**My own notes**

It seems like these two design patterns using blocks show up a lot. They seem equivalent.

For example:

f.each\_line {|line| p line}

OR

f.each\_line do |line|

STUFF ON MULTIPLE LINES WITH line

end

Normally, irb will print a lot of output. E.g. if you iterate through an array or hash and do something very little, irb will carry out your command but will also print the *entire* array or hash at the end. To suppress this output:

conf.echo = false

*Something on Canvas, just bits of Nokogiri code*

page = 'https://www.cia.gov/library/publications/the-world-factbook/'

index\_page = Nokogiri::HTML(open(page))

Countries\_HTML = index\_page.css('div.option\_table\_wrapper').css('option')

*HTTPS and Windows*

If you're still having trouble with validating certificates on Windows, try using the 'certified' gem - it does essentially what some of the other solutions do and does it in a nicer/cleaner way.

gem install certified

and in your .rb file - require 'certified'.

https://rubygems.org/gems/certified

https://github.com/stevegraham/certified

*Sublime and Ruby indenting*

I think in sublime, you can set as following.

// The number of spaces a tab is considered equal to

"tab\_size": 2,

// Set to true to insert spaces when tab is pressed

"translate\_tabs\_to\_spaces": true,

And specify settings for a certain file type, for example, Python, you should place them in Packages/User/Python.sublime-settings.

https://www.sublimetext.com/docs/2/indentation.html

*To create a project with an old version of rails:*

rails \_3.2.19\_ new my\_website

*List of commands you can follow to create a new project and push it to GitHub. Unfortunately it’s pretty buggy, but there are some clues that can be helpful*

rails \_3.2.19\_ new physic

rails generate scaffold doctor NPI:string first\_name:string middle\_name:string last\_name:string suffix\_name:string gender:string age:integer practice\_location\_street\_info:string practice\_location\_suite:string practice\_location\_city:string practice\_location\_state:string practice\_location\_zip:string practice\_location\_phone:string specialty1:string specialty2:string specialty3:string

rake db:migrate

git init

This automatically created a .gitignore file, but to this, I concatenated github.com’s “suggested” Rails .gitignore file

git add --all

git commit (used this for commit message: initial commit physic basic scaffold)

In Gemfile, made these changes

Line 10: gem ‘twitter-bootstrap-rails’

Line 11: gem ‘less-rails’

Uncomment this: gem ‘therubyracer’, :platforms => :ruby (this is within group :assets do / end)

git add Gemfile

git commit (used this for commit message: altered Gemfile to work with Boostrap)

bundle install

rails g bootstrap:install less

rails g bootstrap:layout application fixed (said Y to overwrite request)

rails g bootstrap:themed doctors -f

git add --all

git commit (used this for commit message: Made pretty with Bootstrap)

Then I went to github.com and created a repository with the name ‘temp’

git remote add origin https://github.com/rohitgupta3/temp.git

git push origin master

git branch makeusers

git checkout makeusers

rails g scaffold user name:string, email:string, username:string, password:string

**Whoops! I accidentally made these comma-delimited because I was copying it from a text document**

rake db:migrate

**This created a massive error because of the above comma-delimiting issue, so I discarded changes since the last commit by doing the below couple commands**

git checkout master

git reset HEAD --hard

git clean -fd

(from http://stackoverflow.com/questions/4630312/reset-all-changes-after-last-commit-in-git)

**TAKE TWO**

git checkout makeusers

rails g scaffold user name:string email:string username:string password:string

rake db:migrate

**This didn’t seem to work**

**TAKE THREE – THIS SHOULD WORK**

rails \_3.2.19\_ new physic

rails generate scaffold doctor NPI:string first\_name:string middle\_name:string last\_name:string suffix\_name:string gender:string age:integer practice\_location\_street\_info:string practice\_location\_suite:string practice\_location\_city:string practice\_location\_state:string practice\_location\_zip:string practice\_location\_phone:string specialty1:string specialty2:string specialty3:string

rake db:migrate

git init

This automatically created a .gitignore file, but to this, I concatenated github.com’s “suggested” Rails .gitignore file

git add --all

git commit (used this for commit message: initial commit physic basic scaffold)

**Stopped here**

*Creating tables, MVC etc. (from iteration 2)*

Dealing with tables + MVC through various rails generations

* rails g scaffold – this makes nice things so if you do “references” here it will flow through
* rails g migration AddColumnsToPhysicians – if you do “references” in here, it doesn’t flow through e.g. if you create somewhere on the view to add these extra columns (like the “state” attribute of physicians which references the State table) then it will not save the given state.
  + What you have to do, for instance, is modify the controller to take in the extra parameters.

First create a migration (rails g migration …) – this only creates a model.

Then do rake db:migrate – this creates a table.

After this, still have to create a view and controller.

Physician is a model.

If you add columns to the Physician model you don’t need a new model, you just need to alter the existing Physician model.

*Facebook authentication notes*

Facebook resources (mainly RailsCasts):

* #235 OmniAuth Part 1, but this was revised to #235 Devise and OmniAuth (revised)
* #360 Facebook Authentication – using omniauth-facebook but not incorporating Devise
* <https://github.com/plataformatec/devise/wiki/OmniAuth:-Overview> - more helpful than RailsCasts, this is what I ended up using

What happened when I was trying to implement Facebook authentication:

* Exporting environment variables first, then running rails s, to do Facebook authentication doesn’t work
* However if I set the environment variables in the same line as the rails s command like so seemed to work: FACEBOOK\_APP\_ID="foo" FACEBOOK\_SECRET="bar" rails s
* In general the Facebook authentication RailsCasts didn’t seem to work easily (I think because they weren’t integrating with Devise)

**Approximate list of steps from** [**https://github.com/plataformatec/devise/wiki/OmniAuth:-Overview**](https://github.com/plataformatec/devise/wiki/OmniAuth:-Overview)**:**

rails g migration AddColumnsToUsers provider uid

rake db:migrate

In config/initializers/devise.rb: config.omniauth :facebook, "APP\_ID", "APP\_SECRET"

In app/models/user.rb: devise :omniauthable, :omniauth\_providers => [:facebook]

What does this mean?

After making a model named User omniauthable and if devise\_for :users was already added to your config/routes.rb, Devise will create the following url methods:

user\_omniauth\_authorize\_path(provider)

user\_omniauth\_callback\_path(provider)

Added this to the first line of the <body> in app/views/layouts/application.html.erb

<%= link\_to "Sign in with Facebook", user\_omniauth\_authorize\_path(:facebook) %>

config/routes.rb, replaced

devise\_for :users, controllers: {registrations: 'registrations'} # The second one is the name of MY controller.

WITH

devise\_for :users, :controllers => { :omniauth\_callbacks => "users/omniauth\_callbacks" }

**stopped writing here but there’s more**

**From book “Engineering Software as a Service”**

Chapter 2:

Web application server typically uses Web application framework

We will be using the Rack application server (comes with Rails) with the Rails framework

WEBrick is a lightweight Rails server I think?

What does WEBrick do, and what does Rack do? I assume WEBrick is the Web server and Rack is the application server right? The application server sits between the Web server and the logic tier (where the application code is), so the application code developer doesn’t have to see the low-level HTTP details and deal with incoming HTTP requests.

MVC:

“Each controller corresponds to one model”. This can’t be right. For instance, you might have a Movies model and an Actors model, so what if you have a form in which the user can update the name of an actor in a particular movie?

Figure 2.9, different Web app architectural patterns in terms of models, views, and controllers – **this is a bit confusing as I only know the MVC I’ve implemented in the past.**

A route associates a URI plus an HTTP method (e.g. GET or POST) with a particular controller and action (action = method or function in application code).

In Rails, the route mappings are generated by code in the file config/routes.rb **(?)**

Interesting: Critically, a RESTful interface simplifies participating in a Service-Oriented Architecture because if every request is self-contained, interactions between services don’t need to establish or rely on the concept of an ongoing session, as many SaaS apps do when interacting with human users via a Web browser. This is why Jeff Bezos’s mandate (Section 1.4) that all internal Amazon services have “externalizable” APIs was so forward-looking.

In 2.9, the third pitfall says to focus on horizontal scalability. **What is this?**

**2-Sep-2014**

No homework announced till at least end of this week or early next week, will be due 10 days to 2 weeks after it’s posted.

Optional recitations on Friday led by TAs, not this Friday but almost certainly next Friday.

Info copied from Canvas:

Prof: Swapneel Sheth, swapneel@cis.upenn.edu, office hours Mon 11am Levine 268 or Levine 288?

TA: Pierre Rohel, email is prohel?

http://piazza.com/upenn/fall2014/cit597

Twitter: #cit597

**4 layers to Internet**

**Application** – where your applications run. This includes FTP, SMTP (and I think all protocols besides the low-level transport protocols (TCP and UDP) and the low-level Internet protocol (IP)).

**Transport** – has a bunch of different jobs. It only includes TCP and UDP. It includes application addressing (port numbers).

* Packets, segmentation – chopping up, say, a 10MB attachment to a file into bits that will be sent in separate packets. The transport layer needs to then know how to put them back in order.
* Flow control
* Congestion
* Error checking
* All of the above jobs can be summarized as end-to-end message transfer.
* TCP – Transmission Control Protocol, “reliable” delivery (but “slower”). Solve it with acknowledgement i.e. bi-directional communications (recipient tells sender that the packet was received successfully, otherwise packet is re-sent). TCP is built on top of IP to create reliable delivery (as well as doing the other functions listed above in the bullets). Use TCP for banking and other things where reliability is more important than speed.
* UDP – User Datagram Protocol, “unreliable” delivery (but “faster”). UDP can be used in games, videos, VoIP, etc. where dropping a frame doesn’t matter too much.
* The main difference is that TCP is reliable in the sense that it will give you guarantees that a packet is sent.

**Internet** – includes IP. 32-bit in latest version. v6 is bigger. IP does two things: (1) host addressing and (2) packet routing. Does packet routing by “best effort delivery”. I think this is inherently unreliable.

**Link** – underlying hardware

FTP – File Transfer Protocol. One of the earlier protocols.

SMTP – Simple Mail Transfer Protocol. Earliest form of email, essentially. IMAP and POP are advanced versions of SMTP.

Which packets goes to which application (e-mail, file transfer, Youtube streaming) is handled by the Transport layer, and this is done by doing application addressing using port numbers. Different packets go to different port numbers. Port numbers range from 0 – 65535. 0 – 1024 are reserved for “well-known” applications. There are commonly agreed on ports for certain uses. Port 80 is used for the Web. Port 21 is for FTP. Port 25 is for SMTP.

In Terminal: lsof -ni. Shows you which ports are open on your computer e.g. which ports are listening, which ports have established connections **(?)**. I think it’s in the form of TCP, my IP address:port -> Their IP address:port. If their port is HTTPS, then this is a well-known port number so the port number is not shown. E.g.

TCP **158.130.108.85:63556**->74.125.228.245:https (ESTABLISHED)

The bold is my IP address:port.

DNS – domain name system. Converts something that computers are good at to what humans are good at (numbers vs. words). DNS maps names to IP addresses. With DNS you actually want speed so it uses UDP. **Doesn’t that mean some small percentage of the time the website won’t load because your browser can’t get the IP address?**

www.cis.upenn.edu – the “edu” is a top-level domain (TLD). There is a centralized servers that lists all the TLDs (country-level, .edu, .com, etc.). First the request goes to the central server and asks where is edu. Then it goes to the edu domain server and asks do you know where upenn.edu is? Then it goes to the upenn.edu server and asks where is the cis.upenn.edu is? (I think he said that UPenn will house the upenn.edu server itself and handle those requests.) Then it goes to cis.upenn.edu and asks where is www.cis.upenn.edu, and it now gets the IP address back.

DNS is used in other contexts besides the Web, e.g. you can use it with FTP. If you SSH into a server using FTP and you use English language words to connect to it (because you don’t know the IP address), then that server you connect to is not running a Web server, but DNS is still involved. URLs do not need to pertain to the Web, is the point.

I think your ISP provides your DNS.

Said some interesting stuff about cyberbunker and Spamhaus. cyberbunker was labeled as spam so it decided to do a denial-of-service attack to Spamhaus. This is tied to the inherent vulnerability of DNS somehow **(but I didn’t catch how).**

Showed a talk by Leonard Kleinrock “Brief History of the Internet”.

**4-Sep-2014**

Have decided on office hours (think he’s talking about the TA). Office hours Wed 10-12

Professor’s office hours are Mon 11-1

Last time we talked about the four layers of the Internet

Application – FTP, SMTP, DNS

Transport – TCP, UDP, port numbers

Internet – IP

Link

ICANN maintains a lot of the Internet. One of their sub-committees is iana (Internet Assigned Numbers Authority) which has to do with numbering things on the Internet. They also do DNS stuff, I think they sort of maintain a registry of DNS servers. E.g. you can see which DNS server you hit when you go to .com (they show you that VeriSign owns it and gives you a list of IP addresses that I assume pertain to .com).

He goes to http://www.simpledns.com/lookup-dg.aspx to see how we get to, e.g., www.cis.upenn.edu - it first hits .edu, then that DNS server gives location of upenn.edu server, and so forth.

Dynamic DNS – this is a service such that any time your public address (the address of your server) changes, DNS servers are updated so that people are still able to hit your servers.

.com used to be owned by the DoD.

To handle conflicting DNS servers (two DNS servers pointing one domain name to different IP addresses), there are authoritative and non-authoritative DNS servers. Authoritative DNS servers have the true address – they dominate over everything else. If I try to point google.com to my laptop, I won’t be given authoritative status (think I need to be given this by Google).

World Wide Web consists of 3 important parts. Four, if you include DNS (not as crucial, just helps you avoid numbers). The three are:

* URIs, URLs, URNs
* HTTP
* HTML (and CSS)

1) URI, URL, URN

Think this pre-dated the Web but wasn’t used that much, and it made a lot of sense to combine this with the Web.

URN = Uniform Resource Name – analogous to ISBN numbers. But if you say “read this ISBN number”, you don’t know how to get to it. That’s where URL comes in.

URL = Uniform Resource Locator – tells you how to get to the resource you want.

URL

<scheme>:<hierarchical part>/[?query][#fragment]

*scheme* is the protocol / method to access. Most common is http. List of some schemes:

http, ftp, https, rtsp, file, mailto, tel, chrome, about, rtsp is the real-time streaming protocol

file retrieves a local file on your computer

tel is one way of specifying a phone number

chrome gives you access to Chrome’s **?**

about:(blank) is a blank page. Not sure (used in old IE days?)

*Hierarchical part* – examples:

www.nytimes.com/opinion/krugman/…

www.nytimes.com/sport

*[?query]* – this is optional, you can send info to get certain info back

The query is key-value pairs separated by &, e.g.:

?name=Swap&dept=cis

*[#fragment]* – this is optional, you can get a piece of a page (anchor tag?), e.g.:

#section1

Also optional – you can specify the port number, e.g. www.nytimes.com:8000/. What this says is: make a connection using http on port 8000. It will default to port 80. You might do this for Web development and obfuscation. If you’re developing a site, and it’s public, and you worry that people might do weird things on port 80, then move it to another port. This port number is the port number on the **server**.

https is secure http, and the convention is to use port 443.

