



# MODELS (CONT)

WELCOME BACK

PLEASE DO THE SURVEY IN YOUR EMAIL



# Objectives

- \* You will solidify our basic knowledge of models and where they fit in our rails applications. This might feel a bit repetitive, so bear with me.
- \* You will display information from a database in a view
- \* Updating models with database migrations
- \* Deleting data.
- \* REST



**CREATE**

**READ**



**Now**

**UPDATE**

**DESTROY**



**CREATE**

**READ**



**UPDATE**

**DESTROY**



**By end of day**



**LET'S GROK THAT WHOLE  
MVC THING**



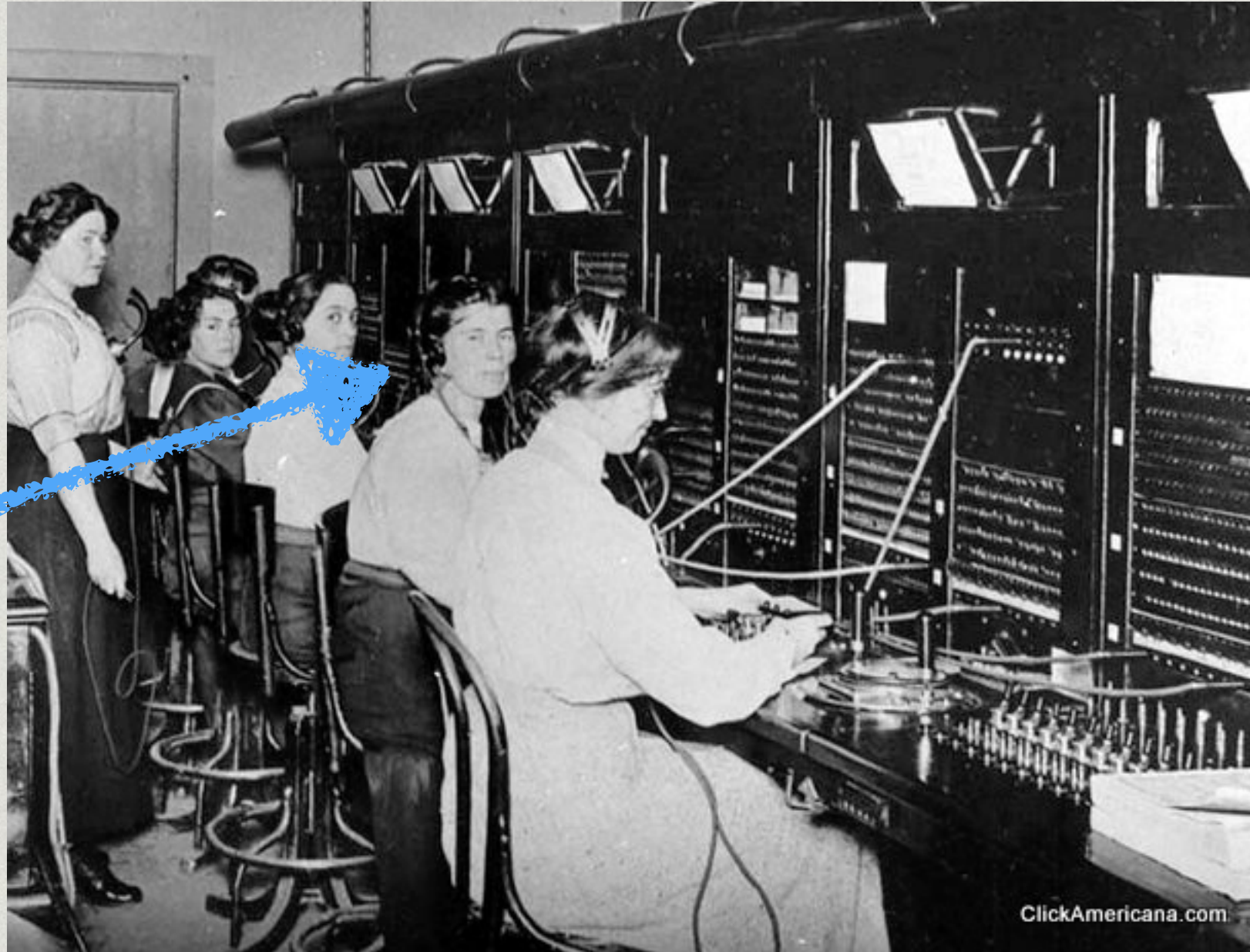
