RAILS ROUTING

Learning objective

- Understand flow of control in a Rails app.

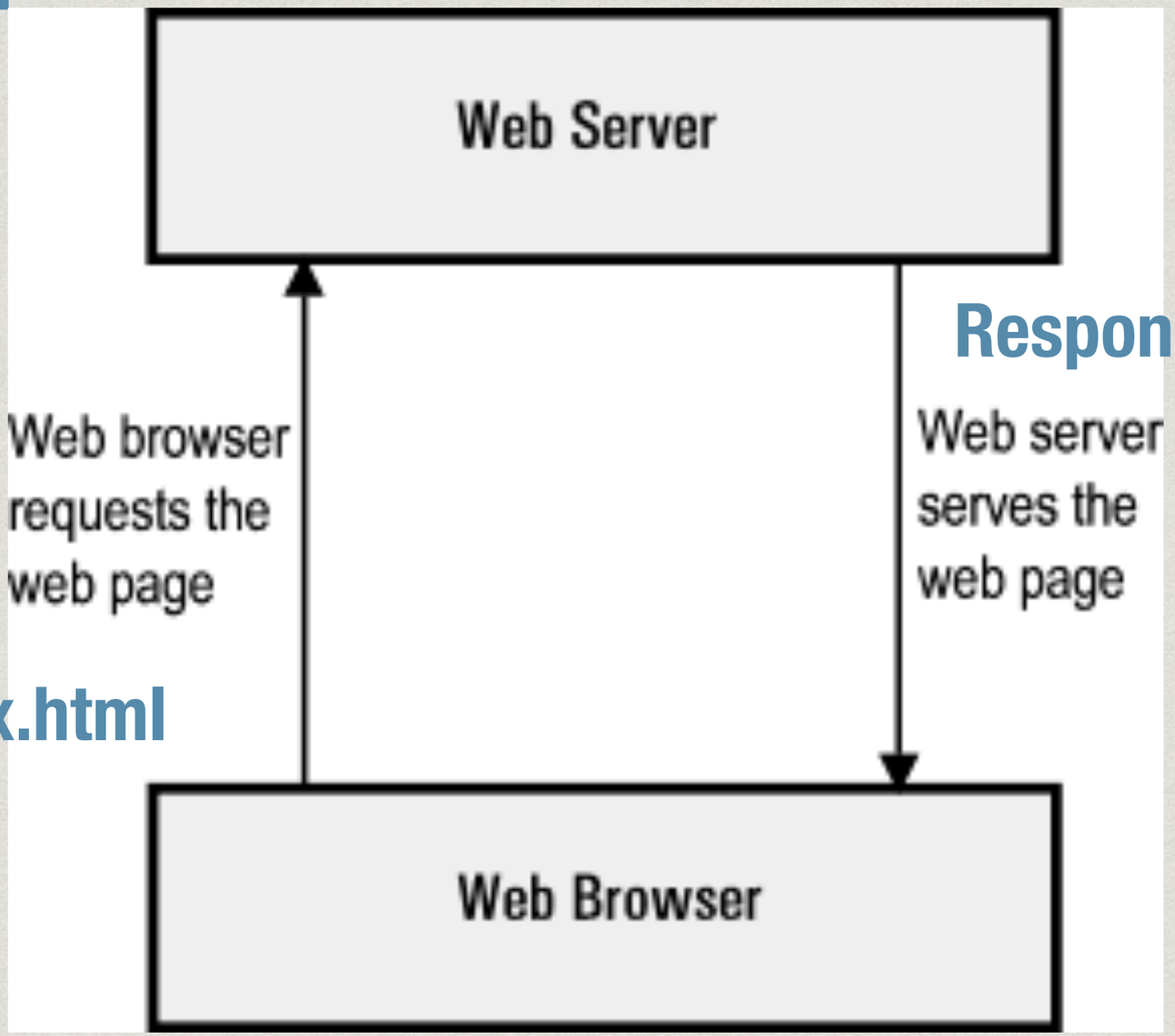
The next few days

- Today: routes and controller actions
- Thursday: introducing models and database tables
- Next Tuesday: hardcore practice on models, controllers, views and routes
- Next Thursday: forms in the browser, meet the database