HTTP was designed to be stateless.

URI = Uniform Resource Identifier. Officially, it could refer to a URL or URN or both. It’s a generic catch-all.

URNs, in practice, may not be used. You can think of a URL as the way to get to a resource on a server, and the name of the specific thing you’re looking for is the URN (?).

Until recently, for URLs you had to use the Latin alphabet with no diacritics. Recently there has been a push to allow other characters e.g. Chinese characters and so on. This can be problematic in some cases, not because of http (which sounds like it’s been well-built for this). The problem of implementing this sounds like the DNS servers are not necessarily built to deal with this.

2) HTTP= Hyper Text Transfer Protocol

One of the most influential people in this was Tim Berners-Lee.

Good story about Roy Fielding / Richard Taylor also.

The most common HTTP actions are GET and POST.

If I’m a client and I want to get a page, I use a GET request.

POST is used for submitting information e.g. if I fill out a form with information and want it to be on a server somewhere.

There are also PUT and DELETE and others as well.

PUT says here is some resource e.g. picture and put this on your server.

DELETE says here is some URL that represents some server on your server, remove this from your server.

PUT and DELETE are not really used anymore because it’s dangerous. People will put and delete stuff from your server. Most Web servers won’t support these anymore.

**Sounds like we’ll be doing stuff in Rails.**

HTTP response codes – the server responds to your request and gives the client information.

He showed us http://tools.ietf.org/html/rfc3986#section-1.1.2 - stuff about protocols

He then showed us http://www.w3.org/Protocols/HTTP/HTRESP.html, a W3C page about the World Wide Web

Response codes of redirection are 3xx. E.g. response code of Moved 301 – most browsers are smart enough to redirect you.

Other HTTP response codes:

4xx – you (client) made a request that doesn’t exist, you did something that doesn’t make sense

5xx – I (server) made an error

First homework – we’re talking about it on Tuesday.

Funny – programmer Ryan Gosling.

**9-Sep-2014**

Unix didn’t always used to be free. But they decided to make it open-source and more accessible but the kernel was still missing. Linus Torvalds took up the challenge though.

For the homework we will have to log into eniac.

Popular – Putty. Lets you remote log into any machine using the SSH protocol.

ssh guptaro@eniac.seas.upenn.edu

ls -l

List with detail.

ls -R

This is a recursive list. The recursive ls will list everything in the current directory, and then list everything in each subdirectory.

mkdir 597

Make directory called 597

cd dirname

Change directory

pwd

Present working directory

pico filename.txt

Simple text editor – like vim but simple.

cat textfile.txt

Concatenates a file and prints it out to the screen

(He uses TextMate, a text editor for Mac. It does SSH tunneling so that he can edit files, e.g., on eniac.)

wc

Word count – shows you number of lines, words, and characters.

cd ..

Go back one directory

w

Shows you who is logged in and what they are doing.

finger (username)

Gives you more info about a person

/ - first directory

In /proc, you get info on processes, CPU usage, etc.

cd ~

Change to user’s home directory

Ctrl-R

Reverse incremental search

cp cv.pdf ../

Copy

mv cv.pdf 597/

Move

mv cv.pdf cv1.pdf

Using “move” to change the name of a file

man pwd

Look at man page for pwd. If they give multiple, enter a number for which option e.g. 1.

whatis ls

Tells you what something does (brief one-liner)

apropos user

Searches for commands that have something to do with “user”, I think this searches the whatis brief one-liners.

scp weka.log guptaro@eniac.seas.upenn.edu:~/597/

Secure copy – can transfer a file from my MacBook Pro to eniac.

* This won’t work in Windows (though I think he said it would work in Putty). So instead, try to use WinSCP which is a graphical version of doing this.
* On the Mac he likes using Cyberduck. You can transfer files back and forth, I think you can also edit files remotely (or at least preview them?).
* Another one someone likes is Fugu.

HTML – Hyper Text Markup Language

**11-Sep-2014**

Permissions:

-rw-r-----

The first dash stands for the directory – ignore this for now

Nine spaces after that, three groups of three

First group: owner

Second group: group

Third group: other (rest of the world)

Group is a group with my name. e.g. if you’re collaborating on a project, you can add these colleagues to a group.

Permissions: rwx, read write execute

Chmod (permissions) (file(s))

Change permissions. “Change mode”

*Permissions part:*

u / g / o

+ / -

r / w / x

e.g. chmod o+rw test.html

You can also consider rwx as three bits. E.g. rw- is 110 = 6 in binary. So you can use it like this.

chmod 646 test.html is equivalent to making permissions equal to 110100110

755 is common permissions, rwx for myself and then r-x for the group and others

644 is common too, rw-r--r--.

The reason the below file was not visible in a browser was that it was originally 640 i.e. others cannot even read it. So the web server on which this file is on cannot read it, so you will get a “Forbidden” when trying to go to the file. So I should change it to 644 so that others can also read.

http://www.cis.upenn.edu/~guptaro/test.html

OR http://www.seas.upenn.edu/~guptaro/test.html

chmod -R will go recursively through directories and change permissions

In HTML, to let the browser know it’s HTML, put <!DOCTYPE html> at the top

Normal structure:

<!DOCTYPE html>

<html>

<head>

<title>My awesome webpage!</title>

</head>

<body>

This is a test page

</body>

</html>

*Stuff on tags*

<h1> to <h6> are valid. <h1> is really big / important, <h2> less so, etc.

<p> - new paragraph

<b> bold, <i> italic

HTML5 is bringing back bold and italic tags, but they recommend giving more semantic information e.g. do you want to emphasize this text <em> or do you want to make it strong <strong>. They say use <b> and <i> if you want to use bold and italics but not give any semantic meaning (??).

HTML is not a programming language.

Comments are

<!--

Put comments here

-->

Images. Source can be local or anything on the Internet (using the URL).

<img src=”cat.jpeg” />

If your picture is not showing, make sure the permissions are ok. E.g. if others can’t read the image file, it won’t show up in the web browser.

In general you end up using relative paths because that way you can move your entire html folder somewhere else e.g. Prof. Sheth moved his entire html stuff from Columbia to Penn and didn’t have to change a single line.

Styles

<p style=”color:red;”>

They recommend you to do things with ‘style’ and set your colors, fonts, etc. In the past you could do a color attribute, font attribute, etc. so don’t do this.

CSS – Cascading Style Sheets

Anchor tag for link

<a href=”LINK”>content</a>

The LINK can be a URL for a webpage or can be an internal (local) file. If you do an external website, you **ALWAYS** start with HTTP. Otherwise it assumes it’s a local file.

Line break

<br />

Lists

Unordered list: <ul> </ul>

Ordered list (numbered): <ol> </ol>

<ul>

<li> First list item </li>

<li> Second list item </li>

<li> Third list item </li>

</ul>

Within the <ul> tags (or <ol> tags), put <li> for each list item.

Tables:

<table> specifies the table, <tr> is each row, <td> is each piece of data

<table>

<tr>

<td> First one </td>

<td> Second one </td>

<tr>

<td> Third </td>

<td> Fourth </td>

</tr>

</table>

He recommends W3Schools as a good reference for HTML and also W3C (maintains standards).

**Swap posted this on Canvas on 11-Sep-2014**

**Want to know if your HTML is in good order?**

W3 will check your HTML to make sure you have open and closing tags, that you haven't forgotten your <!DOCTYPE html> declaration at the top, and more! Just copy your HTML here and it will do the rest!

Note: Some "errors" are really just suggestions. For example, if you don't have an alt attribute for an <img> tag, it will complain. You don't need an alt attribute, but it's helpful!

The “here” above points to http://validator.w3.org/#validate\_by\_input**12-Sep-2014 recitation**

**Pierre Rohel posted this on Canvas on 12-Sep-2014**

**Recitation + Tables in html**

I uploaded the files I used today on canvas. Just put the 2 files in the same folder and that should work. Check the css path at the top if you have styling issues.

There is something I wanted to say about tables during the recitation and then forgot.

- The size of the cells in your table is always defined by the cells in the first row. All cells in the other rows will just match the size of the one above.

- You can have a <th> tag instead of <td> on the first row, it stands for table header and aligns the text in the center by default and set it bold.

- You can have a cell taking the space of 2 normal cells if needed, I encourage you to read more on w3schools about the colspan attribute (and other attributes if you want to explore!)

**Pierre Rohel posted this on Canvas on 15-Sep-2014**

**CSS: "tag1 > tag2" versus "tag1 tag2"**

A clarification regarding the recitation:

- if you use the "tag1 tag2" notation, you select all the tag2 that are nested in tag1, no matter the depth of tag2.

- if you use the "tag1 > tag2" notation, you select all the tag2 that are direct children of tag1.

Example:

<li>

<p><a></a></p>

<a></a>

</li>

In this example, "li a" in the CSS will catch both a tags, where "li > a" will catch only the second one because it is the only a tag that is a direct child from li.**16-Sep-2014 Tues lecture**

Said something about how if you don’t want to put the full URL for your webpage, you have to rename it to index (?).

*Inlining/embedding*

Style. To inline or embed something, put style attributes in, for instance style=”color:red;”. However if you want to do a certain style for multiple tags, it becomes unwieldy.

CSS can be done by:

* inlining/embedding
* internal
* external

*Internal*

If you put this within your <head> tags before the body, you can specify certain tags as always having a certain style. Think this is called “internal”.

<style>

h1 {color:blue}

p {color:green}

</style>

This is done instead of e.g. doing <p style="color:green;"> for every <p> in your body.

Think he said we’re not responsible for recitation content (?) but it will also be posted in Canvas.

*External*

Put this in your <head> tags before the <body>: <link rel="stylesheet" type="text/css" href="test-style.css">

And the test-style.css can have

h1 {color:blue}

p {color:green}

Permissions – put 744 so that others can read.

If you want to validate webpages, go to validator.w3.org/#validate\_by\_uri

*Ruby*

Map, reduce, filters – functional programming (e.g. you can do this in Python)

*Getting started with Ruby on Windows – 5 options*

* #1: rubyinstaller.org/downloads – one click and it will install
* #2: Since we’re doing Rails as well, there is a Rails version of this – railsinstaller.org/en. Probably the better option since we’re doing Rails later on in the semester
* #3: Cygwin which is a Unix-like environment where you can run almost all your Unix commands like Ruby and server stuff.
  + He recommends just installing it all (choose “install all”) i.e. get all the packages, let it go for three hours, and it should be fine.
* #4: can use the virtual machine (VM) that’s running Linux for which the details are in the CIT 597 textbook.
* #5: use pik which is a tool to manage multiple versions of ruby on Windows. github.com/vertiginous/pik
* Probably want to use Cygwin or VM (probably more the VM) so you can use the latest version of Ruby possible.

Use Ruby Version 1.9.3 at a minimum. For Mac / Linux, you can probably do better e.g. Version 2 which is a nicer version. 1.9.3 is fine but he strongly recommends 2 which has a lot of nice features.

To check Ruby version on Mac / Linux, type this in Terminal: ruby -v

He strongly recommends getting RVM (Ruby Version Manager). It will let you have multiple versions of Ruby all installed on the same computer and not conflict with each other.

Tried this to get RVM on my machine: \curl -sSL https://get.rvm.io | bash -s stable

I go this response:

Downloading https://github.com/wayneeseguin/rvm/archive/stable.tar.gz

Installing RVM to /Users/rohitgupta/.rvm/

Adding rvm PATH line to /Users/rohitgupta/.profile /Users/rohitgupta/.bashrc /Users/rohitgupta/.zshrc.

Adding rvm loading line to /Users/rohitgupta/.bash\_profile /Users/rohitgupta/.zlogin.

Fixing environment for \*.

Unknown ruby interpreter version (do not know how to handle): \*.

Installation of RVM in /Users/rohitgupta/.rvm/ is almost complete:

\* To start using RVM you need to run `source /Users/rohitgupta/.rvm/scripts/rvm`

in all your open shell windows, in rare cases you need to reopen all shell windows.

# rohitgupta,

#

# Thank you for using RVM!

# We sincerely hope that RVM helps to make your life easier and more enjoyable!!!

#

# ~Wayne, Michal & team.

In case of problems: http://rvm.io/help and https://twitter.com/rvm\_io

\* WARNING: You have '~/.profile' file, you might want to load it,

to do that add the following line to '/Users/rohitgupta/.bash\_profile':

source ~/.profile

Then check all the Ruby versions with this command: rvm list

*Ruby*

Ruby is:

* Interpreted.
  + I think he said interpreted languages are slower than compiled languages like C.
  + However, a lot of the internal libraries in Ruby are written in C so you get a benefit from that.
* Object-oriented.
  + Java is object-oriented but has non-object-oriented bits. But Java still has basic data types e.g. int, float (which are not objects); they have objects that wrap these primitives but these still exist. In Ruby everything is object-oriented – everything is an object. Even if you write the number 2, it’s represented by an object that you can perform methods on.
* Dynamically typed and strongly typed.
  + In Java, you say int a = 5
  + In Ruby, you say a = 5. a is still typed (it is an int) but this type can change (e.g. to a double). At run-time checks will happen and if you try to do illegal stuff you will get errors. This is dynamic typing. Similar to Python.
  + Also strongly typed. Unlike in C (C does whatever it wants), it will enforce type safety. If you have a string “hello” and a number 5 and try to add them, it doesn’t make sense. In C, it’s all bits and bytes and memory addresses, it will carry this out and give you garbage. In Ruby it will not do this – you will get type errors if you try to do something that’s not allowed.
* Multiparadigm
  + Can do imperative stuff in Ruby if you want (if you’re coming from the world of C, for instance).
  + If you’re coming from the object-oriented world (e.g. Java and C++) you can do that in Ruby.
  + If you like functional programming, e.g. if you don’t want to use any variables, or if you want to use map / reduce / filter, or if you want functions to call each other, you can do this also.
* High-level data types
  + E.g. all the data types you love in Java (util something?) you get these too. Also some are more flexible than Java. If you want an array to have integers and want it to be dynamically sized, you can’t do this in Java easily. In Ruby, it automatically says, “Oh you want to add 10 more elements to this array? No problem”. Also if you want an array to have an int and string and other stuff, you can do that also.
* Only negative thing for Ruby vs. Java. In Java, documentation is slightly better, more precise, and more up to date. In Ruby the documentation is good but not as good.
* Because Ruby is interpreted, dynamically typed and so on, it has a notion of a REPL (read-eval-print loop). You don’t need an entire .java file, for example, with public something, etc. You can just try little snippets, one line at a time. In Ruby this is called irb (Interactive Ruby Shell). So if you type “irb” at the Terminal assuming you have Ruby installed you can do this.
* Anything you do in Ruby will return some value (obviously don’t have to use this value).
* Don’t need semi-colons at the end.
* X = 5; y = 10; y = x; x = 10
  + Think of this more like Java. It’s copying the value over (y will equal 5). Don’t think of pointers and memory and stuff.
* print “Hello World”; puts “Hello World”; p “Hello World”
* Using variables in printing: puts “The value of y is #{y}”. Or: puts “The value of y times 3 is #{3\*y}”
* Apparently in Java if you multiply two large int’s and you get overflow you might get negative numbers so you have to deal with changing the type to have larger storage. (Don’t have to deal with this in Ruby because it will automatically create larger and larger types for you).
* Duck Typing. We don’t need to worry so much about types. The reason we call it “Duck Typing” – if it looks like a duck and it quacks like a duck, I can treat it as a duck. Example file: duck-typing.rb in ENIAC:html/

Defining function, use “def” e.g.: def add(a, b)

To end a function you need the keyword end

Ruby convention is to tab indent two spaces.

