CS 169 Fall 2019 - Week 9 - Advanced Rails

Setup a Rails App:

```
     rails new demo_app
     cd demo_app
     rails generate scaffold user username:string admin:boolean
     bundle exec rake db:migrate
```

Custom Validation

Start from modifying the `User` model as follows:

```
class User < ActiveRecord::Base
     validates :username, :presence => true
     validate :username_format

     def username_format
     end
end
```

First try following commands in `rails console` and see output:

```
     user = User.new
     user.valid?
     user.errors
     user.save
     user.save!
```

Pair Programming: Implement `username_format.` Add messages to the errors collection if

- an username doesn't start with a letter
- an username is shorter than 10 characters

  Hint1: you can directly access `errors`
  Hint2: `errors` has method `add`

Associations Basics

Now we want to create Todo item. Each Todo item belongs to a user. A user can have many todo items. Use the following command to associate Todo with User.

Rails App Preparation:

```
     rails generate scaffold todo description:string
user:references
     bundle exec rake db:migrate
```

Now in `rails console`, type the following ruby code to check the association:

```
     user = User.create(username: "hezheng", admin: false)
     td = Todo.create(description: "todo item 1")
     td.user = user
     td.save
```

Discussion: What would happen if we type `user.todos` inside rails console? Why?

Pair Programming: Fix this with one line of code

After completing this task, you should be able to do the following things in rails console:

```
     User.first.todos.create(description: "test")
```

`User.first.todos` should be a collection now.

When you destroy the user, the related todo items will also be destroyed:

```
     User.first.destroy
```

Life Without Associations

We want to model a one to many relationship between `User` and `Picture`; i.e. a user can own many pictures, and a picture has one owner. To do this, we added a foreign key for users onto pictures (so pictures have a field `user_id`).

Pair Programming: How would we implement the following actions WITHOUT having `belongs_to` and `has_many` on our models.

a. Create a new Picture that belongs to `@user`.

b. Delete `@user` and all of of the pictures associated with that user.

Now say we added `belongs_to` and `has_many` to their respective models. How would implement the two actions above?