CS 416
Web Programming

Ruby on RAILS
Chapter 1

Dr. Williams
Central Connecticut State University

Rails development – command line

- Unlike Java, Rails development is largely based on command line instructions
  - While you only technically need to execute the rails instructions at the command line, the more command line instructions you know the easier it will make the development process as you won't be constantly switching back and forth
  - Note that RubyMine automates many of the command line instructions, but even so you can't avoid them all

Creating Rails app

```
rails new hello_app
```

This command does a couple of things:

- Creates a new directory named the web app name you specify
  - Creates rails directory structure
  - Install ruby gems needed by basic rails project

Rails directory structure

File/Directory	Purpose
app/	Core application (app) code, including models, views, controllers, and helpers
app/assets	Applications assets such as cascading style sheets (CSS), JavaScript files, and images
bin/	Binary executable files
config/	Application configuration

Rails directory structure cont.

File/Directory	Purpose
db/	Database files
doc/	Documentation for the application
lib/	Library modules
lib/assets	Library assets such as cascading style sheets (CSS), JavaScript files, and images
log/	Application log files

Rails directory structure cont.

File/Directory	Purpose
public/	Data accessible to the public (e.g., via web browsers), such as error pages
bin/rails	A program for generating code, opening console sessions, or starting a local server
test/	Application tests
tmp/	Temporary files

Rails directory structure cont.

File/Directory	Purpose
vendor/	Third-party code such as plugins and gems
vendor/assets	Third-party assets such as cascading style sheets (CSS), JavaScript files, and images

Rails directory structure cont.

File/Directory	Purpose
README.md	A brief description of the application
Rakefile	Utility tasks available via the rake command
Gemfile	Gem requirements for this app
Gemfile.lock	A list of gems used to ensure that all copies of the app use the same gem versions
config.ru	A configuration file for Rack middleware
.gitignore	Patterns for files that should be ignored by Git

Bundler

- Bundler used to install and include gem libraries in projects
- To add/remove libraries, modify the `Gemfile`
- Syntax:

`gem 'sqlite3'`	Installs latest version of the specified gem
`gem 'uglifier', '>= 1.3.0'`	Installs latest as long as version is greater than or equal to the specified version (so would install ver 7.x potentially)
`gem 'coffee-rails', '~> 4.0.0'`	Installs latest as long as it is equal to or greater than the specified version **and** it is no newer than just the numbers after the last "." are different

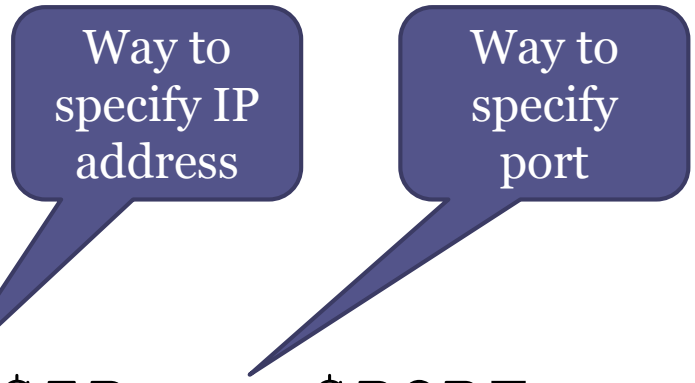
Bundler cont.

Anytime the `Gemfile` is modified run:

`bundle install` to update the relevant gems
(It may prompt you to run `bundle update` first)

If the server was running, restart the server

Rails server



Start the server

Local: `rails server`

C9: `rails server -b $IP -p $PORT`

VirtualBox VM (pass through IP):

`rails server -b 10.0.2.15`

Test if Rails working visit IP and port specified – by default on local machine:

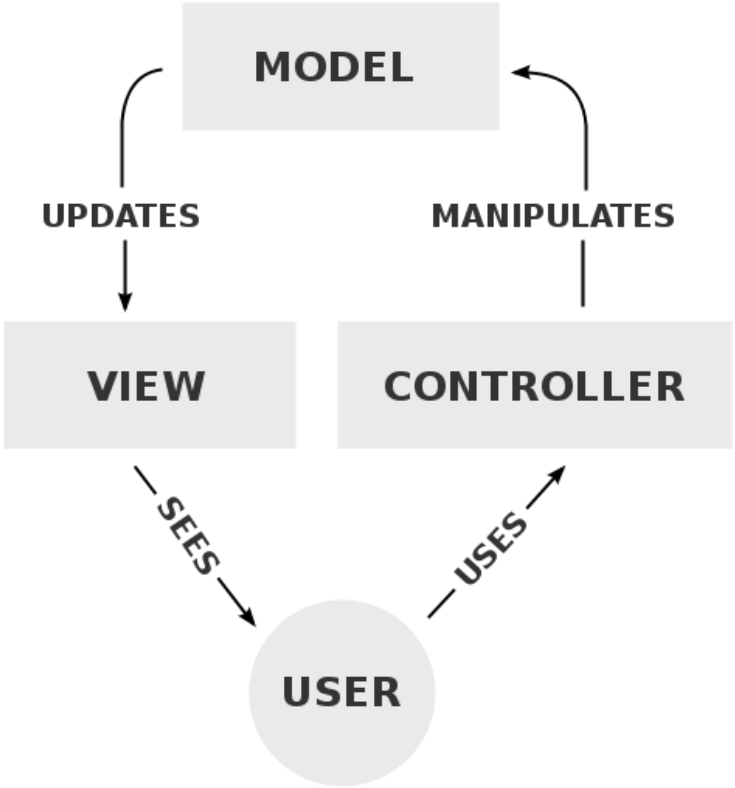
`http://localhost:3000/`

Rails – Model-View-Controller (MVC)

- In generated directory structure application directory contains all application code. Specifically contains 3 subdirectories:
  - `app/models`
  - `app/views`
  - `app/controllers`
- Framework designed from ground up to encourage good MVC design and development practices

MVC

- Design pattern that can apply to any application that has a GUI
- Goal is to separate internal representation and persistence from how it is displayed to the user



Rails MVC - Model

- Model classes extend ApplicationRecord

```
class Classroom < ApplicationRecord
```

- Ties object data to DB tables with plural of same name (`/db/schema.rb`)

- **Represents data** within the site and encapsulates how the data is loaded, stored and queried from DB
- Also contains **business logic** related to the represented object: validation, rules, calculations, relationships to other objects

Rails MVC - View

- Generates output based on data passed from controller and model
- Views named `page.html.erb`, similar to JSP a static **template** (HTML) and code to **inject dynamic elements**
- Controller decides what portion of the Model to load and what other information should be passed to View then template formats passed data for display

Rails MVC - Controller

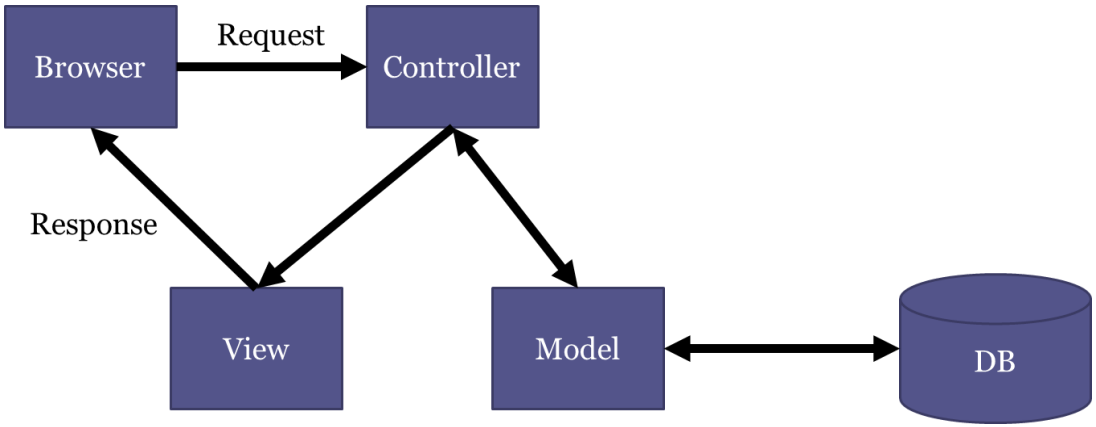
- **Retrieves/loads** Model elements
- Sends **updates** to the Model
- Decides what to do next – application **flow**
- **Passes data** to be displayed to the View