# The request

- \* As we know, all requests go to the Rails Router first, also known as the dispatcher.
- \* The dispatcher then routes the request to a controller.



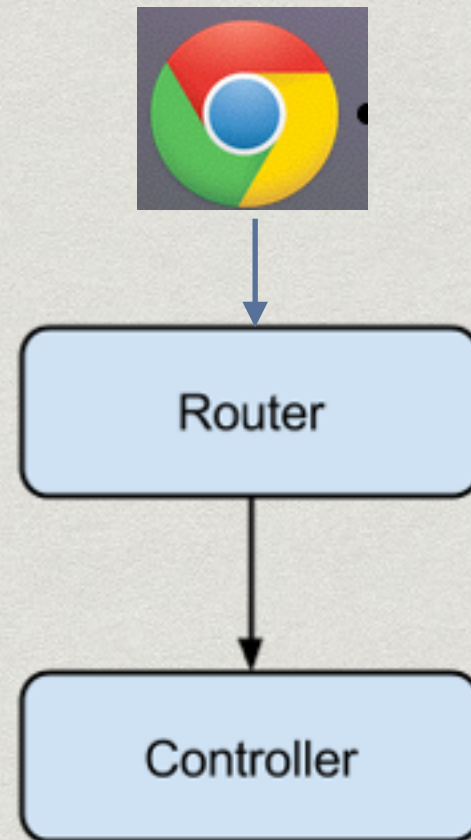
# Dispatcher

Amped  
for work



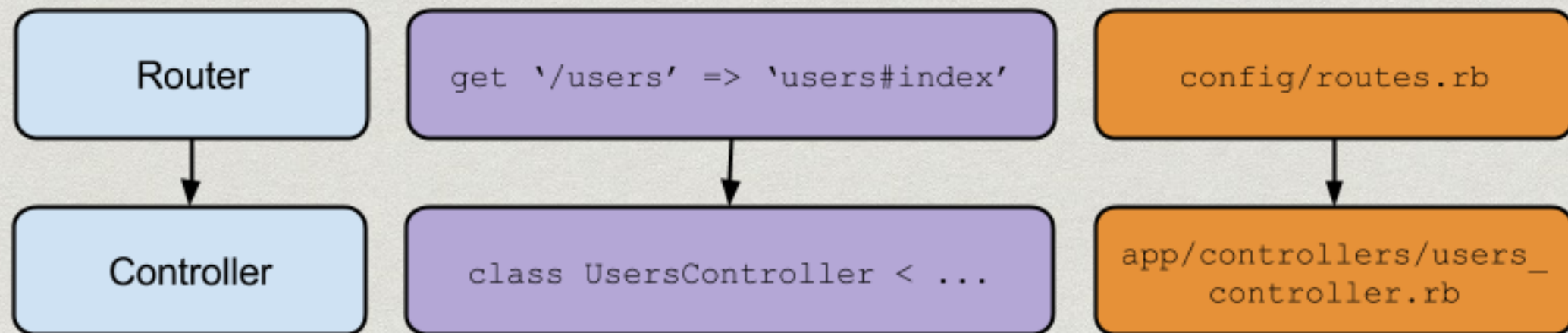


# The request





# More Specifically





# Controller Logic

- \* It is the controllers logic to figure out what to do when receiving a request
- \* This includes retrieving information from a database with a model.
- \* IE: If requesting all of the users of a website, you'd need to retrieve all of the user rows from the users table using the User model

