Enhance. Enhance. Enhance.

1

Yesterday, we extended the functionality of our app by adding the ability to upload pictures. We still can't see those pictures from the app, though, as Shehzan so loudly pointed out when he called us at 3AM last night to report the bug.

"It's easy. Just copy the pictures down from the production server and view them locally," we handily explain.

He doesn't like this idea.

Let's make sure that Shehzan can easily see the pictures folks upload to the app, so we can get a good night's sleep.

First things first - we need to clone our repository down to our computer, and set up the appropriate remotes so that each partner can push the code back to GitHub. I'll start by cloning down my repo:

```
git clone https://github.com/techpeace/ideator.git
cd ideator
```

Now, I need to set up the appropriate remotes for myself and Harsh, my partner for this week:

```
git remote add techpeace  https://github.com/techpeace/ideator.git
git remote add harshpatel https://github.com/harshpatel/ideator.git
```

Note that if your repos have the same name, you can just paste the same repo URL and edit the username to match your partner's.

2

We also need to add a remote to point our app to our Heroku server. The `heroku` command line tool can help us out with this. First, we run `heroku apps` to find the name of our app from yesterday:

```
$ heroku apps
deanandbri-ideator
holly-and-krista-ideator
matt-and-harsh-ideator
```

Ah! There's the one I need - that last one. Now we run the following to find out the URL for the `heroku` remote that we'll create:

```
$ heroku apps:info --app matt-and-harsh-ideator
=== matt-and-harsh-ideator
Git Url:      git@heroku.com:matt-buck-ideator.git
Owner Email:  students@themakersquare.com
Region:       us
Stack:        cedar
Tier:         Legacy
Web URL:      http://matt-buck-ideator.herokuapp.com/
```

Now I just need to add a new git remote named `heroku` that points at that git url, there. Then I'll be able to `git push heroku master` to deploy later on.

Environments to the rescue

3

Yesterday, we also noticed that we had to switch from the `sqlite3` gem to the `pg` gem for our database driver in order to make our app work on heroku. Unfortunately, our local machine isn't set up to use a postgres database just yet, so we want to stick to using sqlite locally so we can add the functionality without worrying about how to set up postgres on our laptops. But how can we use different gems on heroku than we use locally?

Rails uses the concept of "environments" to allow us to use different settings based on where we're running our app. Here are the default environments:

- **development**: This is the environment Rails apps run in by default.
- **test**: This is the environment that we'll use to run tests to ensure that our code runs properly.
- **production**: This is the environment that will be used when our app runs on the server (in this case, Heroku).

Open the file we use to specify which gems are part of our project. Replace the line `gem 'pg'` with the following:

```
group :production do
  gem 'pg'
end

group :development do
  gem 'sqlite3'
end
```

Here, we're specifying that when we run in development (locally), we want to use our sqlite3 database that we'd been using before. When we run in production (on Heroku), use the postgres database that Heroku requires.

After we've done that, we need to install the bundle again, and then commit our changes and push to each partner's repo.

Okay, let's do this

4

In order to display the pictures, we need to open the file that displays an idea. Open up the file `app/views/ideas/show.html.erb`. Change the line

```
<%= @idea.picture %>
```

to this:

```
<%= image_tag(@idea.picture_url) if @idea.picture.present? %>
```

Here, we changed from outputting `@idea.picture`, which was just a string, to outputting an image tag pointing to the URL for the uploaded picture (stored in `@idea.picture`), but ONLY IF a picture has been uploaded for this idea. Load it up in the browser and check it out.

Now commit that change:

```
git commit -am "Display the pictures."
```

5

We've still got a problem. Whenever a user navigates to the root of our app, they're still presented with the default Rails landing page. Let's fix that by deleting the default landing page from our git repo:

```
git rm public/index.html
```

6

Now we need to point the *root* of our application to the `index` action of the `ideas` controller. We would make this change inside of the `config/routes.rb` file. Using the Rails guide on routing, can you figure out how to make this change? You'll know that it's worked if heading to localhost:3000 takes you to the list of ideas.

Alright! Now this thing looks like it's ready for prime time. **Push your changes up to heroku** and find an instructor (they'll be wearing a tweed jacket with suede patches, probably smoking a pipe or something) so you can show off your work.

If you've made it this far, you're doing great! You've learned everything about Rails that we expect you to for today. If you've run out of time, feel free to work on the extensions later at home. If you don't have time to do that, you'll still be all set for what we'll be covering tomorrow.

After you've spoken to an instructor, head outside with your partner, high five next to the nearest tree, and then

take a 10 minute break.

Once you head back inside, help out any of your classmates that haven't reached this point. Once everybody's finished, we can all start working on the extensions.

Don't forget to **commit and push your work to both partner's repos** before you finish up for the day.

Extensions

7

Figure out a way to incorporate the display of pictures into the list of ideas. You can find the HTML code that displays the list inside the file `app/views/ideas/index.html.erb`.

8

Let's do something to make that index view a bit prettier, eh? Keep in mind that you don't even have to use a table to list the ideas, if you don't think that's the best way. Just make sure to include the links to either show, edit, or destroy the idea.

Whoa, way to go! You're just blazing through this stuff. Find an instructor again and explain how you went about changing the index view.

9

Once you've finished that up, start running through the exercises in Rails for Zombies, if you haven't already. If you've already completed that course, give the tutorials on CodeLearn a try.