Agenda

- Routes
- Controllers
- Views
- Layouts
- Lab Time

Old school request-response



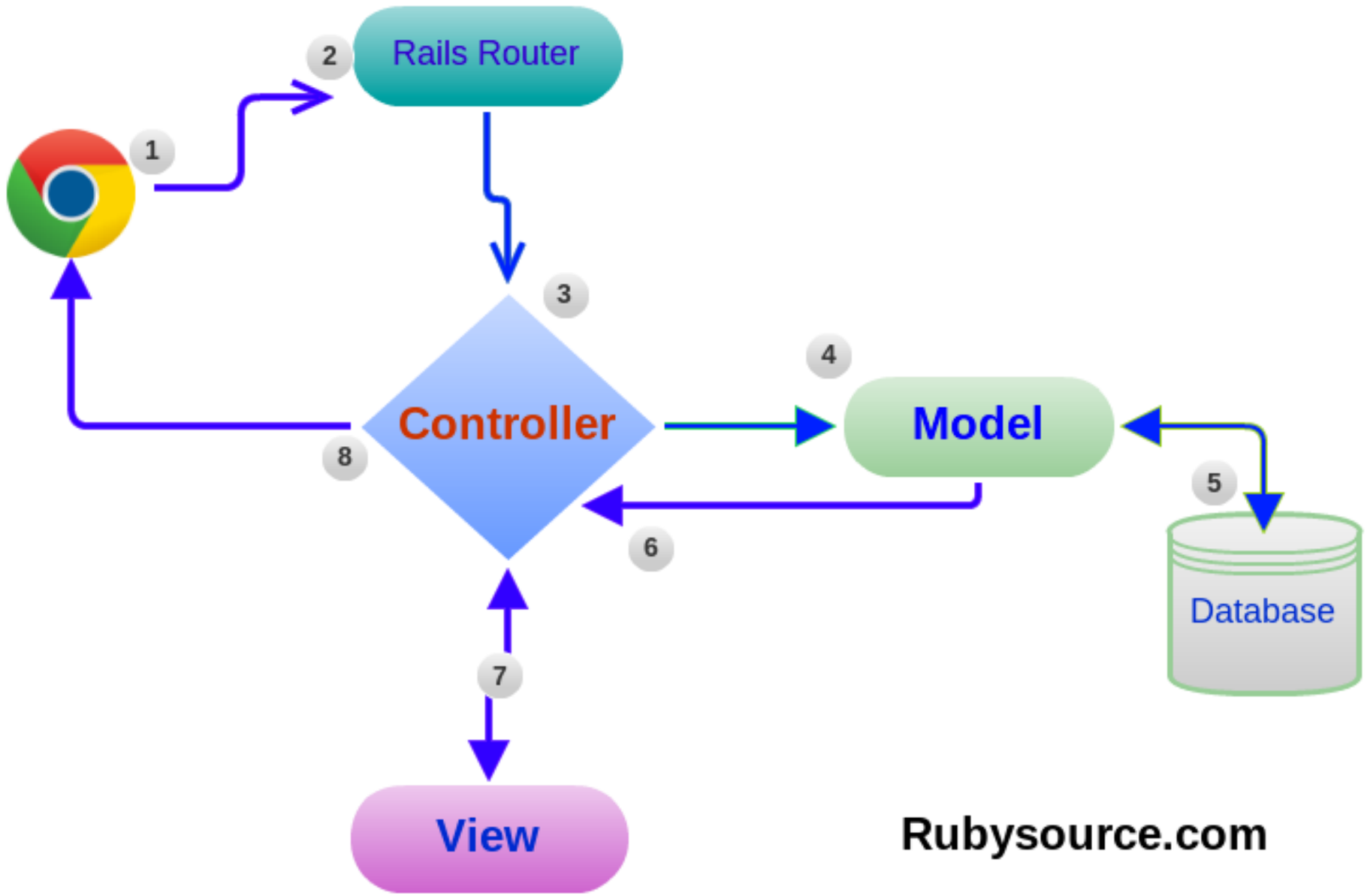
GET index.html

Respond with index.html

User sees index.html

- Rails does this with the public folder. We did this before Thanksgiving break.
- A file is literally sitting on a server, a browser requests it, and the server sends it back, no questions asked.
- Imagine if Facebook or Twitter worked that way. That would be bananas.