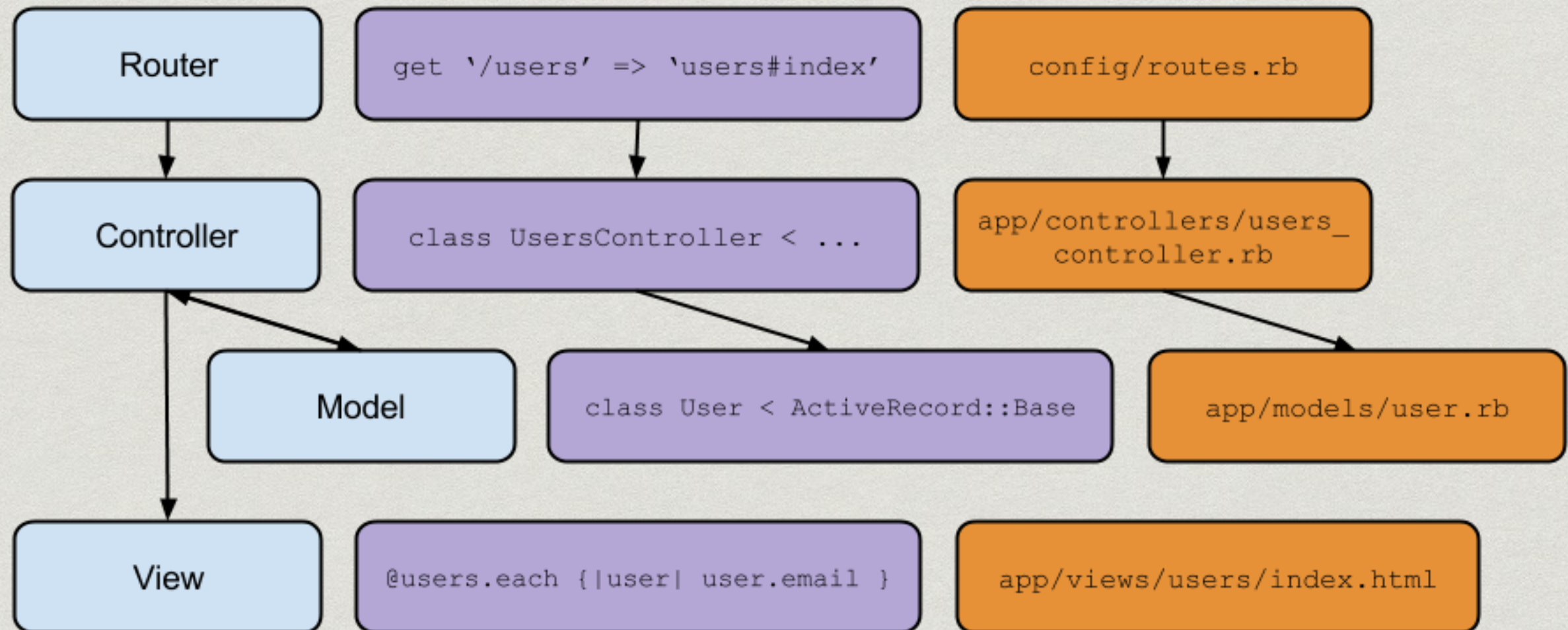


# Controller Logic

- \* After the model gives back the information from the database to the controller, the controller should render the view.
- \* When rendering the view, the controller will give the information it retrieved from the model to the view
- \* The view will use the information render all of the users and their information

