\*Also look at the word doc: basics\_of\_ruby\_steps\_for\_refrsher for more useful info and refreshers

**Userful urls to look at**

* Getting started in rails, the basics of rails, and installing: <http://guides.rubyonrails.org/getting_started.html>
* command basics (terminal commands): <http://guides.rubyonrails.org/command_line.html>
* To see data types for columns and useful info (model stuff): <http://api.rubyonrails.org/v5.1.3/classes/ActiveRecord/ConnectionAdapters/SchemaStatements.html#method-i-add_column>
* For advanced and basic model generators - terminal commands - <https://railsguides.net/advanced-rails-model-generators/>
* rails migration cheatsheet - <https://gist.github.com/amejiarosario/2950888>

**USEFUL CODE**

EMAIL\_REGEX = /\A([^@\s]+)@((?:[-a-z0-9]+\.)+[a-z]+)\z/i

<input type="hidden" name="authenticity\_token" value="<%= form\_authenticity\_token %>">

date\_here. strftime("%B %d, %Y")

date = Time.new().to\_date - this creates a date for today that can be used, compared to, etc.

**TO START A PROJECT**

Navigate to your desired directory in the terminal

To Start Project- type: **rails new AppName --database=postgresql**, to create a new folder with rails installed using postgres. If you leave out the “--database=postgresql” part, then it will use SQLlight

* ex: **rails new product\_dashboard --database=postgresql**

If you’re using postgres, then create the database by typing: **rails db:create**, you should do this after creating the project, and preferably before generating your models, though that may not matter. This is not finalized until you type **rake db:migrate**, but that can wait until after you make your models

Install Useful Gems

* Copy and paste the below gems into the Gemfile
* type: **bundle install** when done
* Starting from the # Use ActiveModel….. below, you can copy and paste that over the bcrypt gem, which is already in the gemfile, just commented out
* After bundle install type: **rails g rails\_footnotes:install** for the rails-footnotes gem, and then type: **rails g rspec:install rspec.** for rspec (testing)
* In order to store your sessions data in actual sessions, instead of the browser’s cookies (which has limited space), after you bundle install the gem ‘activerecord-session\_store’ below, follow the comments after it. Type: **rake db:sessions:create**, then type: **rake db:migrate**. After that, go to the session\_store.rb file in the initializers folder in the config folder, and change the code “:cookie\_store” to “:active\_record\_store”. Restart the server if necessary, and you’re good to go.
* To be able to use bootstrap with rails, see the “using\_bootstrap\_with\_rails” word doc, you need to make some changes to the application.css and .js file to use bootstrap
* See below the RSPEC/Testing section for what you need to copy and paste into the spec/rails\_helper.rb file

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* START OF GEMS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

gem 'faker'# allows you to fill the DB with random data for names, titles, content, etc. github can be seen at: https://github.com/stympy/faker#fakergameofthrones

gem 'hirb' # type: Hirb.enable in the rails console to better see query results. In the event that you get a query result too large, and it shows “(END)”, just type “q” to exit, you can also scroll up and down the results with the arrow keys

gem 'rails-footnotes', '>= 4.0.0', '<5' # allows you to see footnotes in the web browser with useful info, after pasting in gemfile, type: rails g rails\_footnotes:install

gem 'bootstrap-sass', '3.2.0.2'

gem 'sass-rails', '~> 4.0.3'

gem 'pg' # Gem for Rails to interface with PostgreSQL

gem 'activerecord-session\_store' # this allows you to use the sessions instead of the browser's cookies to store data. After bundle install type: "rake db:sessions:create" followed by "rake db:migrate". Then go to the directory inside this project /config/initializers/session\_store.rb, and change the ":cookie\_store" to ":active\_record\_store", after restarting your server, the data will now be stored in sessions instead of cookies

# Use ActiveModel has\_secure\_password

gem 'bcrypt', '~> 3.1.7'

group :test, :development do

gem 'rspec-rails'

gem 'database\_cleaner'

end # this allows use of rspec for testing, after pasting, type in the terminal:

# rails g rspec:install rspec.

group :test do

gem 'capybara'

end

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* END OF GEMS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

To create a new controller type: **rails g controller ControllerName methodName methodName**

* ex: **rails g Users index show create etc**.
* Basically, after g (generate) is the name of the controller which needs to be plural and capital, in the folder if will be users\_controller. After the controller name, separated by only a single space, are the methods which will be put into the controller upon creation, just saves time, not necessary to include the methods here.