MODEL - VIEW - CONTROLLER


Rails Router
2
1
3
Controller
4
Model
8
5
6
7
Database
View
Rubysource.com

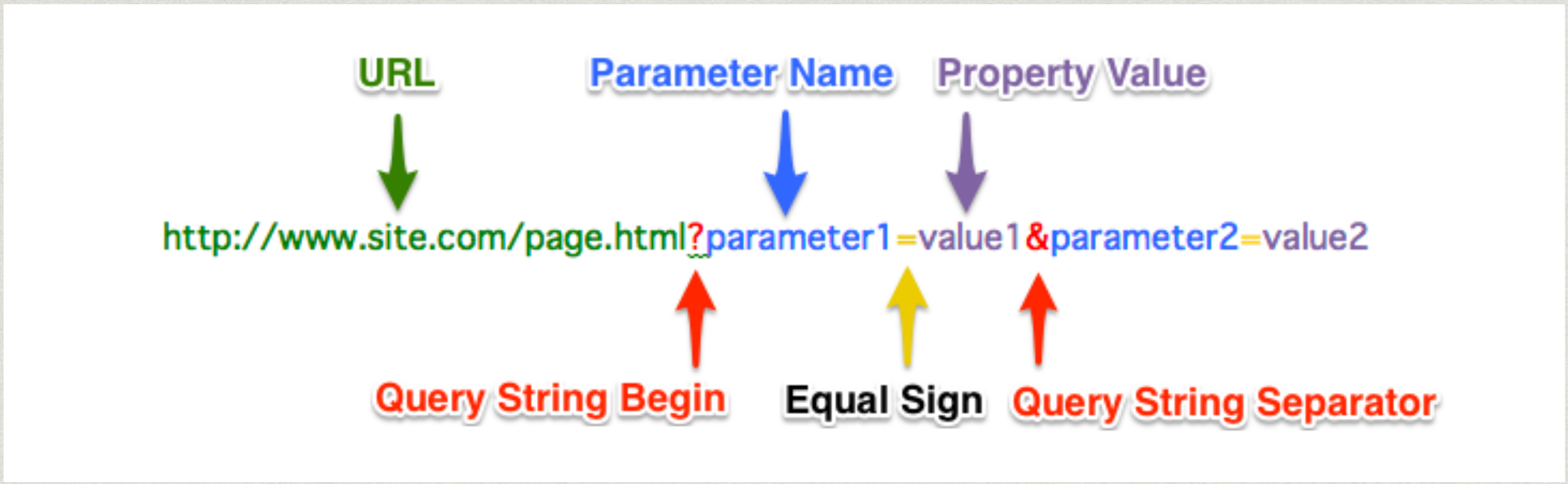
What's a route?

- Connects URLs to code
- So given the URL: "http://example.com/hello"
- A route could define " GET /hello" and direct it to the appropriate controller (ruby code) that determines what to do
- From the rails documentation: "The Rails router recognizes URLs and dispatches them to a controller's action"

What's a URL

- A URL is a string that contains information leading to a resource.
- It contains many parts, such as protocol, domain, path, query, etc…

URL Breakdown



Method breakdown

- The HTTP protocol has many methods of request.
- There are methods that specify creating, deleting, updating, viewing, and more.
- Remember what they are?

Method breakdown

- The HTTP Methods we'll need are:
  - GET - This specifies the request only wants information from the application, nothing else
  - POST - This specifies the request wants to create data in the application (think, registering a new account)
  - PUT/PATCH - Update a record in the application
  - DELETE - Delete a record in the application