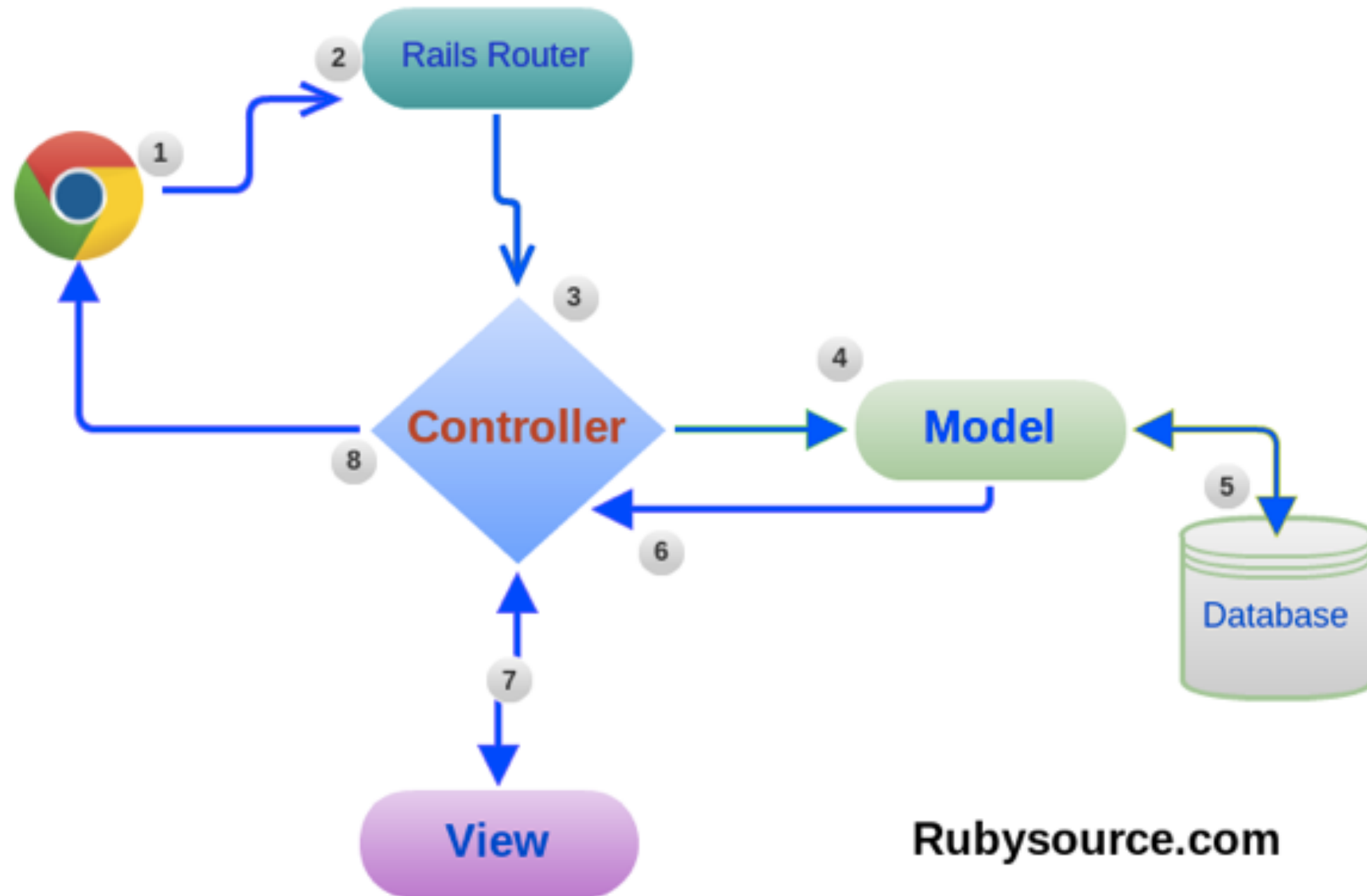


# Building On





# Diagram Recap





# A few notes

- ✱ A controller does *not* need to use a model. In fact, it is quite common to have a welcome controller to handle landing page.
- ✱ A controller *could* return data that is not returned by the database model.



