Sinatra

1. DSL: domain specific language
2. Install: gem install sinatra thin sinatra-contrib
3. High level structure



1. App Server: THIN -> Mongrel -> Webrick
   1. ./code.rb -o 0.0.0.0, -p 80, default localhost, 4567
2. Code.rb

require 'sinatra'

require 'sinatra/reloader' if development?

# method 'route' {handler}

get '/hello' do

%{<h1> i got your request, this is my response</h1>}

End

# key value via **params** hash

get '/bet/:stake/on/:num' do

stake = params[:stake].to\_i

number = params[:num].to\_i

end

# key value via **block**

get '/hello/:name' do |n|

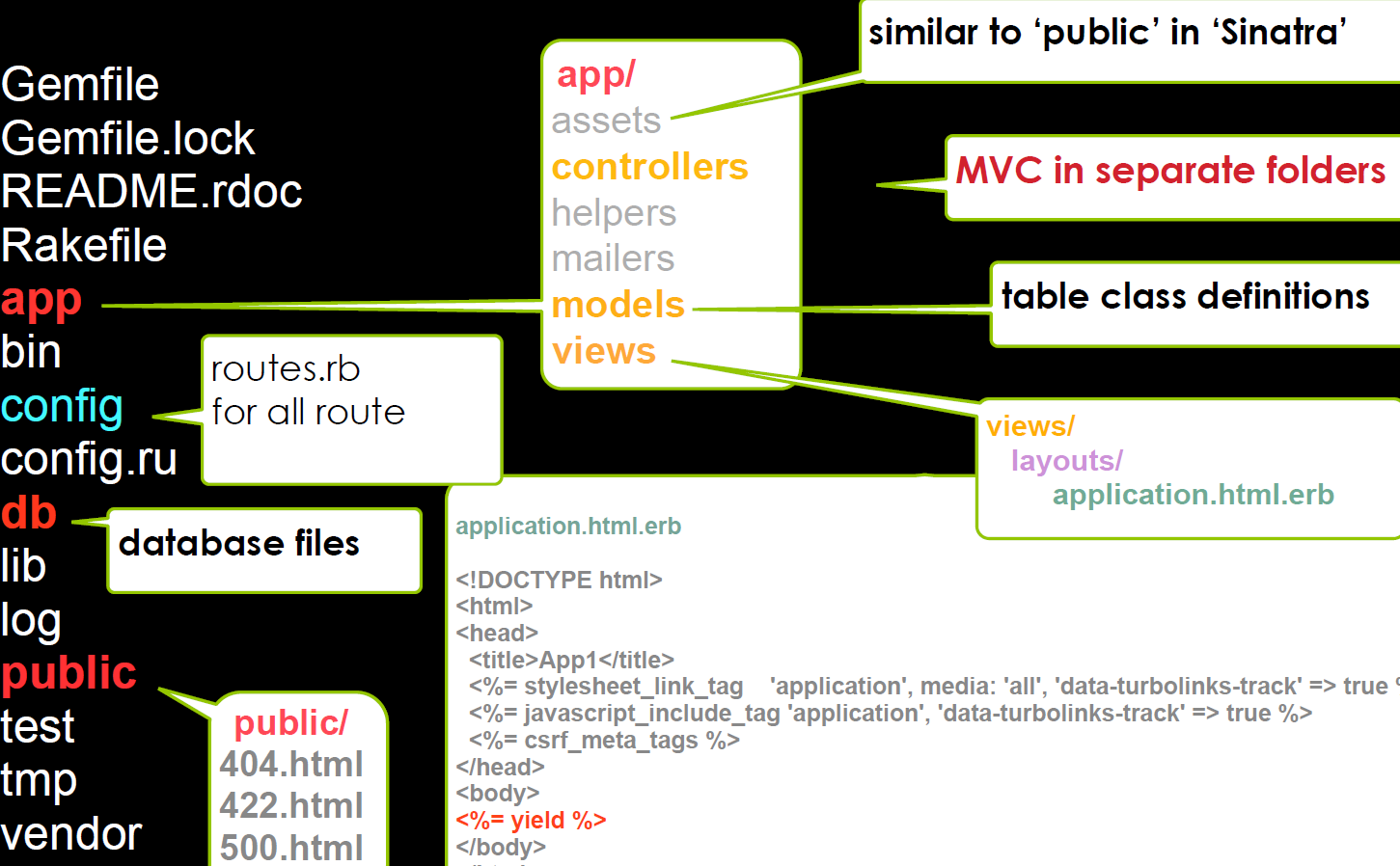
"Hello #{n}!"

end

1. Methods: get, post, put, delete

Rails

1. Generator
   1. controller: rails g controller Home page1
2. Directory structure



1. Start server: rails s -b [ip] -p [port], default localhost:3000
2. public/page1 => localhost/page1
3. Resource convention: /[src]/#route 🡪 [action] in [src]\_controller 🡪 /views/src/[action] view
4. Naming convention
   1. resource\_controller.rb
   2. class ResourceController < ApplicationController
5. Routes and handler defined separately
   1. Routes:/config/routes.rb
   2. Handlers:/app/controllers/ src\_controller.rb
6. Helper
   1. Pass value
      1. from ctrl to view : instance var @var
      2. to partial: local var hash
   2. link