Review with a partner…

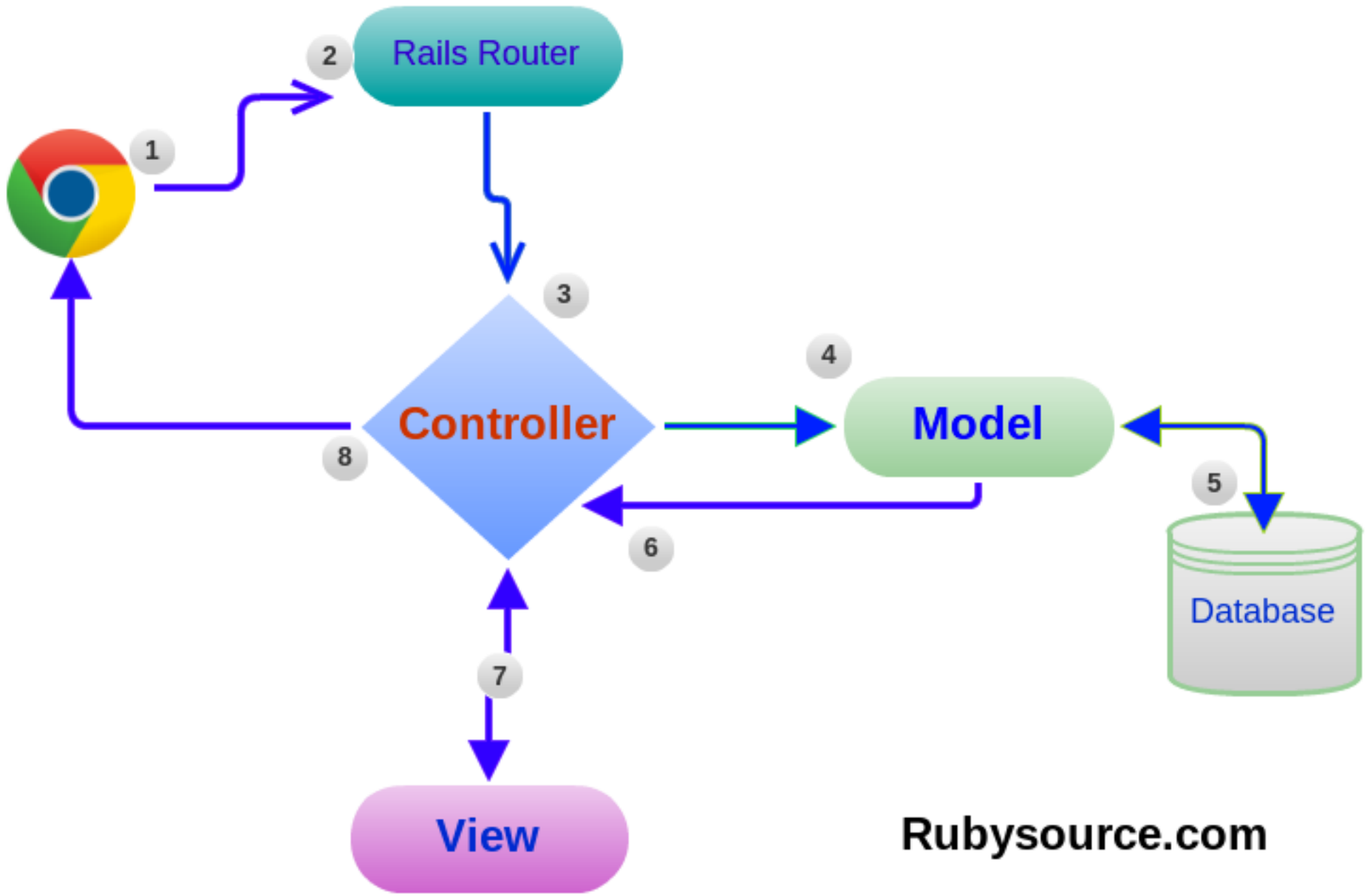
- Construct the following URL from these bits:
- Path: “/hello-world”
- Domain: “placekitten.com”
- Protocol: “https://“
- Query Parameters: “cute=totally”

Review with a partner…

- Answer:
  - https://placekitten.com/hello-world?cute=totally

I’m just an HTTP request

- Let’s examine the life of a request to a rails app
- Say a user types http://generalassemb.ly/classes into a browser.


Rails Router
2
1
3
Controller
4
Model
8
5
6
7
Database
View
Rubysource.com

Now to the router

- The first part of the app that gets hit is the router. It's kind of like those old telephone switching stations.
- It checks that there is in fact a /classes route to respond to the GET method.
- It's like the operator first checking whether the recipient exists.

Now to the router

- If the route doesn't exist, you get an error, just like if you asked the operator for a fake person.
- It also checks whether that route allows that request method.
- In this case, GET /classes is valid, but not POST /classes.

LET’S DO IT TOGETHER

Rails Routes

- All rails routes are defined in a file “routes.rb”
- It is located in the “config” directory in a rails application folder
- So “config/routes.rb”
- You do not need to specify much in a rails route, commonly only the path portion of the URL.