- Extends ApplicationController

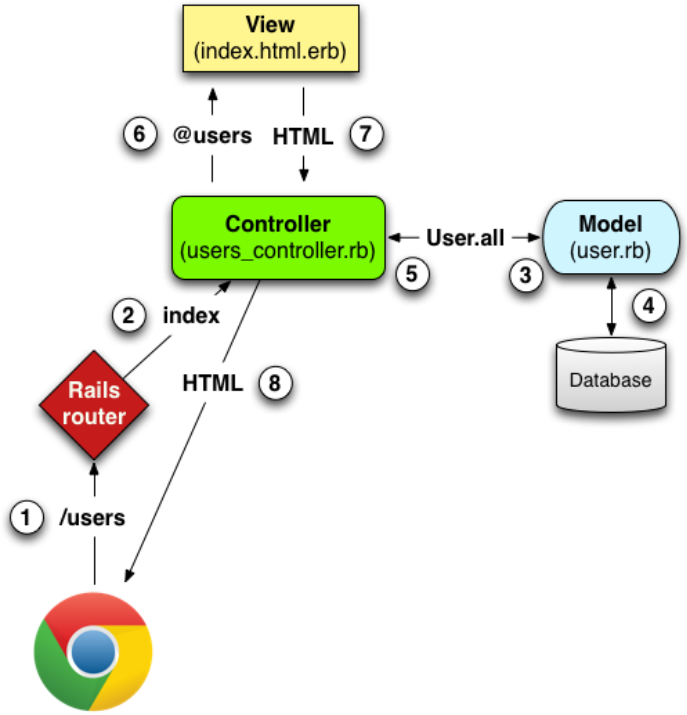
```
class ClassroomsController < ApplicationController
```