Btw don’t even need “return” at the end of a Ruby function because every statement returns a value so by default the last statement return value is what the function returns.

Example of short script that defines and calls a function:

def add(a, b)

a+b

end

p add(3, 4)

p add(2.2, 5.6)

He is using TextMate to run all these Ruby scripts.

Arrays in Ruby: [1, 2, 5]

If you multiply a string by 3, it will concatenate the string to itself 3 times. Same for multiplying an array by an integer.

Summary

Functions: use def and end, don’t need curly parentheses.

Don’t need explicit types.

Don’t need to explicitly “return” something in functions, the last line will return whatever.

Underscore (not Camel Case) is the convention for Ruby.

Default values e.g.

def add(a, b=10)

(a+b)

end

p (5, 3)

p (3.2)

The first p will print 8, the second will print 3.2. b’s default value is 10. This way you can have 1 or 2 arguments. If you have two arguments, it adds them. If you have one argument, it adds 10 to this.

**18-Sep-2014**

<https://www.ruby-lang.org/en/> - has some good resources. Go to “Documentation”

“Programming Ruby” – this is the book that made it really famous

A fun / good way to learn Ruby – “Why’s (Poignant) Guide to Ruby”

Check “Ruby From Other Languages” also

“Editors and IDEs” – he recommends TextMate for Mac but we’re free to use anything.

*Data structures*

* Arrays.
  + Can dynamically re-size. Also, arrays are not typed so you can put whatever you want in arrays.
  + a = []
  + a = [1, 2, 3, 4, 5, “hello”]
  + a.reverse will return the array in reverse order
  + a[10] will return nil
  + a[-1] returns the last element, a[-2] is second to last, etc.
  + a[1, 3] will return an array that starts from a[1] and has 3 elements
  + a[-3, 2] returns the array starting from a[-3] and having two elements
* Ranges
  + a[3..4] will give you an array from a[3] to a[4]
  + my\_range=(1..10); my\_range.to\_a; This will give you an array of 1 to 10. Can obviously just do (1..10).to\_a
  + (‘a’..’p’).to\_a gives you an array of the letters from “a” to “p”
  + (‘hello’..’kelly’).to\_a – this will literally iterate over each letter to get from ‘hello’ to ‘kelly’
  + (‘a’...’z’).to\_a – this has 3 dots, and therefore will not include ‘z’.
* Hashes
  + Do them with {}
  + No types so you can have anything hash to anything

my\_hash = {

“cello” => “string”,

“flute” => “wind”,

“drum” => “percussion”,

“violin” => “wind”

}

* + my\_hash[“drum”] will return the value mapped to by the key “drum”
  + my\_hash[“oboe”] = “wind” will add this key-value to the hash, and if the key already exists in my\_hash it will replace the value
  + Can explore this last method: ri Hash#[]=

Boolean values is the usual thing, true and false. Difference from C: 0 is true. The only thing in Ruby that’s false is false and nil **(I think).**

Operators: elsif is one word.

Also has “unless”. Instead of saying “**if** a is not equal to b do something” you can say “do something **unless** a is equal to b”.

Also has “until”.

You can use “and” and “or”, you don’t need && or ||

Comments: #

Multi-line comments are not very common

Loops / if, else

End “while” loops with “end”. Also end “if” statements with “end”. You need “end” for most things with a couple exceptions (while, for, etc.)

If you want to do an entire “if” statement on one line, use “then” e.g. if (a < 5) then a++

Can also do: b=c if a==5

Can also do: b=c unless a==5

Otherwise, just use a new line.

Interesting syntax for loop:

3.times do

stuff

end

*This will do “stuff” three times.*

om number generator

rand(1..6) will return a random number from 1 to 6 e.g.

def roll\_dice

rand(1..6)

end

Without an argument, rand returns a number from 0.0 to 1.0 (**think inclusive)**

Is the convention to use “puts”? That’s what he seems to use a lot.

Iterating over an array a. Can do a.foreach, or maybe a.each, some such thing. People don’t usually iterate over the index.

You can do:

puts “The computer’s score is now #{score + round\_score}”

i.e. you can add variables within a string and it will print the final result

To get user input:

again = gets

again.chop!

*gets*: will get user input. The chop! method removes leading and trailing whitespace and line breaks, etc. and maybe some other stuff.

Methods that end with an exclamation like chop! will mutate the state **(I think he called them “destructive methods”)**. For instance in:

again = gets

again.chop!

The second line will chop the variable again and then actually change again.

Btw, these two lines are identical:

again = again.chop

again.chop!

Similar idiosyncrasies – methods that end with a ? return a Boolean **(I think)**. For instance, on a number you can call the method is\_prime?

He calls “gets” “get S” and “puts” “put S”

*Blocks*

A block is a group of code that’s together in some way.

Use blocks in conjunction with “yield”.

def three\_times

yield

yield

yield

end

three\_times { puts “hello” }

You can think of the block as an argument you’re passing to the method here (BUT – smthg smthg smthg).

You can arbitrarily pass variables from your method to your blocks. **I think he is going to expand on this later.**

In Ruby, you can swap values via: a, b = b, a

Fibonacci using blocks and yield.

def Fibonacci(max)

i1, i2 = 1, 1

while (i1 <= max)

yield i1 # yield the first value (i1) to my block, the block does what it wants with it

i1, i2 = i2, i1 + i2

end

end

Fibonacci(100) {|x| puts x} #this notation says |x| takes whatever value is yielded to it (?), and then the “puts x” says what you want to do with it

**He showed us a way to see all the methods you could do on an array but I missed it. I think you just have to tab-complete but it didn’t work on mine. I think he said to make it work you might need to change your irbrc file.**

Good resource:

rdoc

ri – command that will give you a lot of documentation about in-built classes, methods, etc. (in the shell not in irb)

ri Array

ri Array#[] – look up the [] method.

ri -i – this is interactive mode. Then you can say e.g. Array#c and it will give you a list of array methods that start with “c”.

Swap / Pierre Rohel posted this on Canvas on Thursday

Range steps

After Spriha's question in class, here are some elements about range steps:

- there is actually a step function on ranges to specify the step you want. Example:

001:0> (1..20).step(4).to\_a

=> [1, 5, 9, 13, 17]

- It won't let you give a negative step here though, so you can only increment.

- Interesting fact, the numeric class also has a step function. You can apply step to a number to generate all numbers up to an other one, and give it a step.

003:0> 2.step(10).to\_a

=> [2, 3, 4, 5, 6, 7, 8, 9, 10]

004:0> 2.step(10, 3).to\_a

=> [2, 5, 8]

- And this last method will let you have a negative step:

006:0> 20.step(-5, -3).to\_a

=> [20, 17, 14, 11, 8, 5, 2, -1, -4]

Swap: You can also do downto (which is useful for loops) with numbers.

(from ruby site)

=== Implementation from Integer

------------------------------------------------------------------------------

int.downto(limit) {|i| block } -> self

int.downto(limit) -> an\_enumerator

------------------------------------------------------------------------------

Iterates the given block, passing decreasing values from int down to and including limit.

If no block is given, an Enumerator is returned instead.

5.downto(1) { |n| print n, ".. " }

print " Liftoff!\n"

#=> "5.. 4.. 3.. 2.. 1.. Liftoff!"**23-Sep-2014**

Homework due Thurs Oct 2, midterm Tues Oct 7

def fibonacci(max)

i1, i2 = 1, 1

while (i1 <= max)

yield i1

i1, i2 = i2, i1 + i2

end

end

fibonacci(10000000000000000) {|x| puts x}

Whatever is in the yield statement in terms of variables, will be stored temporarily in the block variables that are stated in the beginning e.g.

yield i1

This will pass the value of i1 to the x variable which is in {|x| puts x}

If you do this instead:

yield i1, i2

And in the block you have {|x, y| puts “x is #{x} and y is #{y}”}

The beauty of this is that the block does not need to be hardcoded in the fibonacci method. So anyone who is calling the fibonacci method can pass their own block to it, and the values that we care about are passed to the block.

Yield calls the block, but there is a lambda/closure concept here, and a scope issue, etc. We’ll get to that on Thursday.

Default values in Ruby are either nil or 0, he can’t remember.

You can use certain iterators in Ruby using a block.

fruits = [“strawberry”, “mango”, “avocado”, “banana”]

fruits.each {|f| puts f}

There is also a method called each\_with\_index in case you need the index e.g.

fruits.each\_with\_index do |fruit, index|

puts “Fruit #{fruit} is at index #{index}”

end

p Array.new(3) – create a new array with nil in each position (3 positions)

p.Array.new(3) { 3 } – create a new array with 3 in each position (3 positions)

p.Array.new(3) { Array.new(3) } – an array with 3 arrays of 3 nil’s each

p Array.new(3) { Hash.new } – an array with an empty hash in each position

p Array.new(3) { |index| Array.new(3) {index} }. This gives you [[0,0,0], [1,1,1], [2,2,2]]

p Array.new(3) {|index| index} – this will return [0, 1, 2]. The idea I think is that the “new” method yields the index variable. If you don’t use this variable in the block that’s fine, but in this case I did.

*Regular expressions*

=~ is the fastest way of doing regexp matches e.g.

“cats and dogs” =~ /do/

This will return 9, which is the index where the first occurrence of the search string is found.

“cats and dogs” =~ /ee/

This returns nil because it’s not found.

Digits

“cats are 5 and dogs are 9” =~ /\d/

“55 hello 44” =~ /\d/

This returns 0

Can do something like this:

a = “cats are 5 and dogs are 9”

puts a if a =~ /\d/

.match method

“55 hello 44”.match(/\d/)

This returns an object of type MatchData

“55 hello 44”.match(/\d+ (.\*) (\d+)/)

This is saying: match 1 or more digits, then one or more things in the round parens, and finish with one or more digits.

I think this is actually matching the whitespace included in the regexp string.

“55 hello 44”.match(/(.\*)(\d+)/)

The (.\*) will be greedy and match up to the first 4. But it will be as greedy as possible but still try to match the d+, is what I think. In other words, it will be as greedy as possible with the constraint that \d+ is still valid.

“55 hello 44”.match(/(.\*?)(\d+)/)

The (.\*?) here will match as little as possible.

Returning the results of a regexp

my\_match = “55 hello 44”.match(/\d+ (.\*) (\d+)/)

my\_match.inspect

my\_match[0]

require ‘open-uri’

require looks for 3 things

It will look for a class with that name. It will look for the class in the path where you’re running your Ruby file from, otherwise it will try to find other Ruby classes that are part of the standard library, and otherwise if you have your own Gem (external third-party library?) it will look there.

require 'open-uri'

my\_uri = "http://www.cis.upenn.edu/~swapneel/test.html"

open(my\_uri) do |site|

p site.status #200 OK, for instance

p site.last\_modified

p site.content\_type

p site.charset

site.each\_line do |line|

p line

end

end

each\_line will take each line and yield each one in a block

HW:

If he replaced “South America” with “Europe” it should be an easy fix. Have to be abstract to the level of continent, for instance.

Think of “South America” as an argument to the method. You don’t have to make it so that the user has to input “South America” but this could be a way to show that it’s not hard-coded.

On the CIA World Factbook website, there is a drop down that says “look at the low bandwidth version”, and it’s much, much easier to parse.

**On 23-Sep-2014 (today), Swap posted this on Canvas**

If you're using rvm, it won't install the ri docs for you automatically.

Here's info on how to generate them.

## Examples

Generates all ri, rdoc and gems documentation for the current ruby:

rvm docs generate

Generates only ri documentation for the current ruby:

rvm docs generate-ri

Generates only rdoc documentation for the current ruby:

rvm docs generate-rdoc

Generates only gems documentation for the current gemset:

rvm docs generate-gems

**On 23-Sep-2014 (today), Swap posted this on Canvas**

A correction about regex - if your regex contains a space, it will match it.

2.1.2 :001 > "55 hello 44".match(/\d+ (\w+) \d+/)

=> #<MatchData "55 hello 44" 1:"hello">

2.1.2 :002 > "55 hello 44".match(/\d+ (\w+) \d+/)

=> nil

2.1.2 :003 > "55 hello 44".match(/\d+(\w+)\d+/)

=> nil

Line 1 has the spaces, so they were matched.

Line 2 has 2 spaces before the 2nd \d+, so it did not match

Line 3 doesn't have spaces, so it did not match.

(\w is any word character - so 0-9, A-Z, a-z, and \_ are allowed.)

But if you want it to be "more clever" like match tabs, new lines, multiple spaces, etc. you should use the character classes.

**25-Sep-2014**

Run time around a few minutes is ok

“55 hello 44”.match(/\d+ (\w+) \d+/)

\w is word characters. You will save “hello” from this.

“55 hello 44”.match(/\d+(\w+)\d+/)

This won’t match anything. You have to explicitly include a space if you want to match that.

sub method will replace

“55 hello 44”.sub(/l/, “\*”)

This replaces the first occurrence of “l” with “\*”

If you want to replace all of them, use global sub

“55 hello 44”.gsub(/l/, “\*”)

“55 hello 44”.gsub(/\d/, “\*”)

If you want to change the string using this, use the !

a = “55 hello 44”

a.gsub!(/\d/, “\*”)

Gems are the Ruby version of packages / modules / libraries

Ruby Gems consist of three things

* Code (tests, utilities)
* Documentation
* Gemspec – gives you nice meta version about the Gem e.g. author, dependencies, etc.

To install a gem:

gem install (name of gem you want)

For instance:

gem install nokogiri

Now you need to “require” this

require first checks the same directory as your script, then checks the standard library, then checks for a Gem.

Using nokogiri

require ‘nokogiri’

require ‘open-uri’

page = “http://www.cis.upenn.edu/~swapneel/test.html”

doc = Nokogiri::HTML(open(page))

# will give you each row within <table> tags

doc.css(“table”).each do |row|

puts row

end

# will just give you rows that are <td>, note using row.content to just get content

doc.css(“table tr td”).each do |row|

puts row.content

end

puts will print it out

p will show the type more clearly. p uses the object’s inspect method to print it out.

Also, one of these will print out newlines, the other will print out \n.

Nokogiri has a steep learning curve.

Array of 10 elements, want to square everything in this. Note << is append. \*\* is exponent.

numbers = Array.new(10) {rand(1..100)}

p numbers

squares = []

numbers.each do |num|

squares << (num \*\* 2)

end

p squares

Functional programming revolves around three concepts.

