Lesson 11:

Validating form input

Objectives

– Implement declarative validation
– Customizing data validation
– Introducing regular expressions
– Implement custom validation method

What is input validation?

Applications are as good as their data.

Validation means to verify input is of the correct data type before storing it, notifying the user of any errors.

How are validation rules defined in Rails?

Rails defines validation rules in the models for an application. By validating in the model, the rules apply regardless how invoked.

Validation helpers enable easy implementation of validation rules using the general syntax:

```
validates [field_name], [validation_type], [options]
```

For example:

```
validates :tweet, length: { maximum: 140 }
validates :email, uniqueness: true, presence: true
```

How are validation rules defined in Rails?

Validation may also be implemented using more readable helper names, such as:

```
validates_presence_of [field name], [options]
validates_length_of [field name], [options]
validates_format_of [field name], [options]
validates_inclusion_of [field name], [options]
validates_uniqueness_of [field name], [options]
validates_numericality_of [field name], [options]
```

Rails also supports custom validation using either regular expressions or custom methods

How do you implement a declarative validation rule?

Assign and configure validation rules in the application model. For example:

```
validates_presence_of :password

validates_length_of :password, in: 8..14,
    message: “8 to 14 characters!”

validates_numericality_of :release_year,
    greater_than: 1910,
    less_than_or_equal_to: Time.now.year,
    only_integer: true,
    allow_nil: true
```

How do you specify when validation occurs?

By default, validation occurs on both creation and update of a model (i.e. the "save" of a model). The :on parameter enables specifying validation which occurs on one but not the other:

```
on: :create
on: :update
```

Example: it will be possible to update email with a duplicated value, but not on create

```
validates :email, uniqueness: true, on: :create
```

Exercise: Implementing validation helpers

What is a regular expression?

Regular expressions are a mini-language built into Ruby and many other programming languages.

They are a ***pattern matching*** syntax.

Regex syntax is involved enough to be beyond the scope of this course, but useful resources include:

http://www.oreillynet.com/pub/a/ruby/excerpts/ruby-learning-rails/ruby-guide-regular-expressions.html
http://rubylearning.com/satishtalim/ruby_regular_expressions.html

What is a regular expression?

Regex may be used in form validation, using the :with parameter

Requiring a character in the range from 0 through 9

```
validates_format_of :password, with: /[0-9]/
```

Requiring a valid email address, and providing message

```
validates_format_of :email,
 with: /\A[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]+\z/i,
 message: “may not contain a valid email address”
```

How do you conditionally validate depending on field values?

Validation helpers support an :if parameter. A custom method, which returns a boolean (true or false), can be implemented to determine if a validation rule should apply:

```
validates_presence_of :synopsis, if: :movie_is_g_or_pg?,

def movie_is_g_or_pg?
    if self.rating == "G" or self.rating == "PG"
        true
    else
        false
    end
end
```

Custom validations

```
# Note the singular ‘validate’
validate :check_reserved_usernames

def check_reserved_usernames
    reserved = [“admin”, “rik”]
    if reserved.include?(self.username.downcase.strip)
        self.errors.add(:username, 'is not allowed')
    end
end
```

Exercise:
Conditionally validating fields

What is Bundler?

Bundler is a tool that maintains a consistent environment for a developer's code across all machines.

When a developer's code is run on different machines, the libraries the developer originally built their code on may change. This breaks the developer's application.

Bundler ensures that the libraries will remain the same, taping everything together so that the developer's application runs smoothly on every machine where it is deployed.

Gemfile

```
# Keep gem on the very latest version
gem ‘rails’

# Keep gem always on 3.2.8
gem ‘rails’, ‘3.2.8’

# Keep gem on 3.2.8 or later up to but earlier than 3.3.0
gem ‘rails’, ‘~> 3.2.8’

# Keep gem on 3.2.8 or later up to but earlier than 4.0
gem ‘rails’, ‘~> 3.2’
```

Gemfile

Make sure all dependencies in your Gemfile are available to your application.

```
bundle install
```

Update all your gems to the latest version

```
bundle update
```

Lab