# In-class exercise

- \* In your class 10 folder (HALF WAY, whoa)
  - \* `rails new business_time`
  - \* `cd business_time`
- \* After that, please generate (hint hint) a model named “Company” with 2 attributes:
  - \* `name` with a type of `string`
  - \* `number_of_employees` with a type of `integer`



# Routes Exercise

- \* Remember, all routes exist in the “config/routes.rb” file within your Rails application directory.
- \* Please add a route for a GET request to “/companies” that routes to a controller and action:
  - \* Companies controller
  - \* index action



# Controller Exercise

- \* Remember, all controllers must be added to the app/controllers directory
- \* Please create a controller called Companies with an action called “index” (manually!, **no** rails g)
- \* Remember, actions are fancy methods on controllers.



# View Exercise

- \* Remember, all views live in the app/views directory, with a subfolder that is named after the name of the controller.
  - \* So given a controller called “users”, the directory path would look like: “app/views/users”
- \* The name of the action is also the name of the view file. So with an action called “index”, the file would be:
  - \* “app/views/users/index.html.erb”
- \* With that, please create a view file for the controller we generated, leave it empty.



# Passing information to the view

- \* Remember, it is the controller's job to pass information to the view
- \* We're going to play around with that a bit.



# Controllers & Views

- Controllers use what's called an "instance variable"
- Instance variables start with an @ sign

## controller.rb

```
1  class MyController < ApplicationController
2    def hello
3      @my_instance_variable = "Hello World!"
4    end
5  end
```



# Instance Variables

- \* They're very similar to normal variables, but they are for the instance of a class, and *thats it*.
- \* This means that 2 different instances of a class can't access the same instance variable.
- \* Rails creates a new instance of your controller class with every request.
- \* Anything outside of the instance of that class can't access the variable.




# Controllers & Views

- \* Controllers must assign an instance variable in their action methods to send data to the view file.  
For example:


## controller.rb

```
1  class MyController < ApplicationController
2    def hello
3      @my_instance_variable = "Hello World!"
4    end
5  end
```



## hello.html.erb

```
1  <h1>
2    <%= @my_instance_variable! %>
3  </h1>
```





# ERB Tags


- \* ERB Tags are fancy tags that can run and echo Ruby into HTML.
- \* They start with `<%` and end with `%>`
- \* When an ERB tag starts with `<%=`, this means “print the value of this ruby code”



# ERB Tags

So this...