* Map – takes a collection e.g. array, goes over the values, does some transformation on each value, and stores this result in an array
  + Can do this instead for above squares. squares = numbers.map {|num| num \*\* 2}
* Filters. The most common you’ll probably use are select and reject.
  + selected = numbers.select {|num| num % 2 == 0 } # choose even numbers
  + “reject” is the opposite. Replacing “select” with “reject” above gives you the odd numbers in “numbers”
* Reduce. It takes an array and applies a transformation. It reduces the array into one value. E.g. if you need to add an array of numbers.
  + total\_sum = numbers.reduce(0) {|sum, num| sum = sum + num}\*\* (see below)
    - “sum” is running across all the iterations. Think of “sum” as the running total.
    - “num” is what’s returned in this iteration (each value, one at a time, as it goes through the array)
    - Ignore the “0” for now. Think this is the initializing value for “sum”. For instance if you want the running product, you should use the number 1: total\_sum = numbers.reduce(1) {|sum, num| sum = sum \* num}. If you leave out this argument completely completely, it will use the first number as the initializer I think (?)
    - Shorthand. p numbers.reduce(:+) OR p numbers.reduce(:\*)
      * The colon here is a “symbol” in Ruby.

\*\* e.g. if doing a sum using “reduce” as per above example, this is how it works as you go through using the example of [1, 2, 3]

sum=0, num=1

sum=1, num=2

sum=3, num=3

6

Computing the factorial in three different ways

1) Iterator

def factorial\_iterative(num)

factorial = 1

2.upto(num) do |value| # looping from 2 to the argument, do the following

factorial = factorial \* value

end

return factorial

end

p factorial\_iterative(5)

2) Recursive

def factorial\_recursive(num)

if num == 1 then

return 1

else

return num \* factorial\_recursive(num-1)

end

end

3) Functional

def factorial\_functional(num)

(1..num).reduce(:\*)

end

**Errors in installing Gems.**

**Guy in class suggested running this before installing a Gem:**

**xcode-select --install**

**(This worked)**

**Swap suggested installing RVM.**

When Swap and I were trying to figure out what was wrong with my version of nokogiri, he used some of these commands to figure out which Ruby I am using etc.

gem --version

gem list

ruby -v

**26-Sep-2014**

Fri recitation

He doesn’t like:

def fun(a, b)

if condition then return true else return false end

end

Just do:

def fun(a, b)

condition

end

THESE TWO ARE EQUIVALENT

def fun(a,b)

f = 2

f = f || 5

end

def fun(a,b)

f = 2

f ||= 5

end

So in the above, we are checking if “f” is nil, and if so we are assigning 5 to it.

x = 3 unless a == b

Asking if any or all of an array elements satisfy a certain condition

def fun(array)

array.any? {|e| e==5}

end

def fun2(array)

array.all? {|e| e ==5}

end

p fun([3, 12, 2, "hello"])

p fun([3, 23, 5])

To do this with the functional stuff we learned yesterday:

array.reduce(false) {|ored, elt| ored or (elt==5)}

array.reduce(true) {|anded, elt| anded and (elt==5)}

Or map and reduce

array.map {|x| x==5 }.reduce(false) {|ored, cond| ored or cond}

array.map {|x| x==5 }.reduce(true) {|anded, cond| anded and cond}

**30-Sep-2014**

Midterm

75-80 minutes

Four broad questions

1. Vocab e.g. HTTP, DNS, etc. He’ll ask 10 terms. Give 2-3 sentence def’n.
2. 10 multiple choice. E.g. what protocol implements packet reliability – UDP, TCP, IP?
3. Debugging – he will provide code that we’ve seen in class and modify it to introduce bugs. Two types of bugs: one that will make it crash because of type errors / invalid stuff, but these will be more syntax errors. The other will be logical errors e.g. doing multiplication instead of addition (won’t crash but will output the wrong stuff). There will be something like 7 bugs and we have to identify 5.
4. Writing some Ruby code. Will give us short description and perhaps some Ruby code e.g. a Ruby method / method signature, and he’ll have us write code. Examples: implement *any* and *all* (recitation) using *.each*

Only focus on the class material.

Lazy evaluation

Doing .lazy.select will result in lazy selection, rather than if you do .select in which case it will try to select all of them. If you just do .select, it will run forever. With lazy.select, you can run up to how many ever you want.

This line – p compute\_all\_primes.first(1000) – will do the first 1000 primes.

Note that doing something like the below would be nice if e.g. you are writing a Twitter client. If you want an array of all tweets, this is an infinite list. If you want to return the most recent 100 tweets at any time, you are using an infinite list which you can’t compute on easily (because your computer will work forever). So you can use the below for such applications as well.

*Code (lazy.rb)*

def is\_prime? (number)

if number < 2 then

return false

else

2.upto(Math.sqrt(number)) do |x|

if (number % x == 0) then

return false

end

end

return true

end

end

def compute\_all\_primes

(1..Float::INFINITY).lazy.select do |x|

is\_prime? x

end

end

#p is\_prime? (3)

#p is\_prime? (30)

p compute\_all\_primes.first(1000)

p compute\_all\_primes #will not do this

We’ve gone over the imperative and functional aspects of Ruby. Now we’re going to cover the object-oriented aspect of Ruby.

In Ruby all instance variables are private by default, unless you explicitly write code to make them public. You have to implement getters and setters.

I think getters and setters make instance variables public (anyone can now get and set the ivars).

Note that in Ruby you don’t usually need the parens for any method call, e.g. see the setter in the code. The name= method actually has an “=” in its method name. The two lines below are equivalent. But the space between “name and =” below is syntactic sugar provided by Ruby, I’m not sure in general that you can put such spaces.

p1.name=(“50 shades”)

p1.name = “50 shades”

The book calls leaving the parens out “poetry mode”.

A lot of the design philosophy around Rails is DRY (don’t repeat yourself).

Ruby has nice shortcuts that auto-generates getters and setters. Put attr\_reader (getter), attr\_writer (setter), attr\_accessor (both) under the declaration of the class.

In general, what does attr\_reader :v do behind the scenes?

It’s creating something like:

def v

@v

end

What does the colon in attr\_reader :v mean?

To motivate it, he asks – what if you want a program in which something can move either N, S, E, or W? How could we save this.

One option – strings. “north”, “south” But then you have a lot of storage and also you might forget if it was capitalized or not.

Another option – enum.

Another option – use ints (which is the old way of doing things e.g. in C). int North = 1, int East = 2, int South = 3, int West = 4. But what if the user passes 2.5?

Ruby solves this sort of problem with symbols. It will not match anything else. For instance, you can use :north, :south, etc. It still has issues around perhaps forgetting capitalization, but it’s a good solution.

If you define a method it will generate a symbol.

def add(a, b)

a + b

end

=> :add

**When I did this, I got => nil. Why does he get => :add?**

Virtual attributes

Say our books are now available in Europe and we want to be able to represent the price also in euros. We don’t necessarily want another attribute for the euros price. E.g.

def price\_in\_eur

@price / EURUSD

end

def price\_in\_eur= (eur)

@price = eur \* EURUSD

end

I don’t know why price\_in\_eur isn’t just a public method but he then says why isn’t Book.isbn just a public method.

Aside: Any variable that starts with a capital letter is treated as a constant e.g. EURUSD = 1.27. Note though that Ruby won’t give you an error if you try to re-set the value (will only give you a warning).

In Ruby you can do all the same as in Java, public, protected, and private.

Protected – only subclasses can access it **I think?**

He said we won’t really need it, not for the hw and probably not in Rails either.

*Code (book.rb)*

**2-Oct-2014**

Three things to think about

* Attributes
* Instance variables
* Methods

Think of instance variables as something private to the class.

Think of methods as public since you need them to be called from the outside.

Attributes are the outside manifestation of the class.

So attributes = methods? Sounds like yes.

In the example book.rb from last time (backed up as book\_from\_last\_time.rb), nothing is private, so everything (e.g. .isbn, .price\_in\_eur) are both attributes and methods.

You can change class constants outside the class e.g. if you defined EURUSD = 1.27 inside the class, then you could run this code outside the class.

p Book::EURUSD #will print out prior value of Book::EURUSD

Book::EURUSD = 4 #you’ll get a warning here

p Book::EURUSD #will print out 4

Class methods: method name has to start with the class name

Besides local variables, instance variables (start with @), and class constants (EURUSD), you also can use class variables e.g.

@@count = 0

And then in initialize (constructor), you could do something like

@@count += 1

These are Ruby rules: if you want an ivar you have to start it with @, and if you want a class variable you have to start it with @@.

If you make Class.something, it makes it a class method. You no longer need an instance to call this.

Class variable: start with @@

Class method: define as Class.method

Note that if I define @@count = 0 in my class (and then increment it with each new constructor call), I can call Book.count (a class method that prints @@count) without even having one instance of Book.

You can also have global variables (start with $) which is the fifth type of variable (besides local variables, instance variables, class constants, and class variables).

*Modules*

Two reasons one would use modules

1. Group code together

How to define a module:

module \_\_\_\_\_\_\_

(stuff)

end

Note:

* No instance. You can’t create an instance of this
* No subclasses.

The reason we use :: is namespaces.

If I have my own Array class aside from the in-built array class. I could make my own module with my own Array class in it, so I could use my\_awesome\_module::Array.

Started work on lazy\_modules.rb which will use code from lazy\_rb.

In Ruby you can’t do multiple inheritance (which is a weird aspect of C++). You can do single inheritance. In Ruby, you can use modules to get multiple inheritance without having to do multiple inheritance, and in Ruby this is called mixins.

**Are modules synonymous with mixins in Ruby?**

*Mixins*

All the classes in the module get mixed into the class you’re using, I assume that’s like including the entirety of the module’s code into your class. **(ask him this)**

Example: first line of class Book, put

include Comparable

That will include the Comparable module as a mixin. In this case, we also need to provide a method that will allow us to do comparisons, for instance

def <=>(other)

isbn <=> other.isbn

end

Note we don’t strictly need @isbn here, we can just use isbn because Ruby will look for the method isbn and just use that method. We could put @isbn if we wanted though.

Another example of this is the Enumerable model which lets you do all the map, reduce, filter, etc. If you define the method .each, then you get map and reduce etc. for free.

each is just something that needs to return succeeding objects (returns the next object I think).

He thinks subclasses can’t see private variables of the superclass.

If you implement Comparable, you get sort, greater than, less than, between (can ask if 7 is between 3 and 10)

If you implement both Comparable and Enumerable, you get max, min, etc.

If you have something that’s not a method you can use Modules to use that thing.

*Dynamic*

* Reflection. e.g. call 5.methods in irb. It shows you the methods you can call on the object 5. **(Will this work on your own classes? It did work for metaprogramming)**
* Metaprogramming – you can add / remove / modify methods dynamically at run-time. Example, let’s extend the Fixnum class. Check time.rb.
  + Do not, in general, override methods in established classes.

In 5.minutes, if I use “self” in the minutes method, it refers to 5.

If you want to get access to the object that calls a method, use self.

Think there is also a method called respond\_to? or some such which seems like another example of reflection (besides .methods).

Something that could potentially be on the exam

Given self and meta-programming, we can add our own methods to arrays and such.

So if you wanted to do .reduce but you didn’t want to do the in-built reduce.

Look at my\_reduce.rb. You could basically just def my\_reduce inside an in-built class like Range.

**But this isn’t re-writing reduce on everything, just in Range, right?**

When I asked him more about mixins, he said they are sort of a “subset” of modules. Modules serve two purposes. One is around namespaces, and the other is the modules thing. **(reading this later I assume I meant to write “the mixins thing” not “the modules thing”)**

**Midterm review (recitation)**

Internet

* Application – HTTP
* Transport – TCP/UDP. Reliability vs. speed. Example of UDP is DNS where speed is important. TCP – makes sure packet loss doesn’t happen, preserves order. UDP has less overhead than TCP, has 8 bytes vs. TCP has 20 bytes or so (assume this is packet size).
* Internet
* Link – hardware

Ports

The more ports, the more applications you can have. There are 65,536 ports possible (0 – 65535). HTTP is port 80 by convention. We should use ports 0-1023 as a general rule.

HTTP = hypertext transfer protocol.

URL – uniform resource locator

URN – uniform resource name

URI – uniform resource identifier

URL and URN are subsets of URI

Client – Server

Client sends HTTP request to the server using any of the following actions / verbs.

GET, POST – used all the time

DELETE, UPDATE – not used much anymore (dangerous)

Server sends back 200 if everything is ok. 404 (Not Found), 500 (Internal Server Error)

GET … HTTP 1.1

HOST

stuff

Get back:

HTTP 1.1 200 OK

ALLOW

stuff

<html>

GET vs POST – GET has the query parameters in the URL, POST does not (more secure).

Use POST when you’re trying to update something.

POST

* use to update
* hides information / more secure

DNS – domain name system. Mapping between IPs and domains.

edu – the edu server then sends you to columbia or upenn or whatever you want.

If you then go to columbia, why do you want another branch e.g. cis?

I think the reason you want another branch (e.g. cis) is because you have a lot of data on just cis, then you want to have a separate machine for that. The downside is it takes longer to do DNS. **Right?**

Host file – 127.0.0.1, my own computer

**14-Oct-2014 – make up done on 20-Oct-2014**

A lot of old software was not user-friendly.

Now we think – we are using software as a user-facing things. Also we are writing code within a team. This is where software architecture comes in.

First software architecture idea – monolithic. Everything is one big piece. It’s hard to separate the components. Good because it’s easy and you can get it to work quickly, bad is it’s hard to port.

* Office is an example. It’s very hard to port because it was based on the OS.
* Games ecosystem. Something that works on Xbox doesn’t work on Playstation, something that doesn’t work on Windows doesn’t work on Mac.

Guy holding a stuffed animal – Linus Torvalds. Let’s do a micro-kernel, a kernel that’s really small that does the important things. Around it are all the libraries and plug-ins that give you functionality.

Unix – Bell Labs developed it, and because of lawsuits they had to give it away initially, but then they split off a child company and licensed it to the child company who can sell it. Linux was super expensive (in the 1980s), only rich entities could buy it.

Then free software (Richard Stallman etc) came around who created a free Unix version. They had everything but the kernel.

(Device drivers / keyboard / mice / printers / Wi-Fi are not part of the kernel. The kernel does basic things like process mgmt., memory mgmt.)

The kernel took a long time – Torvalds wrote it in 6 months.

Torvalds said old version control was horrible. He wouldn’t use CVS or SVI. He would use email and TAR files. One day he takes a day and implements Git and it’s the best version control system that exists.

New stuff – cloud-based environment, this is service-oriented architecture. Bezos sends email to Amazon in 2000 or 2002, this is what we will do henceforth. When we have large pieces of software, let’s not have weird ways of accessing it, let’s just have a simple public way to access each thing over the network. E.g. if I want to use your service, I don’t want to have to use a jar file to access your Java code, I want a network-based API that I can use to access your endpoints of your program and get answers I need.

Last five years, SaaS became common. Everything ran as a service. Instead of Office, you have Google Docs – everything is in the cloud, everything exists.

