Conventions:

[optional]  
italics means fill it in (using the convention the holder uses)  
cc\_<\*> means use camelCase  
sc\_<\*> means use snake\_case  
ec\_<\*> means CapitalizeEveryWord  
ss\_list\_<\*> means space separated list

Create a project

rails [*version*] new app\_name

edit Gemfile  
bundle install [--without production]

bundle update

Set Up minitest

edit test/test\_helpers.rb to include

require "minitest/reporters"

Minitest::Reporters.use!

Set Up Guard

bundle exec guard init

put the standard code into GuardFile

make sure spring/ is in .gitIgnore

In a separate terminal:

bundle exec guard

Naming Conventions

controllers: camel

Page Test statements

get sc\_controller\_action\_url

Special test methods

setup

Page Asserts

assert\_response status

status can be

:success

:redirect

:missing

:error

assert\_select htmltag, expected

checks that there is an html tag   
 matching the expected value

Misc. Rails Commands

rails test[:models]

rails test[:integration}

rails console [--sandbox]

rails server [-b ip -p port]

rails generate controller cc\_name ss\_list\_actions

* generates sc\_name\_controller.rb
* for each action, generates action.html.erb in view/sc\_name

rails destroy controller cc\_name [ ss\_list\_actions ]

rails generate model ec\_name [attr:type]

* can add multiple attributes space separated

rails generate integration\_test sc\_test\_name

* generates sc\_test\_name\_test.rb in test/integration

Important Directories

/app/controllers

* ruby files for each sc\_controller

/app/views/

* html.erb files for each sc\_controller

/app/views/layouts

* html.erb files containing shared formatting

/test

/config

* config info like routes.rb

/app/assets

* assets specific to this application

/lib/assets

* assets for libraries written by us

/vendor/assets

* assets for third-party vendors

DB Actions

rails db:migrate [VERSION=number]

rails db:rollback

|  |  |
| --- | --- |
| assert\_select "div" | <div>foobar</div> |
| assert\_select "div","foobar" | <div>foobar</div> |
| assert\_select "div.nav" | <div class="nav">foobar</div> |
| assert\_select "div#profile" | <div id="profile">foobar</div> |
| assert\_select "div[name="yo"] | <div name-"yo">hey</div> |
| assert\_select "a[href=?]",'/' count:1 | <a href="/">foo</a> |
| assert\_select "a[href=?]",about\_path, text:"foo" | <a href="/about">foo</a> |

Model Objects

myUser = User.new(name:"me", email:"me@gmail.com")

* creates it in memory

myUser.save

* writes it to the db

myUser = User.create(name:"me", email:"me@gmail.com")

* creates it and writes it to the db

myUser.destroy

* removes it from the db (still in memory)

Ways to find an object

* User.find(1) #by id
* User.find\_by(email:me@gmail.com)
* User.first
* User.all

myUser.reload.email

* reloads the specified field from the db

myUser.update\_attribues(hash of attributes)

* updates and saves changes

validates :varname [, presence: true] [, length: {maximum: int}]

[, format: {with: regex}] [, uniqueness: [true | case\_sensitive: false]]

Model Object Callbacks

before\_save { ruby code }

Routing

Rails.application.routes.draw do

root ‘application#hello’

end

each row maps a url ending to a controller action  
other legal formats:  
 get ‘sc\_controller\_name/action’

get ‘/url\_end' to: 'controller\_name#action’

root ‘sc\_controller\_name#action’

To make a routes that use a record from the db

resources :dbTableName

This give us these RESTful routes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **HTTP Request** | **URL** | **Action** | **Named Route** | **Purpose** |
| GET | /dbTableName | index | dbTableName \_path | Page to list all rows |
| GET | /dbTableName/id | show | dbTableName\_path(id) | Page to show one row |
| GET | /dbTableName/new | new | new\_dbTableName\_path | Page to make new row |
| POST | /dbTableName | create | /dbTableName\_path | create a new row |
| GET | /dbTableName/id/edit | edit | edit\_dbTableName\_path(id) | page to edit a row |
| PATCH | /dbTableName/id | update | dbTableName\_path(id) | update row |
| DELETE | /dbTableName/id | destroy | dbTableName\_path(id) | delete row |

Model tests

special methods: setup

test "test name" do

#stuff

end

things you can assert

* @user.valid?

Embedding Ruby in html

<% ruby code %> #just execute it

<%= ruby code %> #execute and put result in html

provide(:symbol value) #attaches value to symbol

yield :symbol #get the value back

<%= render 'layouts/file' %> #go get layouts/\_file.html.erb

# the \_ means it is a "partial"

<%= link\_to "name", named\_path\_from\_routes\_file %>

#link to internal site page

General Ruby

Comments start with # and go to the end of that line

Single quoted strings will have no interpolation - literals

|  |  |
| --- | --- |
| **Action** | **Code** |
| giving a variable a value | @var\_name = value |
| Using a variable in a string: | “string containing #{@var\_name}" |
| Outputting a string with new line | puts "string" |
| Outputting a string without new line | Print "foo" |

Ruby Control

if condition

thenBlock

else

elseBlock

end

if condition

thenBlock

elsif condition

elseBlock

. . .

end

Conditional Actions:

action if condition

action unless condition

Blocks

arrays or ranges have methods that accept blocks

rangeOrArray.each {|varName| statement}

rangeOrArray.each do |varName|

stuff to do

end

#call a method on each thing in an array or range

rangeOrArray.each ($:methodname)

Other block accepting methods:

map #build array with the results

Ruby Methods

def sc\_name(var = default)

. . .

end

Ruby Classes

< is equivalent to extends in class declaration

self is equivalent to this

Ruby String Methods

.length

.empty?

.nil?

.toS #convert anything to a string

.split() #splits on spaces

.split('c') #splits on the char c

Ruby Arrays

indexes wrap instead of going out of bounds

.length

.empty?

.include?

.sort

.reverse

.shuffle

# for previous 3, add ! to make it change the array itself

.push(x) # add x at the end of the array

a << 7 # same as a.push(7)

.join #convert array to string, param can specify separating string

can index by a range: a[6..9]

%w #make string array

Ruby Default Validators

validates :terms, :acceptance =**>** **true**

validates :password, :confirmation =**>** **true**

validates :username, :exclusion =**>** { :**in** =**>** %w(admin superuser) }

validates :email, :format =**>** { :with =**>** %r\A([^@\s]+)@((?:[-a-z0-9]+\.)+[a-z]{2,})\Z/, :on =**>** :create }

validates :age, :inclusion =**>** { :**in** =**>** 0**..**9 }

validates :first\_name, :length =**>** { :maximum =**>** 30 }

validates :age, :numericality =**>** **true**

validates :username, :presence =**>** **true**

validates :username, :uniqueness =**>** **true**