 **hello.html.erb**



```
1 <h1>
2 <%= @my_instance_variable! %>
3 </h1>
```

Would print whatever the value of  
“@my\_instance\_variable” is into the view



# Code Along

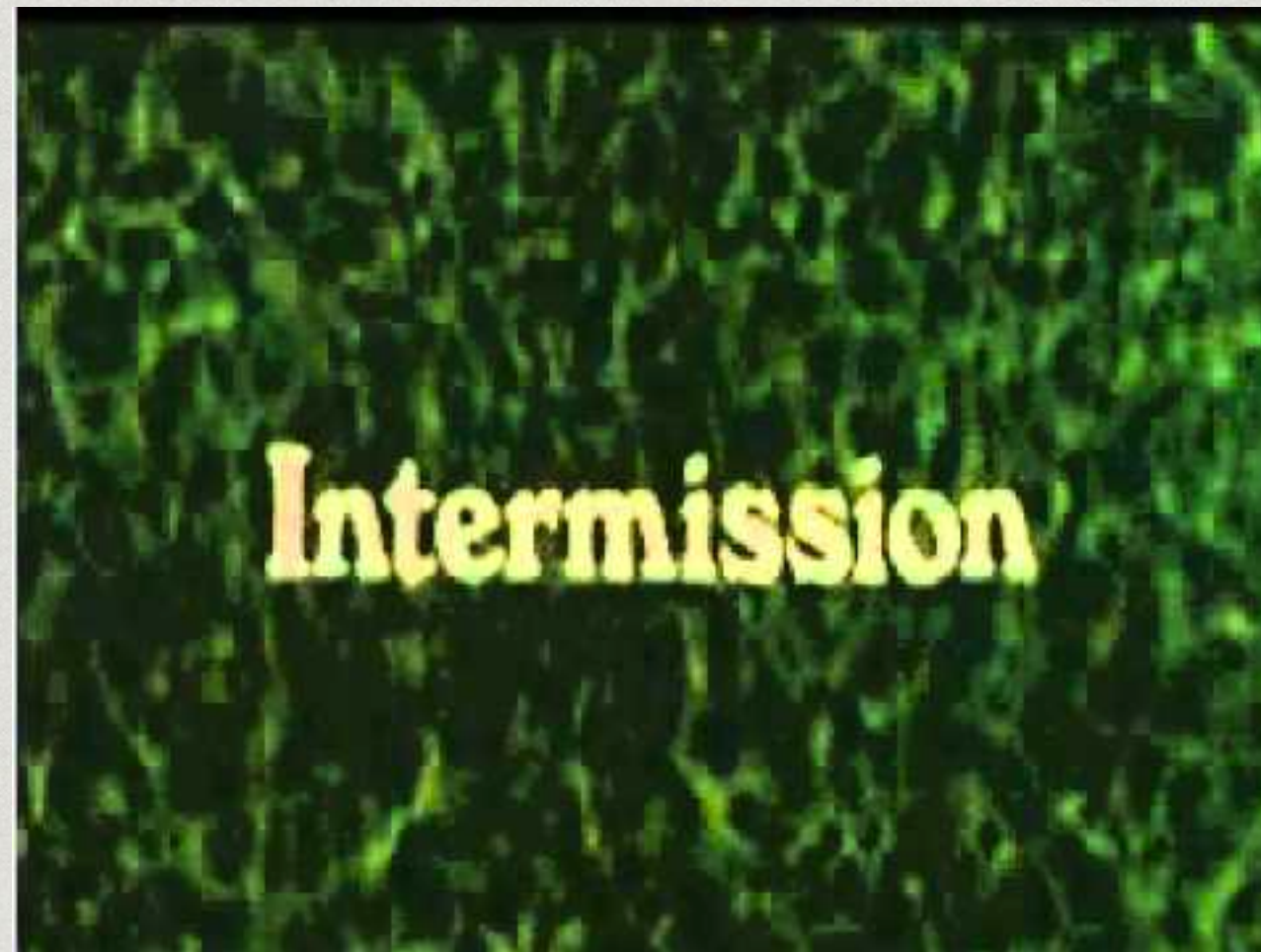
- \* Adding a simple “instance variable” to our controllers that our view will display.
- \* This is the basis for passing information between controllers and views



# Lab

- \* Within our companies index view, next to the company name that is displayed, display the ID of the company as well.
- \* Remember, all things in our database automatically get an ID.
- \* Extra time? Display when they were created too.







# Database Seeds

- \* Rails has a feature called “seeding”
- \* It allows you to create data inside of your database easily
- \* Seeding is typically used to populate the initial data of a database



# Database Seeds

- \* There is a file “db/seeds.rb” that will contain our seeds.
- \* This file is just executed top to bottom. It’s basically a plain ruby file that has all of your models loaded into for you.
- \* To execute seeding, you run the command:
  - \* `rake db:seeds` in your terminal (from within your rails app directory)



# Seeding Notes

- \* A few notes about seeding:
  - \* If you run “rake db:seed” multiple times, you will end up with duplicates.
- \* If you want to reset data and then seed, you can run consecutively:
  - \* rake db:reset
  - \* rake db:seed
- \* Again, this blow away **ALL** of your data. Not just your seeds, so move with caution.