Rails MVC – Flow revisited conceptual vs actual

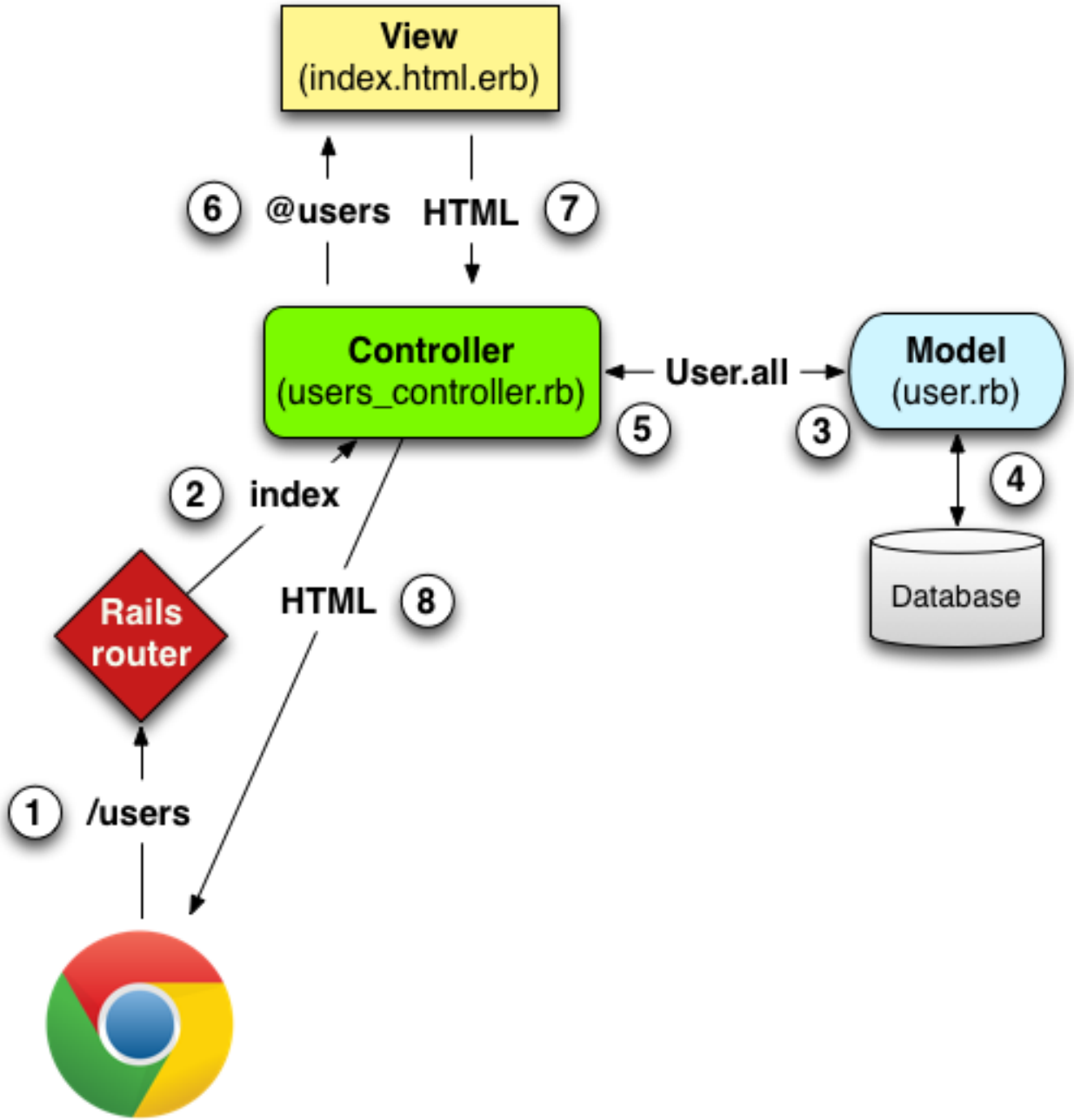
Conceptual flow



Actual flow



Rails MVC detailed flow



Simple “hello world”

- What we are doing:
  - Adding a route: given a URL where the framework should direct the call
  - Having controller pass data to view
  - Having view use the passed data

Hello world changes

- Adding a route (*config/routes.rb)*

```
Rails.application.routes.draw do
  root 'application#hello'
end
```

This sets a reference of "/" to route to the `hello` method on the `application` controller

Add controller method

- Adding a *hello* action to the Application controller

(In *app/controllers/application_controller.rb)*

```
class ApplicationController <
ActionController::Base
  protect_from_forgery with: :exception

  def hello
    render html: "hello, world!"
  end
end
```

The view

- Note no changes to:
views/layouts/application.html.erb

```
<html>
  <head>...</head>

  <body>
    <%= yield %>
  </body>
</html>
```

Add repository to git

- From root directory of your created project
- `git init`
- `echo "# My application" >> README.md`
- `git add .`
- `git commit –m "Initial version"`
- Create repository on GitHub and copy URL
- `git remote add origin https://github.com/.../something.git`
- `git push –u origin master`

If using Cloud 9 online text has instructions for configuring git

Also refresher of git command line we covered

Rails deployment

- While `sqlite3` is great for development as it requires zero configuration it isn't scalable
- Very common scenario is having different configurations for development and production
- Recall `Gemfile` specifies libraries Rails app is dependent on, solution is to specify different set of libraries dependent on deployment target!
- *Java equivalent is different XML configuration files for each target*

Target dependent Gem configuration

`Gemfile` libraries to include for production:

```
group :production do
  gem 'pg', '0.18.4'
end
```

Note this includes PostgreSQL, remove reference to sqlite3 for all deployments

`Gemfile` libraries to include for development and test:

```
group :development, :test do
  gem 'sqlite3', '1.3.11'
  gem 'byebug',  '9.0.0', platform: :mri
end
```

Then:

```
git add .
git commit –m "Production Gemfile"
```

Bundle for different deployments

- With no options specified `bundle` will install libraries for all environments
- Instead run `bundle` specific to deployment environment:

```
bundle install --without production

bundle install --with development test
```

Rails cloud deployment

- Heroku makes deploying to the cloud with rails incredibly easy
- Git driven – their servers pull directly from your git repository
- Step 1: Connect with Heroku
  - **`heroku login`**
  - **`heroku keys:add`**
- Step 2: Create new app at Heroku
  - **`heroku create`**
- Step 3: Push from your git repository to Heroku
  - **`git push heroku master`**
- Done!
- *Note if you cloned the repository you may need to run:*

`heroku git:remote -a falling-wind-1624`

Substituting "falling-wind-1624" with the app heroku created for you

See textbook for install of heroku if working on platform other than Cloud 9