# app

<%= link\_to “url”, textCaption %>

1. Partial: \_partialName.fileType.erb

<%= render “partialName”, locals: {name:'val'} %>

1. Routes

Rails.application.routes.draw do

# static page

get '/home/page1'

end

Rails - Database

1. Migration
   1. Create model (w/ migration)

# data type: string(default) , text, date, time, datetime, timestamp, created\_at & update\_at & id also created

rails g model Model col1 col2:integer

# > db/migrate/dateTimeStamp\_create\_models.rb

class CreateModels < ActiveRecord::Migration

def change

create\_table :models do |t|

t.string :cal1

t.integer :cal2;

end; end; end

# limit -> integers, string, binary, {}

rails g model User name:string{50}

# limit -> decimal, {total digits, decimal digits}

rails g model Product price:decimal{5, 2}

# limit -> uniqueness

rails g model User name:string{50}:uniq

# only migration

rails g model User name:string{50}

* 1. Migration – add/remove field

# add a field

rails g migration AddTestNumToProducts test\_num:integer

# > db/migrate/add\_test\_num\_to\_products.rb

class AddTestNumToProducts<ActiveRecord::Migration

def change

add\_column :products, :test\_num, :integer

end; end

# add filed with index

rails g … test\_num:integer:index

… add\_index :products, :test\_num …

# ../remove\_test\_num\_from\_products.rb

rails g migration RemoveTestNumFromProducts test\_num:integer

# => .rb

class RemoveTestNumFromProducts < A…::Migration

def change

remove\_column :products, :test\_num, :integer

end; end

* 1. Migration – add/change table

# add a table

rails g migration CreateUsers name

# ../create\_users.rb

class CreateUsers < ActiveRecord::Migration

def change

create\_table :users do |t|

t.string :name

end; end; end

# changa table

…

change\_table :products do |t|

t.remove :description, :name

t.string :part\_number

t.index :part\_number

t.rename :partcode, :part\_code

end; …

# changa column, =>interger

…

change\_column :products, :part\_number, :integer; …

* 1. change method

add\_column add\_index add\_reference add\_timestamps add\_foreign\_key create\_table

create\_join\_table drop\_table (must supply a block)

drop\_join\_table (must supply a block) remove\_timestamps rename\_column

rename\_index remove\_reference rename\_table

* 1. Execute migrate

rails db:migrate VERSION=<stamp> # default all

# rollback first, then edit. **Better** to gen new migrations

rails db:rollback STEP=5 # default 1

1. Add/remove data

# with migration

class AddInitialProducts < ActiveRecord::Migration

def change

5.times do |i|

Product.create(name: "Product ##{i}",

description: "A product.")

end; end; end

# with db/seeds.rb

5.times do ...

# with db/seeds.rb better

data = [[col1\_data1, col2\_data1, ...], ...]

data.each do |col1, col2, ...|

Product.create(col1:col1, col2:col2, ...); end

# then execute

rails db:seed

# with instance

p = Product.new; p.<col1> = col1\_data, ... ; p.save

p.new\_record? # true / false => UNsaved / saved

# remove

.destroy Product.first.destroy

1. Methods

.all .first(all.first) .last(all.last)

.limit(n)

.order(:col) .reverse\_order # order

.average(:col).to\_s # avg

.count .maximum .minimum .sum

1. Query

# default primary key

.find(1) .find([1, 3, 4]) # multiple

# .where(col:condition), can be multi

.where(name:"a") .where(num: 1..999)

.where(name:["a", "b"], num: 1..999).first

.where('name like ?', '%an%') # sql

.where('num > ?', '100')

.where('name like %an% and num > 100'')

1. execute sql

Product.connection.execute('UPDATE `products` SET `price`=`free` WHERE `part\_number`=1’)

1. Validation, one-to-many, callback function: /app/models/<src>.rb

class Country < ActiveRecord::Base

validates :name, presence: true, length: {in: 1..20}

# one to many relationship

has\_many :lineitems

# callback function, called before 'destroy' method

before\_destroy :make\_sure\_no\_line\_items; end

# sample items

length: {minimum: 2 , too\_short: 'Warning'}

numericality: {greater\_than\_or\_equal\_to: 0.01}

uniqueness allow\_blank

format: {with: %r{\.(gif|jpg|png)\Z}i,

message: 'must be GIF, JPG, PNG images'}

1. Rails irb

# start rails irb

rails console

# check environment (development/production...)

Rails.env

# check application path # connect db

Rails.root Product.connection

1. Be able to use generator to generate different entity including controllers, models, migrations Be able to explain Ruby on Rails default directory structures,
2. Understand Ruby on Rails routing, including resources route
3. Understand functions of each basic controller actions, like Controller#index, Controller#show, Controller#new, Controller#create, Controller#edit, Controller#delete
4. Understand basic functions of each action’s corresponding views, and differences between them. How parameters/values are passed to controllers and views.
5. Understand Ruby on Rails’s convention over configuration philosophy, be able to use example to explain this philosophy
6. Understand purpose of functions like validates, before\_destroy, before\_action.
7. Understand rails asset pipeline
8. Know database table 1-many relationship and how to use ORM to do basic database query.
9. Basic concept of data issue and web application security (lecture 10.2)