# Seeds Code Along

**Adding a few company seeds using our  
Company model**



# **SORTING COMPANIES**



# Research

- \* Using Google, Stack Overflow, various documentation or whatever, see how you might sort companies by descending order by number of employees on the /companies/rankings page
- \* you have 15 minutes



# Lab

- \* Add another route for GET for “/companies/rankings” that points to our companies controller, and an action called “rankings”
- \* Create the action method and view file for the route.
- \* Within our new action, create an instance variable (@ranked\_companies) and assign it to a string “nothing yet”
  - \* Display this instance variable inside of the new view file you created.
- \* Create a link on the index view to the ranking page
- \* **Bonus:** Print the data type of @ranked\_companies



# Sorting by number of employees

- \* `Company.order(:number_of_employees).reverse`
- \* OR
- \* `Company.order(number_of_employees: :desc)`
- \* Try that in the rails console
- \* Then make it happen in the view, similar to index.
- \* **Bonus:** The companies are ordered by employee number. Let's also number them ordinally.



# UPDATING DATABASE TABLES



# Migrations

- \* We've done migrations when we created models
- \* Let's say you want to update a model that already exists.
- \* You use a different generator for that.



# Add city to company model

- \* If the migration name is of the form "AddXXXToYYY" or "RemoveXXXFromYYY" and is followed by a list of column names and types then a migration containing the appropriate `add_column` and `remove_column` statements will be created.
- \* Okay what does that mean...



# Add city to company model

- \* `rails g migration AddCityToCompanies city:string`
- \* This changes the structure of the database so there is now a column for city.
- \* Before `Company.city` would error. Now it works.



# Try it in the console

- \* Create a new company with a city attribute.
- \* Problem: the old data is sans city.
- \* How can we solve that?



# PARAMS



# Parameters

- \* A request can include a Params hash.
- \* This could be passed in manually. For instance, if you want to know how many people visited your sight from a Twitter campaign like 'click here to save 15%!' you can include a query string in the link like so:
- \* [https://myapp.com/signup?lead\\_source=twitter](https://myapp.com/signup?lead_source=twitter)



# Parameters

- \* [https://myapp.com/signup?lead\\_source=twitter](https://myapp.com/signup?lead_source=twitter)
- \* The controller that handles this get request will have a hash like this:
- \* `params = {"lead_source" => "twitter" }`
- \* If you get an error page, you will see the params hash plain as day.
- \* You can access values in the hash by it's key. Usually as a symbol, but string is allowed.



# Parameters

```
def signup
  if params[:lead_source] == "twitter"
    admission_price = 85
  else
    admission_price = 100
  end
end
```



# Parameters

- \* Oftentimes, you will code parameter keys directly into routes.
- \* Let's make a `/companies/cities/<somecity>` route to demonstrate.



# Filtered queries

- \* `@companies = Company.where(city: params[:city])`



# Filtered queries

- \* On the cities page, we want to show not ALL of the companies, but all of them that are in the specified city.
- \* Hit the docs for 5, see if you can come up with a statement that queries the DB for the companies that match the one the user put into params.
- \* `/companies/cities/london` should return all of the companies where city is london. (case matters)



# Lab

- \* Implement a route, controller action and View to show companies by city.
- \* **Bonus:** Display the requested city in the view.
- \* **Bonus:** Display a special message to the user if no companies are found for that city.



**MOAR LAB**



# Lab

- \* Your client changed its mind. They no longer care about number of employees. They feel that their investors are better informed by rankings by `market_capitalization`
- \* Remove `num_employees` with a new migration, add the requested attribute
- \* Rework the `/rankings` route to accept either `rankings/` `market-cap`
- \* If you finish early, make a route for `rankings/profit-margin` that uses the same controller (you will need another migration)



# RESOURCEFUL ROUTES



# HTTP Verbs

- \* Can you name four of them?
- \* Which one have we done exclusively since we abandoned rails scaffolding?



