Lab 2.2: Models with Foreign Keys
IN705 Databases Three

Introduction

Last time we built a simple model that we served with a REST API. Our model only used text fields, however, and we'll need to link some models together using foreign keys to build the full application. We'll do this today by building a new model to represent users' posts. These records will use a foreign key field to associate the posts with their owners. We will also have to add an additional method to our user controller to return a list of a user's posts.

1 Creating our second model

If Twitter posts are called "tweets", then Splatter posts should be called "splatts". We need a model for them. The model only needs two fields for this interation.

1. body: a string
2. a foreign key field that stores the primary key for the associated user.

As before, we'll generate our initial code by running the following command from within your `splatter` directory.)

```
rails generate scaffold Splatt body:string user:belongs_to
```

"belongs_to" is the Rails way to indicate a one-to-many relationship between the users and their splatts. When we run our database migration it will place a foreign key field, `user_id`, in the splatts table. Run the migration now with the command

```
rake db:migrate
```

to create the new table.

Incidentally, now we can see why Rails starts migration file names with a timestamp. Rails keeps track of what migrations have already been done and only runs the ones with newer timestamps. We can also roll the database migrations back by specifying a point in time to which we want to roll back.

You should take a look at your new model, contoller and migration file to be sure you understand what has been done.

2 Tweaking the controller

Examine the controller class in `app/controllers/splatts_controller.rb`. In our application we won't allow users to update splatts, so we should remove the `update` method. Recall from the previous lab that we also need to whitelist input fields so that we can send them to the Splatts constructor method.

```
private

  def splatts_params(params)
    params.permit(:body, :user_id)
  end
```

Use this method to send input into calls to `Splatt.new`. If we wanted to do some more thorough input sanitation, this method would be a sensible place to do it.

3 Create a new splatt using curl

Start your server with the `rails server` command like we did in the previous lab. In a second ssh session, enter something like the following:

```
curl -i -H "Content-type:  application/json" -X POST http://lastname.sqrawler.com:3000/splattss
-d '{"splatt":  {"body":"Hello, Splatter world", "user_id":1}}'
```

This will create a new post belonging to the user with the id value 1. You may need to use a different id value depending on what users exist in your database.

4 Retrieving splatts for a specified user

Since splatts belong to a user, we should be able to retrieve the splatts that belong to one of them. We need to add few things to our code for this.

First, add the following line to the User class in `app/models/user.rb`:

```
has_many :splatts
```

This is how we specify the other end of the one-to-many relationship.

Next, we need to add a new method to the UsersController class in `app/controllers/users_controller.rb`:

```
def splatts
  @user = User.find(params[:id])

  render json: @user.splatts
end
```

We need one more thing - a way to map a URL to a call to this method. In Rails we do this by specifying a *route* in `config/routes.rb`. Add the following line to yours, after the lines that begin with "resources":

```
get 'users/splatts/:id' => 'users#splatts'
```

This will route an HTTP GET request to `users/splatts/1` to the `splatts` method of the `users` controller. The value 1 in this example will be placed into the `params` hash at `params[:id]`.

Now you should be able to get all the splatts belonging to the user with id 1 with the commmand

```
curl -i -H "Content-type:  application/json" -X GET http://lastname.sqrawler.com:3000/users/splatts/1
```

5 Looking ahead

Using Twitter, we can follow other users and see a feed of posts belonging to all the users we follow. We will implement this in the next lab.