* Don’t need to install anything.
* Can release updates as often as you want.
* Much easier to test what users like (A/B testing).
* Massive data sets. One project he mentioned used genome data for animal and plant species (run an algorithm with a sequence to figure out what species it is). Every couple months it was updated with 20-30 GB. There is no way to download that over the Internet quickly. Now I have one server that I can update rather than updating everyone’s machines.

Disadvantages:

* Total spyware, gaping back door – server operator has power over the user.
  + What if Google asks us tomorrow that we need to pay $1000 to keep accessing our email?

He says SaaS = service-oriented architecture, everything has a service interface that is open. In the extreme, everything is on the cloud. But ultimately, I think SaaS implies more about the architecture / development side rather than the user side.

Web Services (WS-\*). You have massive XML files to configure your application to do different things. It got too complicated (joke WS-DeathStar).

Simple Object Access Protocol (SOAP) – simple relative to WS but still extremely complex.

2000, Roy Fielding (largely responsible for the Web initially in the 1990s with Tim Berners-Lee), part of his doctoral dissertation was REST. All this stuff with SOAP and WS-\* is way too complicated, let’s use HTTP instead (GET, POST, etc., that we know well). Let’s use this and make Web applications.

You have resources – Amazon has users, shopping cart, etc. (all resources). In the URI, you have the resource and what I do with it, rather than having some implicit state that exists. This is stateless, now you can do caching because you don’t need to worry about state. The client worries about state because it should.

SOAP / WS not HTTP-based?It can be but it could also not be. Something along the lines of Java servlets, can send stuff back and forth (code?). RPC – remote procedure call. REST is only HTTP, only use HTTP.

**Edwin Lou’s notes:**

* it all starts with monolithic softwares, tightly bound with hardware and OS
* component-based software engineering: micro-kernel of Linux (early 90s)
* early internet days: client-server model
* service oriented architecture: all services must be utilized only through a interface designed for external uses
* Software as a service:
  + access anywhere
  + mandated push updates
  + unified execution environment
    - no setup/installation
    - less CPU constraints
    - no local storage
  + content ownership has become a controversial issue
* But web API has been complicated; to counter that, REST (representational state transfer)
  + collection of resources on which specific operations can be performed
  + URI names resources

**16-Oct-2014 (missed Tues 14-Oct-2014)**

Sounds like REST is the standard now.

Example of DRY (don’t repeat yourself): validation – just do it in the Model layer, don’t do it also in the View layer

Database independence – ?

SQLite – good for development, not good for production

PostgreSQL – sounds like he likes this

Inheritance in Ruby – Post inherits from ActiveRecord::Base, for instance:

class Post < ActiveRecord::Base

ActiveRecord can generate SQL code for you. It does a good job with relations, with joining, with searching and filtering, etc.

ERB – embedded Ruby (in HTML). You have embedded Ruby code in HTML.

Rails – broadly the two options are Rails 3 and Rails 4. The text mainly uses Rails 3, so he’ll use Rails 3. So if we want to follow with the book it might be better to use Rails 3 but we can use either version.

Install Rails 4: gem install rails

Install Rails 3: gem install rails -v 3.2.19

You might need to do:

gem install bundler

Command line:

rails new spotify

When I tried to install rails, rdoc conflicts with executable (have screenshot)

mate .

rails server

Somehow he got it so that localhost:3000 could be loaded in a browser.

Click on Gemfile – think this is from the “rails new spotify” command which creates a spotify directory in which there is a Gemfile.

Think the below refers to line 3 of the Gemfile (which says gem ‘rails’, ‘3.2.19’ in my file)

*First option:*

gem ‘rails’, ‘3.2.19’ (see line 3)

*Second option:*

gem ‘sass-rails’, ‘~>3.2.3’ (see line 14)

The above is semantic versioning. In A.B.C, the C refers to patches (minor changes). So upgrading from 3.2.3. to 3.2.5 shouldn’t really change anything. But A is the major version and B is the minor version. Changing from 3.2 to 3.3 (doesn’t exist) will cause some big changes though.

*Third option:*

gem ‘sass-rails’, ‘>=3.2.3’

*He said the semantic version is probably what he would recommend.*

Gemfile.lock tells you – for what’s running right now, what’s installed. And tells you the dependencies.

This lock file is nice because if you’re working with teammates, then if you add this lock file to version control, it will ensure it will use the lock file first and not resolve dependencies using the Gemfile.

If you get Gemfiles from somewhere else (?) you can use “bundle install” and it will resolve dependencies and install whatever you are missing. It will use Gemfile.lock first.

Need to do bundle install if you change your Gemfile (?).

I think you need to run bundle install from the root of your project (e.g. in spotify/).

Do “bundle install” every time you change your Gemfile. If you’re not changing any of your Gems you don’t have to do it.

Btw if you go to the root of the project and do “rails server” then localhost:3000 should work.

ActiveRecord

Class: album

Table: albums

Table has to have a primary key (auto-increment integer)

Rails has generators which can generate code, e.g.

rails generate --help

rails generate scaffold --help

Trying this generator thing:

rails generate scaffold album name:string price:float release:date

The above will create an MVC i.e. the whole thing (which is why it’s called a scaffold)

Now go to the app folder (which has everything related to your app) – in app/models/ there should be album.rb. In app/controllers/ there is albums\_controller.rb

In apps/views/ there is a directory called albums/ which has things like index.html.erb

There is also a reverse generate thing e.g. rails scaffold destroy, or some such thing to undo what you may have accidentally generated.

Go to config folder and go to database.yml. Since it’s doing it by default it’s assuming you’re using sqlite3. But you change this in your Gemfile.

So far it’s created a bunch of things but hasn’t made a database.

Btw if you have a database, it can do it backward and create the model from that.

Or you could do it like we have, creating model first and then creating db.

rake is Ruby’s version of make, so it lets you automate scripts (?)

rake db:migrate

This takes whatever state the db is in and update it to the latest version (e.g. you might have done added tables, added columns to tables, done joins). So it makes it up to date. I think this is database migration.

Also, note “rails s” = “rails server”

“rails g” = “rails generate”

Now run the server and go to: http://localhost:3000/albums. You can even add an album and do a lot of stuff by now!!

Also, note all the URLs are RESTful. If you mouse over “Show” for the album you made – you see localhost:3000/albums/1. Similar for Edit, New Album, etc.

You can see that it’s localhost:3000 by the text that appears right under “rails s”

You can see not only all the REST stuff but the db stuff (e.g. committing transactions and so forth) in the output in the console that occurs as you navigate the webpage.

Smthg (?) lets you keep versions of your database in sync.

Think he said we’ll go over this later.

Essentially saying, though, that Rails handles this (I think).

Check db/migrate/20141016165159\_create\_albums.rb

rake routes

This will show you all the routes that are allowed or that your webapp supports.

First column is shorthand that you can use when, e.g., writing your views. The second column shows the HTTP verb. The third column shows what the URL looks like in the browser.

Install Bootstrap (“the new hotness”) into your application is as follows (note Bootstrap is a Gem):

In Gemfile:

Line 10: gem ‘twitter-bootstrap-rails’

Line 11: gem ‘less-rails’

Uncommented: gem ‘therubyracer’, :platforms => :ruby

Now have to run “bundle install” because you changed your Gems

Now do rails g --help, now there are a lot more generators available e.g. Boostrap-related.

If you want to install Bootstrap into your app (using the less version)

rails g bootstrap:install less

Now I want to use the Bootstrap layout (put “application” so it persists throughout)

rails g bootstrap:layout application fixed

Conflict with a file – let it overwrite because it’s all auto-generated.

Now make albums Bootstrap themed

rails g bootstrap:themed albums -f

(-f forces – update the pages)

(have to have the name in plural form because that’s what Bootstrap wants?)

Now erb file has a lot of stuff, and also check the browser.

**20-Oct-2014 office hours**

RVM – which to install, and how this works

If I have two Rails apps, one in 1.9 and one in 2.0, RVM can help you manage this.

You can maintain Gemsets. I can have virtual sets of Gems if I’m doing one thing in Rails 3 and one thing in Rails 4.

Gemfile – don’t understand. Think if you use a Gem, this file tells the app which version of the Gem to use.

Project – my idea

* Raw idea
  + **Koala – Ruby gem that talks to Facebook**
* Privacy / feedback in general

Mobile app to link into Rails back-end – there could be a Gem thing that does it for me. He sent me this link:

[http://www.takeofflabs.com/posts/8-Rails-and-iOS-a-sample-starting-setup](http://www.takeofflabs.com/posts/8-Rails-and-iOS-a-sample-starting-setup" \t "_blank)

For RVM he had me run:

\curl –sSL https://get.rvm.io | bash (from the RVM website – in order to install RVM)

rvm list (then it said to do ‘rvm reload’)

rvm reload

rvm list (only printed ‘rvm rubies’, then it said ‘No rvm rubies installed yet.’)

rvm list known (printed out a bunch of stuff)

rvm install 2.1

**21-Oct-2014**

Why is software hard? Think about it in terms of cost, quality, and schedule.

* Can’t foresee problems
* Dependencies
* Integration
* Dealing with people
* Hard to estimate time
* Developers over-/under-estimate
* Requirements are not clearly defined
* Requirements change
* Requirements are not realistic

Waterfall / Plan & Document

Steps you take (in order):

1. Requirements
2. Design
3. Implementation
4. Test
5. Deploy / Maintain

The problem with the waterfall model is it’s inflexible. If you need to change something, it’s very hard to go back and change things.

In response to this, “Agile” was created around 2001. Story was they were at a ski resort in Utah, etc. etc. They created the Agile Manifesto. The most important component of this is responding to change.

Kent Beck – wrote most of the unit testing stuff in Java

Extreme Programming – extreme end of all these ideas. Extreme Programming improves a software project in five essential ways; communication, simplicity, feedback, respect, and courage.

User Story

* A lightweight description of how the app will be used.

For example (fill in the blanks):

As a [stakeholder], so that [goal] I want to do [task].

Spotify example: As a user, so that I can keep track of my albums, I want to be able to add albums to the listing of everything.

Spotify example: As a user, so that I can see my albums in alphabetical order, I want to be able to sort albums.

**Validation** – am I building the right product? What you’re interested in, as far as the **behavior** goes.

**Verification** – am I building the product right? What you’re interested in, in terms of **implementation**.

User stories help with validation.

Later, we worry about verification.

BDD – Behavior Driven Development

TDD – Test Driven Development

Not sure what the below is in reference to, but he said most people think of this as BDD, and others think of it as Agile Programming and XP. He said it’s most appropriate to think of this as “steps in Agile development”.

* First do User Stories.
* Then an important step is estimation – some user stories are quick to implement, others are hard. To do this, use point values. One example:
  + Give 1, 2, or 3 to each user story based on how hard it is. 1 is straightforward, 2 is medium, 3 is complex.
  + 1-5 is popular
  + Fibonacci is popular. You can rate a user story at: 1, 2, 3, 5, 8 (some people have 13 also but usually not).
  + Interesting point – if you have 1-3 and then you see something and say “that’s a 4”, then this is bad. You need to split that user story up.
* Planning iteration. Prioritize. It could be priority in terms of the estimation above but it could also be in terms of what the customer wants / MVP. If customers really want a 3, probably work on that rather than the 1’s and 2’s that customers don’t think are that important.
  + You want 1 week to 3 week iterations. We’ll do 2 week iterations.
* Velocity – speed at which you’re going. “Our goal was to do X points in this iteration time of Y weeks”. When you then exceed or fail to meet your goal, this tells you that in this iteration you managed to do Z points. Over time you get a sense of estimating how much work you can do per iteration. E.g. if we’ve been doing 15-20 points per iteration then it helps you plan for the future (of course this is not externally useful, it’s just useful internally to your own team).

He said a good way to think about the 1-3 system is:

1 – 1 day (of uninterrupted work)

2 – 2 days

3 – 3 days

Pivotal Tracker – online tool to do all this stuff e.g. user stories, point values, you can do iterations and velocity, etc.

The book says user stories need to be SMART.

Specific

Measurable

Achievable

Relevant

Timeboxed

Spotify example

Specific

* As a user, I can sort the albums. This is not good because it’s not specific. So change:
* As a user, I can sort albums by price descending.

Measurable

* As a user, the page should load fast. Not measurable. Rather:
* As a user, 95% of the pages should load in < 3 seconds.

Achievable – literally is this feasible. More narrowly, can it be achieved within one iteration? A lot of small point value tasks is better than a few big point values.

Relevant – why do I care about it e.g. why should I care that 95% of the pages load in < 3 seconds? Because people have no patience, so it’s important.

Rails makes it easy to do all these things. There is a thing called Cucumber (talking more about this next time). It can do lots of stuff e.g. run all your user stories and test them (I think).

Tests are good but they don’t make sense or matter unless you can automate them.

Lo-Fi UIs.

Low-fidelity UI

Don’t invest a lot of time in HTML and CSS and Rails etc. Research has shown that you won’t get feedback from customers – they’ll feel bad and think you did a lot of work and won’t tell you the changes they would like.

Just use pencil and paper!!! Then customers will give you feedback.

I think he said Lo-Fi UIs are in the MOOCs (?). Then he said something about how this is used in the world of film and art, etc.

**23-Oct-2014**

**Basic Git stuff we should know (will spend the first half of recitation on this)**

Git

Remote

Branches, merging in branches

Logging

Add / push / clone

Back to Spotify example

Gemfile.lock – which Gems are being used, don’t need to worry about so much

config.ru – tells the server which is the application to run and what are the parts to load and so on, we don’t need to worry about so much

app directory – will need to deal with this a lot

* assets: JavaScript, CSS, images, sounds (media)
  + Note javascripts has bootstrap stuff, so you could modify bootstrap here
* helpers: DRY. Extract common functionality across models and views and controllers and put it in here
* controllers: application\_controller has something simple, protect\_from\_forgery, some stuff around stopping cross-site scripting. albums\_controller.rb, what we can do with the scaffold, etc.
* views
  + albums: has \_form.html.erb which is a partial form which allows you to embed functionality in multiple forms.
  + layouts: regardless of your specific view, your application should an overall certain look and feel, that’s in here
* config directory
  + database.yml file. Think in Rails (?) you have three versions: dev, testing, production. In database.yml you have development.sqlite3, test.sqlite3, and production.sqlite3.
  + routes – what maps to what (?). When we generated scaffold it automatically added this “resources” thing. (?) When you say add an album go here, etc. that’s all in resources.
* In db directory
  + development.sqlite3, nothing in there yet
  + schema.rb – will generate this from your database schema. This is the state of the world when it last ran (not since any other migration). When you are doing things in testing, and you move to production, you typically don’t want to do migrations because there might be thousands of migrations. You can do database schema.load (?) and it will do something for you.
  + seeds.rb. db:seed on command line will automatically populate certain databases so you don’t hae to do it manually
  + migrate directory has all your db migrations
* log directory
  + development.log shows all the logs when you did “rails server” over time. Shows you all the requests being made, etc.
* public directory. Two purposes:
  + 1) Serve static html files like 404.html etc.
  + 2) In production mode, when you do assets, you need to pre-compile assets because in development mode all the CSS etc. gets evaluated, but it’s easy to do in development b/c you don’t care about speed, in production you do so it will change all your CSS files so it’s smaller (is this minification?)