To create new table/model - type: **rails g model ModelName ColumnName ColumnType**

* ex: **rails g model User first\_name:string age:integer**
* table name needs to be singular and capitalized, the column names followed by their data type (integer, string, etc.) separated by just a colon mark( : ), and separated from other column names by a single space
* IMPORTANT NOTE: you don’t need to write out a created\_at or updated\_at column, those two columns are both automatically added for each model
* IMPORTANT NOTE: if the data type for a column is a string, then you can leave the data type for that column blank, ex: **rails g model User first\_name last\_name email age:integer**
* FOR PASSWORDS: You write “password\_digest” as a string when creating the model, see further details in the password section below. ex: **rails g model User first\_name password\_digest:string**
* To see examples, see the txt files in the Rails\_models folder, in the ruby folder, in the Dojo\_projects, on the desktop
* See the section “Generating Models Info” below for more detailed and advanced info
* Possible data types for columns you can use: :primary\_key, :string, :text, :integer, :bigint, :float, :decimal, :numeric, :datetime, :time, :date, :binary, :boolean.

To migrate (finish and implement the changes to your project) your models once you are finished, type: **rake db:migrate**

* Important Note: you should check the db folder in your rails application, the migrate folder which will show the tables you’re trying to make, one file per model. Double check that they are how you want them to be before you use rake db:migrate, otherwise you will have to do more migrations before you fix them.

To destroy say a model or controller you generated with rails g, you can type: **rails destroy model/controller NameHere**

**TO RUN PROJECT/TERMINAL COMMANDS**

To run a project/start a server type: **rails s**, in the terminal. Go to <http://localhost:3000> to see it, type ctrl+c to quit/shut down the server

* you can also specify the port by typing: rails s -e production -p 4000

**USING PASSWORD WITH MODELS**

* See “belt\_review\_events” project in full\_rails for examples and comments
* When creating the model, use “password\_digest” as a string, ex: **rails g model User first\_name last\_name password\_digest:string** (I write :string after it just to be safe, if left blank, it will automatically be a string)
* Next, after you use **rake db:migrate**, go to the model file, and include “**has\_secure\_password**” on its own line of code. Rails recognizes this and it allows you to save a hashed password, use a password\_confirmation, and allows you to use an authenticate method
* Make sure that the gem **bcrypt** is included in the gem file and installed
* use validations like: **“validates :password, :password\_confirmation, presence: true, length: {minimum: 6}”** and “**validates\_confirmation\_of :password”** for decent additional protection.
* Use the authenticate method, which is allowed because of the “has\_secure\_password” that is included in the model. For example, if I created a user with a password of “foobar”, then I could plug that user into a variable with “user = User.find\_by(email: “j@j.com”)”. Then, user.authenticate(“foo”) would return false, but user.authenticate(“foobar”) would return the actual instance or user itself, which would register as true to a boolean.

**Generating Models Info**

* ex: **rails g model Comment comment:text article:references**
* The references data type is used for foreign key columns, in this case it would create article\_id
* you can type: “--parent tableName” when creating a model, and it basically inherit everything from the parent table. Meaning, if I typed: **rails g model admin --parent user**, then it would create the model admin, with the same columns, data types, references, etc. as the user model.

**RSPEC/Testing**

* To run all the tests in the project, type **bundle exec rspec**  in the project main directory.
* Copy and paste the below code into the “rails\_helper.rb” file in the spec folder. Go to the “RSpec.configure do |config|” section, the last part, and copy and paste the below code over it, copy and paste as is, the indentation will be correct.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* COPY AND PASTE CODE STARTS HERE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

RSpec.configure do |config|

config.fixture\_path = "#{::Rails.root}/spec/fixtures"

config.use\_transactional\_fixtures = true

config.infer\_spec\_type\_from\_file\_location!

config.include Capybara::DSL

config.before(:suite) do

DatabaseCleaner.clean\_with(:truncation)

end

config.before(:each) do

DatabaseCleaner.strategy = :transaction

end

config.before(:each) do

DatabaseCleaner.start

end

config.after(:each) do

DatabaseCleaner.clean

end

end

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* COPY AND PASTE CODE ENDS HERE\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

d