Rails Routes

```
Rails.application.routes.draw do
  get '/hello-world' => "controller_name#action_name"
end
```

Method

Path

The controller

Controllers action

Recap

- Pressing enter in your browser sends a request to a server, and the rails app on the server.
- The rails app looks at the url being requested, and directs the request.
- The directed request goes to a controller, and its action.
- The controller renders the HTML to send back to the requester

CODE ALONG

Routing code-along

- Head to your desktop at the command line.
- Tap in `rails new routing_demo`
- Then `cd routing_demo` (I always forget that)

CONTROLLERS

Controllers

- “After routing has determined which controller to use for a request, your controller is responsible for making sense of the request and producing the appropriate output.”
  - Rails Documentation

- Controllers carry the responsibility of responding to browser requests with the proper HTML and/or text.

```ruby
Rails.application.routes.draw do
  get '/hello-world' => "controller_name#action_name"
end
```

Method

Path

The controller

Controllers

- Controllers live in the “app/controllers” directory in your rails application
- Controllers are JUST RUBY CLASSES
- Controllers always inherit from “ApplicationController”

Controllers

- Controllers only require you to define a method on the class that corresponds with the route defined.

```
Rails.application.routes.draw do
  get '/hello-world' => "controller_name#action_name"
end
```

The controller

Controllers action

- Given a controller called “Users”, and an action called “index”
- Your code would look like:

```
class UsersController < ApplicationController
  def index
  end
end
```

Controllers

- The filename must match the name of the controller.
- So UsersController would be app/controllers/users_controller.rb
- Note: **All controllers must have Controller in the class name**

Private methods in controllers

- Methods that you use in the controller
- After the key word `private`, no methods declared can be used to control HTTP requests.

VIEWS

```
Rails.application.routes.draw do
  get '/users' => "users#index"
end
```

- If we wanted the URL: http://zipline.com/users to go to our Users' controller and its index action, our route would look like this

Leading to…

```
class UsersController < ApplicationController
  def index
  end
end
```

Views

- Views (otherwise known as “Templates”) are the HTML of your application.
- Views correspond to controllers and their actions (but not exclusively)
- The are located in the “app/views/“ portion of your application.

Views

- Correlate based on the name of the controller and the name of the action
- So given a Controller called “users”, with an action “index”
- The path for the view’s template would be:
  - app/views/users/index.html.erb

Template file names

- The file names for a view may seem weird, but also have reason:
- index.html.erb
  - .erb stands for “erubis”
  - the .html is mostly descriptive at this point, it doesn’t necessarily change much in the controller
  - the `.html.erb` is basically saying “This file is HTML with Erubis inside of it”

<%= %>

- Like a really ponderous alien.
- It means hang on, it’s ruby time. Back to HTML later.
- And prints the ruby to the screen

<% %>

- This is where you put ruby that doesn't actually return, such as your `.each` and `end` statements.

```
<% @games.each do |game| %>
    <%= game %>
<% end %>

<% 1 + 1 %> // won't show
<%= 2 + 2 %> // will show!
```

BREAK

Lab

- make a new rails app called `politeness`
- Allow your rails app to receive 2 new requests
  - GET /greeting/english
  - GET /greeting/espanol
- Both of these routes should go to a controller called “Greet”
- Make each new action display a polite greeting in the specified language.

Layouts

- The bread that makes the sandwich of your templates.
- Layouts often contain the common template of your application.
  - Header
  - Footer
  - Navigation bar
  - etc…
- Layouts are **responsible for rendering your template file**!!!!

Layouts

- A layout must have <%= yield %> in order to render the rest of the page
- The <%= yield %> renders whatever your “app/views/“ file is

The application layout

- Every new rails app will be generated with a “application” layout.
- It is located in `app/views/layouts/application.html.erb`
- Go ahead and open that file right now

Finding Yield

- `<%= yield %>`
- This is where your template will be inserted into this file.
- <%= This is indicating “echo this into the template”
- %> is indicating “ok done echoing into the template”

Root routes

- This is the route that gets hit when you visit '/'
- At the top of your routes.rb for your English/ Spanish app add `root "greeting#english"`
- so now when you go to localhost:3000 you will see the greeting in English.
- Try the same for Spanish

Create a new application

- In a new application called "bookshelf", create a route for “/books” that goes to a controller called “books” and action “index”
- Create a controller that wires up to the route you just created (MANUALLY!)
- After that, create a view file for the “all” action that displays text of your choice
- When you’re done with that, go ahead and add text to the application layout so it’s displayed on every page

Homework

- Create a simple rails app called hello_dolly.
- When I run it, and go to localhost:3000/hello/dolly I should see the message "hello dolly!"
- When I run it, and go to localhost:3000/farewell/roger I should see the message "farewell, roger!"
- Read the Odin Project tutorial on routes up until the section "RESTful Routes"
- http://www.theodinproject.com/ruby-on-rails/routing
- This is due Thursday since it’s a layup and you should get some practice before Thursday

Homework

Rock Paper Scissors App

Due next Thursday