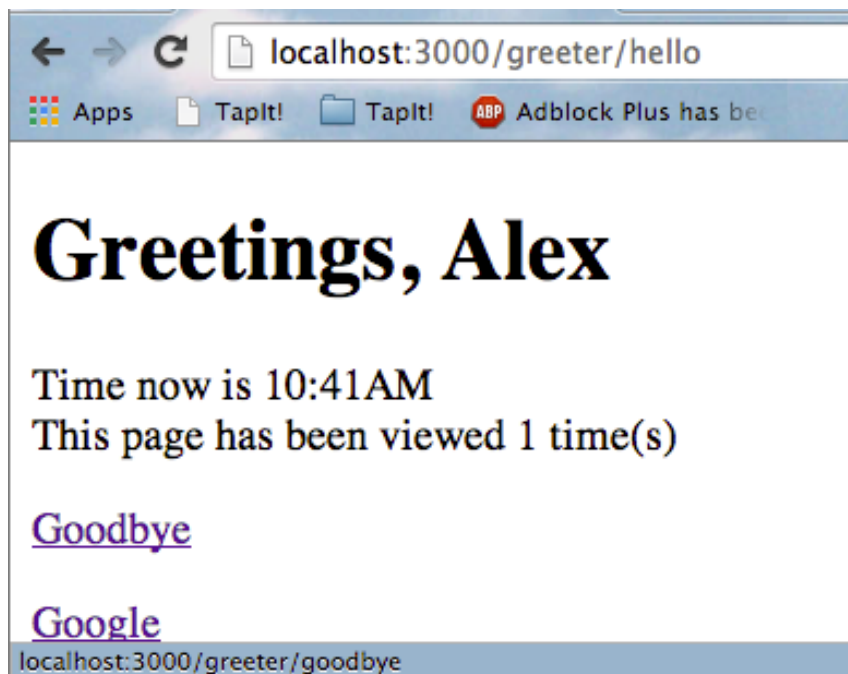
Time now is 10:32AM

This page has been viewed 1 time(s)

Rails built-in helpers – `link_to`

- Rails provides many built_in helpers
- `link_to name, path`
 - Hyperlink generator that displays the `name` and links to the `path`
 - Path could either be a regular string or a route defined in the `routes.rb` file ending with `_url` (full path) or `_path` (relative path)
 - `_url` and `_path` used interchangeably, but according to the spec full path is required in cases of redirection

link_to in action



greeter_goodbye
derived from
routes.rb
(\$rake routes)