Extending your first Rails app

1

Remember: Don't get discouraged if you aren't understanding all of this the first time through. We're just introducing a lot of concepts that we'll cover with more depth as we progress through the class.

Yesterday, we created our first Rails app (which means, of course, that we are totally rad). Today, we'll **extend** the functionality of that app.

One of our clients (today, we'll pick on Harsh) has called up in a tizzy because he can't actually upload pictures. "What good is this app if I can't upload pictures?" he cries. Psssh… clients.

We'll start by adding the following line to the bottom of our `Gemfile`:

```
gem 'carrierwave'
```

The carrierwave gem gives us the ability to upload files to our Rails application. We'll implement the gem inside of our app so that Harsh will calm down.

2

Now that we've added a new gem to our `Gemfile`, we have to alert bundler that we need the new gem installed:

```
bundle install
```

Anytime you change the `Gemfile`, it's good practice to commit the change:

```
git commit -am "Added carrierwave gem."
```

3

Let's use the new **generator** that carrierwave has given us:

```
rails generate uploader Picture
```

4

Now we'll add some functionality to the `Idea` model. Inside of `app/models/idea.rb`, under the line:

```
class Idea < ActiveRecord::Base
```

add the following:

```
mount_uploader :picture, PictureUploader
```

5

Now we need to make a few changes to the form for ideas in order to enable picture uploads. Inside of the file `app/views/ideas/_form.html.erb`, let's change the line

```
<%= f.text_field :picture %>
```

to look like this:

```
<%= f.file_field :picture %>
```

And then change the line

```
<%= form_for(@idea) do |f| %>
```

to look like this:

```
<%= form_for(@idea, :html => { :multipart => true }) do |f| %>
```

Now let's make sure that it worked.

6

Awesome! We've successfully implemented a new feature for our client. Let's be sure to commit our work:

```
git add .
git commit -m "Added ability to upload pictures."
```

Our first deployment

7

So this is all well and good, but it's not on the Internet, so who cares? Let's fix that!

We'll use the excellent Heroku as our hosting provider. We can't use GitHub Pages for this project, since GitHub

Pages only works for static HTML pages.

First, let's create a new app on heroku to house our code. Use your full name in place of `your-name` below. (Duh.)

```
heroku create your-name-ideator
```

8

Before we push to heroku, we have to switch to using postgres as our database, rather than sqlite. Heroku uses postgres on its servers, and we want to be cool like them. Also, it won't work unless we do. Open up the `Gemfile` and replace `gem 'sqlite3'` with `gem 'pg'`. Then run a `bundle install` and commit your changes and push them to both partner's remotes.

9

The `heroku create` command does a couple of things. First, it sets up a place for our code to live on Heroku's cloud infrastructure (buzzword alert). It also creates a new **remote** for our git repo called `heroku`. To deploy our app, we simply push our code to this new remote:

```
git push heroku master
```

10

After that command finishes (it will take a while, the first time), run the following command to migrate your production database on heroku:

```
heroku run rake db:migrate
```

11

Now check your app out on the open Internet:

```
heroku open
```

Whoo-hoo! Copy that URL and send it to all your friends. That's your first live Rails app. **TAKE PRIDE.** You are now certifiably more excellent than most people you will encounter.