Tony's Ruby App - office hours

create application

---

rails new Football_app -T

-T skips creation of testing files

cd into application directory

---

bundle install

create model

---

rails generate model attribute1 attribute2 attribute3

consider the attributes of the model you are creating. In this example we are creating a Football_app.

Team = model name - MUST BE SINGULAR
and the command used to create our Team Model is:

rails generate Team name:string logo:string city:string wins:integer losses:integer

This action create the Team.rb file and a migration file.

create database

---

rake db:create

you only create the database ONCE!

populate your database with a table

---

rake db:migrate

!! make sure your server (elephant) is running!!

This action creates an empty table with the columns named for the attributes identified when you created your model. so you now have a table that mimics your model

(if you use the PLURAL you are referencing the TABLE,
and if you use the SINGULAR you are referencing the MODEL.

Now you are able to use ACTIVE RECORD SYNTAX.

step into your database

---

rails c

this opens an IRB session that is linked to your application

begin inputing data in your database

Team.new

opens a new row in your table that is an object and stores it in a variable locally.

---

raiders = Team.new

following this command you should be able to check it's creation by typing "raiders" which should return an empty object

---

raiders.name = "Raiders"

this is the template for populating your database. Do the same with all your attributes.

---

raiders.save

saves all the information

check your work

Team.first

**alternatively you can populate your database via seeds.rb

Team.create({name: “Raiders”, logo: “Pirate”, city: “Oakland”, wins: 3})

the seeds.rb file is located in the database folder

grab data from seeds.rb and populate it into the database

---

rake db:seed

repeat all processes for any other models

---

did you screw it up? go back to square one…

---

rake db:migrate VERSION=0

!!DO NOT DESIGN YOUR APP UNTIL YOUR DATA WORKS!!

establish a relationship between models

Team has_many Players

Player belongs_to Team

there are 2 places in which these that these relationships need to be identified.

---

rails g migration AddTeamToPlayer team:references

team:references AND the AddTeamToPlayer creates a column in the Player Table with the foreign key.

---

rake db:migrate

then double check

---

rails c

---

player.new

you should see your TEAM ID column added to your Player object!

AND NOW IN THE ACTIVE RECORD

in the model files

---

in the team.rb file has_many :players

the colon is attached to the PLAYERS and NOT to the HAS_MANY

---

in the player.rb file belongs_to :team

the colon is attached to the TEAM and NOT to the BELONGS_TO

To check -
EXIT IRB and RE-ENTER

player.first.team should return your first player with a team id attribute as a foreign key

team.first.players should return your first team with a players id attribute as a foreign key

create setters for the team id and players team

Romo = player.first

Romo_team_id = 1

Romo.save

to check

team.first.players	should return to you	Romo
player.first.team	should return to you	Cowboys

* these return values are samples based upon the class example.*

COMPLETE FOR ALL MODELS AND RELATIONSHIPS

NOW THAT THIS IS WORKING CORRECTLY WE CAN BUILD OUR FRONT-END OF THE APPLICATION

build the controller and the views

---

rails g controller teams index show create new edit delete

create an instance variable within the controller

---

in the Teams controller @teams = Team.all

and in the corresponding view

```
<h1>Football Teams</h1>
<% @teams.each do |team| %>
<ul>
    name: <li><%= team.name %></li>
</ul>
<%end%>
```

establish your routes

```
root 'Teams#index'
```

save and start server

---

in the browser localhost:3000/teams/index

and to display the players on a team

```
<% team.players.each do |player| %>
<%= player.name %>
<%end%>
```