IRB equivalent (REPL) for Rails is rails console

His “rails console” didn’t work, so he then did:

rvm use 2.1@rails3

rails console

My “rails console” doesn’t work

After he did rails console, he does:

Album.all

This gave output along the lines of SELECT “albums”.\* FROM “albums” and then some associated data

Album.find(1)

Album.find(1).price

So you can inspect all your objects and modify them, etc.

He added this in Gemfile, line 13

gem ‘annotate’

He really likes this gem.

Annotate takes the db you have and annotate your models with the appropriate table information.

Then do “bundle install” since you changed your Gemfile (this doesn’t work for me now!).

Then I think he did annotate at the command line?

He goes to album.rb (think this is in app/models/) and there is a lot of schema information (?).

Command line:

annotate --routes

It will annotate your routes file, go to routes.rb, you’ll see there is a route map at the bottom (I think mine is at the top?). If you do POST, then it will go this thing, if you do this specific GET request, it will go to that thing, etc.

Pivotal Tracker

When you’re done with something in Current, you would hit Finished, then you would hit Deliver if it’s ready. Then they would hit accept (?)

Now you have a done column, you can look at tasks, who did them, and what they were.

From beginning: add story (upper right) and you get into Icebox.

Create New Project

Add story

Test story 1 on top line, then story type = feature, requester will probably be the person who makes it, owner will also probably be yourself, write a short description. Labels are like hash tags.

Going back up, point value is what is important.

Hit “Save”.

When you want to do the task, drag it from Icebox to Current.

If you try to add more points to “Current” than your “velocity” (upper right), it won’t let you.

You can attach stuff e.g. you might have lo-fi mockups. Just draw it, scan it in, and attach it.

Testing

Just be pragmatic. Prioritize what you test. Ask what the most important features are. You want to test the core functionality. You might want to test things that are error-prone. If you’re integrating with an API that does weird stuff, maybe test that.

In some cases it makes sense to do unit testing but he’s not going to force us to do this for all parts of our project.

Cucumber is a tool that lets you do BDD. Taking a user story, it will automatically open the page in a virtual browser, click the links as you tell it, see if the output matches, etc. Cucumber and capybara are similar.

In Gemfile, line 15

group :test, :development do

gem 'cucumber-rails', :require => false

gem 'cucumber-rails-training-wheels'

gem 'database\_cleaner'

gem 'capybara'

gem 'launchy'

end

Whatever goes in between “group :test, :development do” and “end” will be tests

* gem 'cucumber-rails', :require => false. I didn’t follow what this does. Unless I say require ‘cucumber-rails’, don’t use this (?).
* gem 'cucumber-rails-training-wheels' – handholding for Cucumber rails
* gem 'database\_cleaner' – cleans your db btwn multiple runs
* gem 'capybara' is a Gem that mimics a user
* gem 'launchy' – lets me launch applications in an OS-agnostic way. For example, with tests I can launch browsers, but launching a specific browser on different OS’s is different, doing email clients is different too, so launchy abstracts this from you.

Now do bundle install

rails g --help – you’ll see it has added generators e.g. for Cucumber and CucumberRailsTrainingWheels. Let’s install these:

rails g cucumber:install capybara

This will create lots of files and folders

Cucumber is installed as a gem before we ran this (rails g cucumber:install capybara), but just like we went in and changed the views and make them pretty with Bootstrap, it will go in and change Cucumber to put in the capybara stuff. So it sort of just makes Cucumber more specific. There are other options besides capybara, by the way. capybara is something that allows mimicking of user behavior.

Now let’s install Training Wheels. Note that the file has a dash but the generator has an underscore.

rails g cucumber\_rails\_training\_wheels:install

Let’s look at what changed:

There is now a features directory

* step\_definitions, web\_steps.rb. TL;DR: YOU SHOULD DELETE THIS FILE. He says to delete this file (and he disagrees with the book). He said this helps you get started with user stories, but he says this is really bad. The book also follows this, so ignore this in the book.
* Another file also has YOU SHOULD DELETE THIS FILE and we should also follow this.

Say we want to add a feature. Anything you put in the top folder in features, will be a feature. Let’s create a new file in features, called “AddAlbum.feature”. I’ve put a fake file in cit/spotify\_temp to do this.

Then he goes to command line:

rails server

(rvm use 2.1@rails3 after “Could not find addressable…)

Now when he’s on localhost:3000/albums/new, and he adds a new album, he sees the new album. He wants to test this. This is the AddAlbum.feature file:

Feature: Album submission

As a user

so that I can track my albums

I want to add a new album to the list

Scenario: Add an album

Given I'm on the album creation page

When I add a new album

Then I should see the newly created album

Then show me the page

Then run “cucumber” from the command line.

Cucumber is three things:

* Features / Scenarios – these should be as close to your user stories as possible. And, typically these should be done wit your customer.
* capybara (Web Steps) – the glue that hooks up the bullet above (features / scenarios) to the bullet below (Rails app)
* Rails app. This is a website that works.

web\_steps.rb – it has lots of stuff though it also says TL;DR DELETE THIS FILE

Don’t use this though because the English is really bad (though the code is good)

Going back to what was printed out in the console after “cucumber” (copy-paste the steps that it said were undefined), he copied the stuff e.g. “Given(/^I’m …

Then(/^I…

end”

You can re-use

He puts this in album\_steps.rb

Now run cucumber again.

Yellow is pending / not implemented. Blue is stuff it skipped. Green is what actually passes (we’ll see this later). If it fails it will be red.

Let’s now implement that functionality for the yellow.

Let’s start with the first one (the below in album\_steps.rb which has the below as well as two others)

Given(/^I’m on the album creation page$/) do

pending #

end

How do we implement this:

Given(/^I’m on the album creation page$/) do

visit new\_album\_path

end

If you remember with rake routes, there was an option for new\_album GET etc. So if you use new\_album\_path as per above, it will know what to do.

Now run cucumber again.

Now we will have a thing that passes / fails. It couldn’t find the table passes.

But there’s three things, we have a dev, test, prod. We haven’t set up a test db yet.

Let’s set up the test db.

rake db:test:prepare

This will figure out what the schema should be, create the appropriate tables, etc.

Run cucumber again.

It passes “Given I’m on the album creation page” and it is still pending for the other stuff e.g. “When I add a new album”

Now let’s implement the next one

When(/^I add a new album$/) do

fill\_in ‘Name’, :with => “Lullaby”

fill\_in ‘Price’, :with => “11.66”

click\_button ‘Create Album’

end

You can look at Web steps for all the options here e.g. it can do normal forms, radio buttons, clicking

Multiple forms for same pages – you can specify all your normal CSS Xpath stuff

Run cucumber again, now two things have passed.

Now we have to verify that the album has actually been created. Now we have multiple options, and this is where you need to know CSS stuff.

How do I know the album has been created? Twitter Bootstrap is masking this, but if you look at the page source and search for success you will find “Album was successfully created”. It’s hidden by the bar but it’s there. So we can use this in our Web step to see if it’s successfully created since logically it wouldn’t show up if the album was not successfully created.

Then(/^I should see the newly created album$/) do

assert page.has\_content?(“Album was successfully created”)

end

Has all the stuff for hw2 with Xpath, CSS selectors and queries, so you can do all this stuff.

Run cucumber, all of it should now pass.

We don’t need to implement “Then show me the page” because it is in Web steps already (?).

**Met him in his office later to discuss rvm issues.**

Something was wrong with ruby and basically he thought it was that the libraries were messed up or weren’t linking up correctly, because of the Mac OS update.

He did:

rvm get head

This updates rvm

rvm list known

rvm --help

This showed him the command to just reinstall rvm

rvm reinstall 2.1.3

rvm list

ruby -v

**28-Oct-2014**

BDD – care about user stories not code and database stuff. We will test this stuff in a week or two (e.g. unit tests).

3 levels:

* Features & Scenarios
* Web Steps
* Rails

Feature files – for each feature, you can have as many scenarios as you want.

Gherkin – he’s using this. Plug-in for Text Mate. It’s a Cucumber plug-in.

In features/album\_steps, new\_album\_path works because we saw in rake routes that this is mapped to something.

Adding a scenario

Right now we’ll write a scenario, see what happens, and then change the Rails app. Could have done it the reverse way also.

In AddAlbum.feature, add this:

Scenario: Add an album without price

Given I'm on the album creation page

When I add a new album with no price

Then I should see errors on the page

Then run cucumber

It says undefined. Copy-paste the templates into features/step\_definitions/album\_steps.rb

Copy-paste the earlier steps so that you enter form in without price, and run cucumber again. Now we have two greens.

Let’s go back to the models. Go to app/models/album.rb. All the responsibility for validations is in the models (?). In album.rb, put this on line 16 as part of class Album

validates :price, :presence => true

Now if you try to save an album without a price, it will throw an error.

If you try to add an album, it highlights the price in red. Also, if you check the source you will find HTML has field\_with\_errors i.e. the below.

<div class="control-group">

<div class="field\_with\_errors"><label class="control-label" for="album\_price">Price</label></div>

<div class="controls">

<div class="field\_with\_errors"><input class="text\_field" id="album\_price" name="album[price]" size="30" type="text" /></div>

</div>

To check for a div, do this in album\_steps.rb for “I should see errors on the page”.

assert page.has\_css?('div.field\_with\_errors')

In capybara, you can compare CSS, compare the content, compare the URLs.

Now if you run cucumber, it all works.

Note that if you get rid of the validates :price, :presence => true in app/models/album.rb, then this test (adding an album without price results in errors) will fail. That’s because the model’s handling of validating that price is present means that the model will generate html that allows you to check that something has gone wrong. When you try to check for this html without this validates bit, you will obviously not find it.

Btw Rails 4 has “strong parameters”. Essentially, Rails 4 moved attr\_accessible (currently in album.rb in models for me) from the model to the controller. This is one of the bigger changes in Rails 4.

BDD can’t give you access to db / variables – if you want to do that, you need to use unit testing. BDD only does user interaction stuff e.g. going to a webpage, filling out a form, etc.

Github for capybara – github.com/jnicklas/capybara

Pointed out this on “Navigating” – expect(current\_path).to eq(post\_comments\_path(post)). Is the path that I’m on, is it what I expect it to be?

Also, “Clicking links and buttons” is really nice e.g.

I don’t need to do create album, I can do create, or just cr.

“Interacting with forms” – also good stuff

“Querying” also nice – looking for certain xpath, content, CSS

**(Didn’t catch difference between cucumber and capybara)**

RSpec is unit testing, we’ll see unit testing later. We can do everything with RSpec in capybara. Some stuff you can do there if you prefer it over assertions.

Also if you don’t know what you want to click in terms of buttons, etc. there are “Finders” also.

Cucumber

You can run a report with -f

Use html to make it in html format

cucumber -f html > ~/Desktop/test.html

If something fails, you’ll see in the report, the line number of the failing scenario.

You can also do:

cucumber -f progress

You won’t see all the successful stuff, you’ll just see a dot for each passing scenario and see the full details for the failed scenarios.

Internationalization

We should read declarative / imperative scenarios.

Remember the user doesn’t understand html, CSS, Ruby, etc. so you want the scenario (user stories) to be as close to English as possible. But if your clients don’t speak English, then cucumber has in-built stuff for various languages e.g.

cucumber --i18n help

You’ll see all the languages.

Do:

cucumber --i18n en

cucumber --i18n es

cucumber --i18n en-au

cucumber --i18n en-Scouse

cucumber --i18n en-lol

cucumber --help – good stuff

Let’s add playlists to Spotify.

rails g scaffold playlist title:string description:text

Now have to migrate

rake db:migrate

Now do rails s, and go to localhost:3000/playlists

Doesn’t look great but can do Bootstrap to make it pretty.

Let’s do more BDD tests on playlists.

Make a new feature file. Let’s keep one feature to a file.

AddPlaylist.feature

We can actually use methods that we can use across feature files.

Rename features/step\_definitions/album\_steps.rb to steps.rb (since it’s not just albums)

The only thing that changes on the top is new\_album\_path should be new\_playlist\_path

In steps.rb, because it’s a regex you can do this:

Given(/^I'm on the album creation page$/) do

CAN BECOME

Given(/^I'm on the (.\*) creation page$/) do |thing|

thing will be set to album or playlist depending on the scenario that’s running

Then could do:

visit new\_album\_path if thing == ‘album’

visit new\_playlist\_path if thing == ‘playlist’

(Instead of visit new\_album\_path)

Instead let’s do eval.

visit eval(“new\_#{thing}\_path”)

This will convert it to the appropriate string and then to the object, then it will visit it.

Change

Then(/^I should see the newly created album$/) do

assert page.has\_content?("Album was successfully created")

end

TO

Then(/^I should see the newly created (.\*)$/) do |thing|

assert page.has\_content?("#{thing.capitalize} was successfully created")

end

Note: Don’t use eval in a large-scale app, it’s a bit slow.

Given, When, Then – cucumber

fill\_in, click\_button, etc. – capybara

cucumber doesn’t care that much about capybara and capybara doesn’t care at all about cucumber

cucumber doesn’t pass for new tests – because the test db has not been updated!

rake db:test:prepare

Last missing bit:

When(/^I add a new playlist$/) do

fill\_in 'Title', :with => "Awesome Playlist"

click\_button 'Create Playlist'

end

And if you want to see the actual page it’s making, add “Then show me the page” at the end of AddPlaylist.feature

**30-Oct-2014**

devise is a Ruby gem to deal with user authentication and authorization. It’s a nice way to deal with situations where certain users can do certain things.

He thinks it’s better to divide work vertically i.e. by feature, rather than horizontally (one person does model, one does controller, etc.). This is because when you alter one feature’s view, you might have to, for example, alter that feature’s controller to make it work and so forth.

For user stories, keep in mind that you want them to be SMART.

He said – we should read the declarative vs. imperative scenarios. The idea is that the user stories and scenarios should be as close to each other as possible. It should not contain code, regex, etc.

Use a good gitignore file, it’s useful. Didn’t catch why.

temp file

github.com – go to New Repository. You’ll see Add .gitignore file, and it has one for anything you’d want. There is one for Rails.

It’s a list of things you don’t want to commit to Git. You don’t want to commit your databases, you don’t want to commit your secrets, you don’t want to commit Gemfile.lock (which is why it’s commented out in .gitignore).

Everything is local on Git. If you do a commit, it will use the .gitignore that’s in your folder (local directory).

If you want to put in a .gitignore into an existing repository, then you have to do the below to remove those things from Git (it won’t automatically remove these files from Git because it’s too dangerous):

git -rm

Database

Collection of tables – system to organize, store, and retrieve large amounts of data.

DB Model

Theoretical foundation on how data is organized, stored, etc.

In the beginning when people were thinking about data they thought about it hierarchically and in terms of relationships.

