Lab 4.1: Validating Data in Models
IN705 Databases Three

Introduction

In our database application, our model classes have two important responsibilities. First, they read data from and save data to our data store as appropriate. Second, they *validate* data entered to be sure that we do not save any invalid data in our data store.

It is extremely important that we never save invalid information in our databases.[1]

Through ActiveRecord, Ruby on Rails provides a set of methods that facilitate data validation in our models. You can read about these at `http://guides.rubyonrails.org/active_record_validations.html`.

1 Validating our users

We store the following items of data for each of our users:

- name
- email address
- password
- blurb

Let's enforce the following constraints on our model:

- name: must not be empty
- email address: must be unique, i.e., no two users have the same email address
- password: must be at least 8 characters long

We will do this by using *Active Record Validations* in our User class.

To require a value in the name field, add the following line to the User class;

```
validates :name, presence: true
```

To enforce uniqueness for email addresses, add the following:

```
validates :email, uniqueness: true
```

The problem with this is that you can get around it by varying the case of the supplied email address. To prevent that, do the following:

```
validates :email, uniqueness: { case_sensitive: false}
```

[1] Although we will see later that we sometime relax this requirement in certain deliberate and carefully managed ways.

To enforce our length requirement on passwords, add the following:

```
validates :password, length: {minimum: 8}
```

2 Explicitly controlling validations

Suppose we want to enforce a password policy that includes, in addition to a minimum length, some additional requirements like the mixed-case or the use of a special character. We need to write our own validation method. There are a few ways to do this, but one example is the following:

```
validates :password, length: {minimum: 8}, if: :strong?

def strong?
   password =~ /.*\d+.*/ && \
   password =~ /.*[a-z]+.*/ && \
   password =~ /.*[A-Z].*/
end
```

With this code, the password validates only if the length is at least 8 characters and if the `strong?` method returns `true`.

3 Conclusion

Using the methods here or others that you find on the linked web page, perform appropriate validations on your User and Splatt models.