# Standardize your routes for fun and profit

- \* This is one of those moments where rails rewards you for following convention.
- \* Resourceful routes give you seven standard routes with just one line.



# Resourceful routes

```
Rails.application.routes.draw do  
  
  resources :companies  
  
  root 'companies#index'  
  
  get 'companies/rankings' => 'companies#rankings'  
  get 'companies/cities/:city' => 'companies#cities'
```

**You still need these since they aren't standard.**





# Run rake routes

```
vincent@apple:~/Desktop/business_time $ rake routes
```

Prefix	Verb	URI Pattern	Controller#Action
companies	GET	/companies(.:format)	companies#index
	POST	/companies(.:format)	companies#create
new_company	GET	/companies/new(.:format)	companies#new
edit_company	GET	/companies/:id/edit(.:format)	companies#edit
company	GET	/companies/:id(.:format)	companies#show
	PATCH	/companies/:id(.:format)	companies#update
	PUT	/companies/:id(.:format)	companies#update
	DELETE	/companies/:id(.:format)	companies#destroy
root	GET	/	companies#index
companies_rankings	GET	/companies/rankings(.:format)	companies#rankings
	GET	/companies/cities/:city(.:format)	companies#cities

```
vincent@apple:~/Desktop/business_time $
```




# Run rake routes

```
vincent@apple:~/Desktop/business_time $ rake routes
```

Prefix	Verb	URI Pattern	Controller#Action
companies	GET	/companies(.:format)	companies#index
	POST	/companies(.:format)	companies#create
new_company	GET	/companies/new(.:format)	companies#new
edit_company	GET	/companies/:id/edit(.:format)	companies#edit
company	GET	/companies/:id(.:format)	companies#show
	PATCH	/companies/:id(.:format)	companies#update
	PUT	/companies/:id(.:format)	companies#update
	DELETE	/companies/:id(.:format)	companies#destroy
root	GET	/	companies#index
companies_rankings	GET	/companies/rankings(.:format)	companies#rankings
	GET	/companies/cities/:city(.:format)	companies#cities

```
vincent@apple:~/Desktop/business_time $
```



**Handy path helpers** — we will use these in the controller actions and links.



# link\_to

- \* Rails has an intuitive link format you can work with.
- \* Let's implement this to link the index view to the show (single company) view.



# link\_to

```
<%= link_to 'text', resource %>
```



# link\_to

**For our index page:**

```
<%= link_to 'Show', company %>
```



# link\_to

**Back to the index**

```
<%= link_to 'Back to index', companies_path %>
```



**DESTROYING DATA**



# Deleting data

- \* User clicks the delete button.
- \* DELETE request is sent to the server.
- \* The controller finds the right record to destroy, and destroys it.
- \* Instead of rendering a template, it should redirect to where you were.
- \* The Company model already has a delete route.



# The controller action

```
def destroy  
  Company.find(params[:id]).destroy  
  redirect_to companies_path  
end
```



# The controller action

```
def destroy  
  Company.find(params[:id]).destroy  
  redirect_to companies_path  
end
```

What does this do?



# The controller action

```
def destroy  
  Company.find(params[:id]).destroy  
  redirect_to companies_path  
end
```

What is companies\_path



# Redirecting

- \* For some methods, including destroy, it doesn't make sense to render a template. What is there to see?
- \* For this reason, we will redirect the user to the index view.



# Link for deleting

```
<%= link_to 'Link text', object-to-delete, method:  
:delete, data: { confirm: 'Are you sure?' } %>
```



# Link for deleting

Let's put that on the index page and try it out.



**RECAP**



# Recap

- \* Migrations are used to update existing tables/models in the database.
- \* `.where` on a model filters a database query
- \* `.order` sorts by an attribute.
- \* `link_to` is a handy link helper
- \* `redirect_to` is necessary in controller actions that don't have sensible views of their own.