Relational DB

Describes relationships among data.

* Hierarchical
* Network
* OO
* NoSQL

Properties

* Unordered
* Uniqueness
* Keys – subset of entities
  + Candidate key – any set of attributes that can uniquely identify a row. E.g. in country table example below:
    - Name
    - {Latitude, Longitude}
  + Primary key – is the key you’ve chosen to be the key for your table

Tables

Columns are known as entities (no?)

Entity

* Name
* Data type

Example:

Table Entities

* Name – string
* N\_S – string
* Latitude – number
* Longitude – number
* Interest – string

Default is that the primary key (Pk) for all your tables is called “id” (think in Rails). This will be auto-incremented.

Schema – table definitions. Gives you table names, primary keys, entities

**So it sounds like he calls fields “entities”.**

Active Recordin Rails is an ORM i.e. object-relational mapping. An ORM maps objects (i.e. instances of a class e.g. Rails objects) to the tables / db and vice versa.

Objects

* Attributes 🡨🡪 entity/columns in a table
* Class 🡨🡪 Table (if we have an album class we want a table called Albums)
* Methods 🡨🡪 no mapping. Don’t need to do anything here. Methods are your way of doing things with your objects, but there’s nothing analogous in a relational db.
* Instances 🡨🡪 rows

Active Record does this sort of mapping automatically.

Associations (guides.rubyonrails.org/association\_basics.html)

2.1 The Types of Associations

Rails has support for 6 types of associations e.g. belongs\_to, has\_one, 1many, has\_many :through, one :through, has\_and\_belongs\_to\_many

To do migrations, sometimes you can do this automatically using scaffolds or using generate model or some such thing, but other times you might do it yourself.

belongs\_to sounds like it’s a foreign key reference

A model corresponds to one table.

has\_many

By saying, say, a customer can have multiple orders, you get 16 methods for free e.g. orders.delete, orders.destroy, etc.

By adding any given association, Rails will give you many methods which could be useful.

Active Validation validations – e.g. the name of the firm can’t be blank, password has to be 12 characters or more, etc. Another validation could be, for a given order, there has to be a customer id corresponding to that order.

Convention over configuration.

e.g. customer\_id

If you have a foreign key reference, the convention is the singuralized name of the table, underscore id. So for instance in orders, you have a field that references the id in customers, so this field should be called customer\_id.

Now let’s do this stuff with scaffolds.

Let’s try to create a relationship between albums and songs. An album can have many songs, and a song belongs to an album.

rails g scaffold song title:string length:time album:references

Looking at app/models/song.rb, it has belongs\_to :album

In db/(latest timestamped file), you have some information. (?)

rake db:midgrate

Now if we do rails s, we see it’s added the songs.

**Didn’t get why we have to do this.**

Can’t add a song, there is a bug. In app/views/songs/\_form.html.erb, change “f.text\_field :album” to “f.text\_field :album\_id”.

(The line is <%= f.text\_field :album\_id %>)

Then: annotate

Now song has been annotated.

In app/models/song.rb

Change “attr\_accessible :length, :title” to “attr\_accessible :length, :title :album\_id”

Now creating a song works

In app/models/album.rb:

Before end, has\_many :songs

Now if we delete that line where that was that bug in the form html file, and then put this in its place

<%= collection\_select :song, :album\_id, Album.all, :id, :name %>

:name is what I want to display when I put the drop-down menu.

**31-Oct-2014 recitation**

If you have a user, you probably gave them

id : int

name : string

password : string

email : string

devise adds to your user table, will add

email : string (is this a devise email)

password : string

passwordconfirm : string

If you’ve already added email and try to do this, you’ll get an error. You’ll have to run a new migration to remove the old stuff and keep the new stuff.

If you ever use a gem and run db:migrate and get a conflict in schema then the answer is to make a new migration that removes the things that conflict and run rake migrate.

If I ran:

rails generate scaffold user string:name string:email string:password

And then did “rake db:migrate”, then I would get an error (a conflict between email and password, which existed in the old version and in the devise-generated version). So you do:

rails generate Migration RemoveEmailFromUsers

rails generate Migration RemovePasswordFromUsers

rake db:migrate

Now it should be able to add email and password from devise

Also, if you want to add a field e.g. customerId

rails g Migration AddCustomerIdToUsers

**3-Nov-2014 office hours**

Route maps to a controller (not to a view). The controller executes an action.

**4-Nov-2014**

rails c

Album.all

Album.attribute\_names – get attribute names, not just what I created but also what Rails created for me

Album.create(:name => "My Album", :price => 10) – provide a hash and it will add it to the db.

a = Album.new(:name => "Another Album") – note that the id of this is nil right now. This is because we haven’t added this to the database. This object is just floating around in memory right now.

a.save – trying to save it to the db now. It fails because you need a price.

a

To see why this error happened: a.errors

a = Album.new(:name => "Another Album", :price => 20)

a.save

a – note it has an id now, it has created\_at, updated\_at

You can also do something like a.price = 100 to change a’s price

Everything generally goes into dev db unless you explicitly say something else (?). The commands in rails c went into the dev db.

db/seeds.rb – I think this file is used to seed the db rather than everyone going to rails console and doing the same commands. Put this in seeds.rb

names = ('a'..'z').to\_a

names.each do |album|

Album.create(:name => album, :price => rand(1..100))

end

Now do rake db:seed. You’ll see all these new albums if you start the server.

Structured Query Language (SQL)

You can do:

* Select
* Project
* Join

**Nations**

Name – string

Latitude – number

N\_S – string

Interest – string

Name Lat N\_S Interest

US 75 N Voting

Italy 40 N Pasta

**Select**

Select – take rows from one table to create a new table

SELECT \_\_\_\_\_ From Table\_Name (Boolean conditions)

Didn’t catch whether he put a WHERE in there

Select Interest=Voting from Nations

Select Lat > 50 From Nations

Select \_\_\_\_ from A

**Project** – take columns from table

Project \_\_\_, \_\_\_, \_\_\_ from table

Project N\_S from Nations

**Join** – don’t think he went over it

**Back to Rails**

rails c

Album.destroy\_all – delete all Albums

Song.destroy\_all

Going to seeds.rb and commenting the earlier code out, then do the following. Note the following allows us to connect entities to one another:

who = Album.create(:name => "My Generation", :price => 10, :release => "1965-12-01")

Song.create(:title => "The kids are alright", :album\_id => who.id)

Btw the following two lines are identical – if the hash only has symbols you can use a shorter notation.

who = Album.create(:name => "My Generation", :price => 10, :release => "1965-12-01")

who = Album.create(name: "My Generation", price: 10, release: "1965-12-01")

He put in a bunch of stuff into seeds.rb (three albums which each have some songs) and ran rake db:seed

**Back to console**

rails c

Album.all

Album.find(32) – you can look up an id

Song.find([2, 3]) – look up ids 2 and 3

Song.where("album\_id = 32")

He said – avoid SQL injection. Put the below ? to sanitize it and avoid SQL injection.

Song.where("album\_id = ?", 32)

Song.where("album\_id = ?", #{user\_input}) – if you use a format like this then Rails will check the user input to make sure SQL is not run

Song.order(:title) – will sort things

Song.order("title") – does the same, I think

Take album id of 34 and sort it by name

Song.where("album\_id = ?", 34).order(:title)

**Project**

Song.select(:title)

Song.select("title, album\_id")

Get stuff that has some string in it

Song.where("title like '%to%'") – look for songs whose title has “to” in it with some stuff before and after. Note there are single quotes around %to%.

SELECT COUNT(\*) FROM "songs" – allows you to aggregate

Album.average(:price) – aggregate

Album.first

Album.first.songs

The source for all this is the RailsGuides

guides.rubyonrails.org/active\_record\_querying.html

**6-Nov-2014**

Talked about @songs = Song.all in app/controllers/songs\_controller.rb

Also in the same file,

@song = Song.find(paramds[:id])

When we look at a song (after running rails s):

Processing by SongsController#show as HTML

Parameters: {"id"=>"1"}

Song Load (0.2ms) SELECT "songs".\* FROM "songs" WHERE "songs"."id" = ? LIMIT 1 [["id", "1"]]

Album Load (0.2ms) SELECT "albums".\* FROM "albums" WHERE "albums"."id" = 0 LIMIT 1

Rendered songs/show.html.erb within layouts/application (2.7ms)

Completed 200 OK in 53.0ms (Views: 51.3ms | ActiveRecord: 0.4ms)

You can see that it’s using the show method with a certain id. This corresponds to the show method in which there was @song = Song.find(params[:id])

You can do GET /songs/1 or GET /songs/1.json to do this

The URLs are mapped to controller methods in: config/routes.rb

You can either specify them one by one or use :resources and Rails will generate them.

For “create” method (in songs\_controller.rb)

@song = Song.new(params[:song])

The argument is whatever the params of the song are (that the user put in). If you do rails s and create a new song you will see the Rails hash that corresponds to the user input.

This association of user input, to, e.g. the Rails object :song in the above is done in the view I think. The view corresponding to the form that the user filled in will be mapped to Rails objects, and we’ll see more of this soon.

If you want to see JSON, then instead of doing localhost:3000/songs, do localhost:3000/songs.json.

**He said @song is an ivar but that doesn’t make sense to me.**

SQL

Join

Album

id Name …

1 A

2 B

3 C

Song

id Title Album\_Id

1 X 1

2 Y 2

3 2 1

Join TableX and TableY ON \_\_\_ (Boolean in the blank)

Join Album AND Song ON Album.id=Song.Album\_id

id Name … Sid Title Album\_id

1 A 1 X 1

1 A 3 2 1

2 B 2 Y 2

Say songs have reviews, but albums don’t. But you want an album to have a review based on some average of the reviews of the songs within that album.

rails g scaffold review rating:integer song:references

Go to app/models/review.rb

belongs\_to :song

attr\_accessible :rating

rake db:migrate

annotate

song.rb in models, put in:

has\_many :reviews

**albums.rb – “through association” in Rails, put (?):**

has\_many :reviews, through: :songs

review.rb

Add: attr\_accessible :rating, :song\_id

**Bug**

Change app/views/reviews/\_form.html.erb

Change f.text\_field :song to:

f.text\_field :song\_id

**How do you get the review into the songs view?**

**My song doesn’t show up in reviews**

**Does review’s attr\_accessible need to include song?**

rails c

Review.all

Song.find(2)

Song.find(2).reviews

Album.find(33)

Album.find(33).reviews

Album.find(33).reviews.average(:rating)

**We’ve seen two big things in ActiveRecord, queries and associations (?). Now validations.**

guides.rubyonrails.org/v3.2.19/active\_record\_validations\_callbacks.html

Validations only happen with the methods create, create!, save, etc.

Destructive methods (with !) raise exceptions.

Some methods skip validations, so use them with caution.

Validation helpers – important.

models/album.rb

validates :name, :length => {:minimum => 2}

Now if I try to save something with length 1, it won’t work

You can also do stuff like :maximum, e.g. below (note that I’m using other hash notation):

validates :name, length: {maximum: 5}

validates :price, numericality: true

Also can do:

validates :price, numericality: {greater\_than: 3, even: true}

models/review.rb

validates :rating, presence: true, numericality: {less\_than: 5}

Can do your own error message  
validates :rating, presence: true, numericality: {less\_than: 5, message: “is no good”}

If you want greater than and less than, you can use “in” e.g. in 1..5 (use a range I think).

**Validating relationships**

Song – when you put in album\_id, does it exist in album table?

song.rb in models, add:

validates :album\_id, presence: true #this doesn’t actually validate the album id’s existence

So comment this out and instead do:

validates :album, presence: {message: “doesn’t exist”}

Same for review.rb

validates :song, presence: {message: “doesn’t exist”}

**Why does putting in a new album ask for album id instead of album name? Are all these relationships (in terms of how they’re shown in the view) is through id?**

**Migrations**

Migrations – Ruby DSL (domain-specific language)

Any change to a db is recorded as a migration

db/migrate – contains all the migrations so far, and they’re all timestamped.

The way it keeps track of the migrations is in db/schema.rb

:version – it’s using this version

**How does this :version mapping relate to the migration .rb files in db/migrate?**

When you make changes, it will compare what you’re working with to those migrations.

Look at the create\_albums migration (with the long timestamp)

Rails 4 – can do change now, instead of up and down from Rails 3. Rails 4 is better in this way.

**7-Nov-2014 recitation**

First scaffold user model – name, email, customer\_id, some other fields

Add devise then bundle install. rails g install (?)

Add user (?) – rails generate devise user (?)

Did migration – rake db:migrate

When generating the migration, it may not have the .rb at the end (in db/migrate)

rails destroy user

He thinks drop drops the rows, restart starts from scratch (?) – TA

grep into rake -T for ?

Think rake -T gives you a list of things you can do with rake

Think he then did:

rake db:drop

rake db:migrate

This dropped stuff

Then it worked when running the project in rails s.

There’s a way to make a user logged in before they can see an page, he thinks it’s in application – before action, do user authenticate or something.

user\_signed\_in? (in app/views/layouts/file) tells you whether the user is signed in (?)

To show that the user is logged in as X. It has to do with user\_signed\_in – he added code in app/views/layouts/file. Along the lines of

Logged in as <strong><%= current\_user.email %></strong>

Stuff like this.

OmniAuth – lets you log in via Facebook, Twitter, maybe LinkedIn

Pierre knows how to use OmniAuth, and is going to try to use OmniAuth along with devise.

**11-Nov-2014**

Why we need migrations. The old way is we go in manually and make changes we want. But this is then hard to keep in sync.

db/migrate – the migrations are here, note the timestamp (when the migration was made).

Note create\_albums.rb, you see it inherits from ActiveRecord::Migration which has change method and up and down (think he said “change” is an ActiveRecord::Migration method that e.g. CreateAlbums is overriding). The change method says that there is a table with certain fields which have certain types.

When things get complicated, you might need to define up and down methods instead of change methods. But we shouldn’t have to do that for now.

Do:

rails g migration --help

It shows you stuff about up and down migrations, and what happens when you generate a migration.

Do:

rails g migration AddDescriptionToReview description:text

rake db:migrate

rails s – don’t see it. This is because you need to update the view. You can either update the view yourself or generate the scaffold again. The latter might not be a good idea, might override a lot of stuff (?). I think he said scaffold lets you skip certain things though.

Updating the views, first delete the views

rm -rf app/views/

rm -rf app/views/reviews/

rm -rf app/views/reviews/\*

rails g scaffold Review rating:integer song:references description:text -s

The -s option tells it to skip existing files.

This totally screwed up my spotify (think I wasn’t supposed to run the first two “rm” commands, he was just tabbing out to see what was in those directories)

We can do:

rake db:rollback

This will roll back the migration

Then do: rake db:migrate to go ahead a migration again

guides.rubyonrails.org/v3.2.19/migrations.html – good resource on migrations. There are lots of methods e.g. add\_index, remove\_column, etc.

**I did this to fix the error I made before**

rails generate scaffold album name:string price:float release:date -s

rails g scaffold song title:string length:time album:references -s

rails g scaffold playlist title:string description:text –s

rake db:migrate

**[copied from before**

**If you want to install Bootstrap into your app (using the less version)**

**rails g bootstrap:install less**

**Now I want to use the Bootstrap layout (put “application” so it persists throughout)**

**rails g bootstrap:layout application fixed**

**Conflict with a file – let it overwrite because it’s all auto-generated.**

**Now make albums Bootstrap themed**

**rails g bootstrap:themed albums -f**

**(-f forces – update the pages)**

**(have to have the name in plural form because that’s what Bootstrap wants?)**

**]**

View will never talk directly to model and vice versa. It all happens through the controller.

Routes – a URL is mapped to a controller. The controller then talks to the model and view.

Let’s look at app/controllers/songs\_controller.rb

index

Creates @songs which contains all the songs in the database.

Convention over configuration: the view will be the same name as the method. So “index” is the method in the songs controller, therefore the view name is index.html. Same, for the “show” method, this will call show.html.

However there is no page “upate”, so if I didn’t have these redirects and renders, you’d get an error.

All the routing happens in the config folder. Go to config/routes.rb – this controls all your routing.

Seven things when you declare a resource. index, show, new, create, edit, update, destroy, some others.

In routes.rb, the line “resources :reviews” – all this does is create 7 routes. It maps 7 URLs (/reviews/index, /reviews/show, etc.) to 7 controller methods (index, show, etc.).

You can also exclude certain of these routes from being created from the line resources :reviews.

In routes.rb, you could do:

resources :albums, :except => :destroy

Or you could do:

resources :albums, :except => [:destroy, :show]

Higher in routes.rb file – higher priority. It’ll just go in the order in which it’s written. If you’re having routing issues, it might be because the order is messed up, and rake routes will help you. All the routes in rake routes are in order of priority.

“Route globbing” – route any un-identified URL to something, e.g. to the root. Can put this at the bottom of your routes.rb file.

Top of routes.rb, under routes.draw put:

root :to => ‘albums#index’

This route will ensure that this gets re-directed to albums#index

This didn’t work by going to the root when doing rails s.

Keep in mind hierarchy – public folder is looked at first (static things), and then if it doesn’t find it, it will go to the routes. So go into public, delete index.html then re-run rails s and go to the root, which should now give you the list of albums.

**This didn’t work.**

Nested resources

Say songs can’t exist on their own. Songs are somehow dependent on albums.

In routes.rb, comment resources :albums line

resources :albums do

resources :songs

end

Then comment out resources :songs

Songs can’t be accessed directly, only through the albums that the songs belong to.

Now look at rake routes (and compare to the annotated routes.rb file on the bottom). We used to be able to do /songs/stuff. Now to do this, you have to go to /albums/:album\_id/songs/

Why do we have the “root :to => ‘albums#index’ at the top? Anywhere else, it will work fine. So is there a reason to have it first? The reason is speed. The first created is highest priority. So whenever a request comes in, it matches it one at a time. So for common routes, put them high up.

Let’s edit the songs\_controller to make sure it returns the songs correctly when the user wants to get the songs for an album. In index method:

def index

if (params[:album\_id]) then

@songs = Song.where(‘album\_id = ?’, params[:album\_id])

else

@songs = Song.all

end

respond\_to do …

end

We still get an error, songs\_path not found. Created all the paths initially, but since we nested it, it’s not available anymore. In routes.rb, do this hack:

Uncomment: resources :songs (outside the block)

What we should have done is fix the views.

app/views/songs/edit.html.erb

<%= link\_to ‘Back’, songs\_path %>

songs\_path is stale. This was an old route which is now trying to be called, will result in an error.

**13-Nov-2014**

Swap not here, it’s Josh and Pierre (basically doing a recitation)

user twitter facebook

email name name

password id vid

email

picture (?)

Made user omniauthable

Then smthg else

Koala

FbGraph

DB guardians

model

Application

controller 🡪 USER FIND

routes

Application view = user

before\_filter set\_user (?)

after\_filter

application\_controller

Wherever you are in the app, don’t let anyone do anything unless they are authenticated.

Whenever you have a controller, it comes from ApplicationController (application\_controller.rb)

Be careful of putting stuff in the application html file

Let’s implement a “like” button for the list of quotes.

Do we need a controller? Most of the time you have a controller per model but sometimes you can have separate controllers.

You need a model (Like model).

A like will link a user and a quote. So we can use either the user or quote controller to deal with this, we don’t need a new controller. So we’ll have a Like model and we’ll deal with it in one of the two controllers (either User or Quote controller).

rails g model like user:references quote:references

Go to index.html.erb, in the bit that it loops through the quotes making <td> entries for each quote / button (“Show” button, “Edit” button, etc.), do:

<td><%= link\_to ‘Like’, like\_quote\_path(**quote: quote.id**) method: :post %></td>

This like\_quote doesn’t exist yet, so go to config/routes.rb

post ‘like/quote’ => “users#like”

The ‘/’ in ‘like/quote’ above replaces the \_ we had in the link (I assume in the html erb)

As the above route shows, we’re going to implement the method in the users controller.

We forgot to do the has\_many and belongs\_to. We do this to be able to do:

@user.likes

@quotes.likes

To do these .likes, we have to specify how they’re connected through has\_many and belongs\_to.

Go to models/quote.rb

class Quote < ActiveRecord::Base

has\_many :likes

end

In models/user.rb, under all the devise stuff, I think he also did has\_many :likes

In models/like.rb, it already had belongs\_to :users and belongs\_to :quotes, I think this is because we specified that the Like model references the Users and Quotes models.

In users\_controller.rb

On the top line before\_action, he added :like to this list

Also, put this (not in private):

def like

newLike = @user.likes.new # don’t add this to the database yet (i.e. don’t use create)

quote = Quote.find(params[:quote])

newLike.quote = quote

newLike.save

redirect\_to quote\_path # go back to the index of the quotes

end

On first line of users\_controller:

before\_action set\_user

This is saying that before any of the methods are called from users\_controller, set the user. set\_user is a private method in users\_controller that sets the user. Also the line is more like:

before\_action set\_user, :only => {:new, :edit, …}

I think it’s saying only before the given methods (:new, :edit, etc.) do you need to run set\_user.

rake db:migrate

rails console

User.all

Like.all

Josh

Likes.All

User.find(1) – returns the User with id=1

User.find!(100) – will throw an error because there is no id=100. User.find(100) would just return an empty array and no error. Using ! signifies you know what you’re doing.

User.find([2, 4, 8]) – returns an array of three users with id 2, 4, and 8

Pretend users have names

User.find\_by first\_name: “Josh”

str = “there is beer”

Quotes.find\_by quote:str

Think most / all of these methods also have a ! version e.g. find\_by!

Multiple conditions

User.where(email: “foo”, name: “bar”)

The “find” returns one result I think (LIMIT 1). So if you want multiple results, then use “where” – Quote.where(author: “Arvind”)

Also you can use a “where” clause to show all the quotes I liked that someone else posted (a little more complicated).

rake db:seed

In index.html.erb

<%= form\_tag search\_path, :method => ‘get’ do %>

<p>

<%= text\_field\_tag :search, params[:search] %>

<%= submit\_tag “Search”, :name => nil %>

</p>

<% end %>

Think he did this in routes

get ‘search’ => ‘quotes#search’

In quotes\_controller.rb:

def search

@quotes = Quote.where(“text LIKE ? OR author LIKE ? ”, “%#{params[:search]}%”, “%#{params[:search]}%”)

# think he said the % means anything (wildcard), we don’t care what comes before and after

render :template => “quotes/index”

end

**Office Hours 17-Nov-2014**

Short answer – use has\_and\_belongs\_to\_many, and put this in on both sides of the model. Then create a join table. Then it should work.

rails g migration RemoveSpecialtyIdFromPhysicians specialty\_id:integer

rails g migration CreatePhysSpecJoinTable physicians specialties

rake db:migrate

Now we have to assign an array of Specialty objects to a physician’s specialties. Can create the Specialty(s) first and then assign them to the physician.

aa = Physician.find(1)

aa.specialties = [Specialty.find(1)] #it wants an array since it’s many to many

# also, this should have inserted it into the db (i.e. not just into “aa” which is “floating around”), to check this do:

Physician.find(1).specialties

**Side note –** he thinks to create an instance in another model (e.g. create “NJ” by creating a physician who belongs to “NJ” before “NJ” even exists in the State model), we have to do “accepts nested attributes for”.

**Side note –** to update schema.rb, do rake db:schema:dump

**18-Nov-2014**

Two options for adding functionality to resources: add to individual members of the collection or to the entirety. Do I want something specific to id=26 or do I want something that works overall regardless of the individual id.

In routes.rb:

resources :albums do

resources :songs

end

Within this do-end, put:

member do

get ‘rating’

end

Think this will get the rating for each member (song) of the album?

He’s got a bunch of stuff in seeds.rb to seed the database. At the top, he puts these three lines to first remove all the data, then he adds stuff to the database.

Song.destroy\_all

Album.destroy\_all

Review.destroy\_all

After changing seeds.rb, he runs:

rake db:seed

/albums/37/rating – you want the ratings for the songs in album 37. For now, we get “Unknown action”, “the action ‘rating’ could not be found for AlbumsController”

We did the route (member do from above which was in routes.rb) but we need to add the method ‘rating’ in albums\_controller

Because we have this :through association (songs through albums), we can do Album.reviews to get to the songs. Otherwise we’d have to do Album.songs.reviews.

def rating

Album.find(params[:id]).reviews.average(:rating)

end

Template is missing

Need to add views/albums/rating template

New file views/albums/rating.html.erb

Change controller to:

def rating

@rating = Album.find(params[:id]).reviews.average(:rating)

end

Now we’re storing the average rating for an album in @rating. We use this in the view. The file rating.html.erb has:

<h2> Album Rating </h2>

The Album Rating is <%= @rating %>.

/albums/38/rating, /albums/39/rating, now should work

So we set up the route (the member do get ‘rating’ end – does this create a valid URL /albums/:id/rating?), then we set up the controller method rating, then we set up the view rating.html.erb

Think the route, because it is within the resources :albums code, will look for the “rating” method in the Albums\_Controller.

This is convention over configuration for Rails:

You could do this:

match ‘photos/show’ => ‘photos#show’, via => get

So the URL photos/show is mapped to the show method in photos.

The shorthand method rather than the above is: get ‘photos/show’ which is identical to the longer string.

You have ratings\_path, ratings\_url available through this btw.

The name of the route is rating\_album.

Now let’s do something across all albums (not a “member” functionality), e.g. what if we want to sort albums.

So if we do /albums, we get default order, and if we do /albums/sort, then we want the albums to be sorted in a certain way.

First, we set up the URLs we want as above.

Second, add this to routes within resources :albums do-end:

collection do

get 'sort'

end

Third, in AlbumsController, have to define sort method.

def sort

@albums = Album.order(:name)

end

Still haven’t defined views so the views won’t work (template is missing will be the error).

Now I could create sort.html.erb, OR (DRY) re-use some of the existing views – e.g. want to use index.html.erb in some way since that is already made to show a list of albums.

In albums\_controller.rb:

def sort

@albums = Album.order(:name)

render :index

end

Note that render :index is NOT going to call the index method (key point of confusion). Instead it will go to the index view and show it to the user.

Why would that work? The index view is written in a DRY manner – it has @albums (doesn’t care how it’s generated), so the index uses this and generates the view with it.

Also, instead of:

member do

get ‘rating’

end

You could do:

get ‘rating’, on => member

Same for:

get ‘sort’, on => collection

You would do this rather than the block syntax (they’re equivalent) if it’s short, but if you’re doing a lot of stuff in the block then you should use the block.

Now alter routes.rb

collection do

get ‘sort/:field’ => ‘albums#sort’

end

The parameter “name” if we do albums/sort/name, will be available to the controller.

@albums = Album.order(:name)

has to be

@albums = Album.order(params[:field])

Now it will work if you do albums/sort/price, or albums/sort/release

If someone puts a random field to sort in e.g. albums/sort/dfkjsf, then you get a bad error. The worst thing about this is it puts dfkjsf right into SQL (injection concern).

This problem only happens with order. It doesn’t do this with find. With order you can’t even do the ? sanitization e.g. if you do:

Album.order(“?”, params[:field])

This does not sanitize your SQL in the case of order.

To fix it:

One easy way: just check that the input is a column name.

@albums = []

if (Album.attribute\_names.include? (params[:field])) then

@albums = Album.order(params[:field])

end

render :index

Note that Album.attribute\_names gives you the list of attribute names in Album.

Translated Paths – on Rails guide for routing. This can be useful if you’re doing internationalization.

Aspects of Agile Programming:

* Pair Programming – we won’t do this. In pair programming, you work with another person. Swap has had hit-or-miss experience with it, and thinks in order to do it well you need to find a good pair (you and your partner need to be roughly at the same level of programming knowledge).
* Automated testing. Two things:
  + BDD
  + TDD – before you write the code, you write the tests first. Think he said we won’t do this?
* Refactor mercilessly. If you have good tests, you can refactor and hope that your tests will catch if something has broken.

**Unit Testing**

* Test a chunk of code at a time – a “unit”.
* Usually a method or even a part of a method if the method is large.

Btw: the Cucumber / Capybara testing is *not* unit testing because it’s using multiple methods at a time. It’s doing something with the controller, model, view, back to model, etc. It falls under system integration testing. You’re testing how all the pieces fit together.

FIRST

* Fast – don’t have the tests take two hours
* Independent – should be able to run tests in any order. No dependencies – a test can’t assume another test did something e.g. created something in the database.
* Repeatable – refers to external dependencies. If I use an external API and I can only do 1000 queries a day, then this is not a repeatable test.
* Self-checking – did it pass, did it fail, this needs to be automated. A human shouldn’t be reading the results of the tests and interpreting it.
* Timely. Two things:
  + TDD – want to write tests before code. Will not insist on this in the class.
  + In the context of this class, if you have code you should write some tests, and if your code changes, you have to update the tests. You should be updating tests at the same time as the code.

In Ruby, the default testing framework is MiniTest. It lets you do unit testing as a behavior. That framework in Rails is called RSpec.

romanbug.rb, testroman.rb

In testroman.rb:

require\_relative ‘romanbug’ # have to include the code we’re testing, require\_relative checks only this path

require ‘minitest/autorun’ # minitest provides Ruby testing functionality, autorun lets you run Ruby file as normally plus gives you results of tests

class TestRoman < MiniTest::Test

def test\_simple # convention is to start tests with “test\_”

assert\_equal(“i”, Roman.new(1).to\_s)

assert\_equal(“ix”, Roman.new(9).to\_s)

end

end

Can run this now and see results of these tests.

Let’s add more tests:

def test\_more

assert\_equal(“ii”, Roman.new(2).to\_s) # this failed

assert\_equal(“iii”, Roman.new(3).to\_s)

end

def test\_range

# no exception when I run these, because I should be able to run within 1 and 4999

Roman.new(1)

Roman.new(4999)

# can’t run Roman for 0 or 5000

assert\_raises(RuntimeError) {Roman.new(0)}

end