Chapter 13 Notes:

13.1 A Micropost model

- Modeled as a resource backed by an Active Record model, similar to users.
  - In the micropost model use **belongs_to: user** to declare the relationship to user
  - The user model then **has_many :microposts** to declare it's relationship to microposts
- Require UserID and Content
  - **t.text :content**
  - **t.references :user, foreign_key: true**
- Also create an index on user_id and created_at
  - **add_index :microposts, [:user_id, :created_at]**
  - this creates a multiple key index–using both keys at the same time.

Method	Purpose
`micropost.user`	Returns the User object associated with the micropost
`user.microposts`	Returns a collection of the user's microposts
`user.microposts.create(arg)`	Creates a micropost associated with `user`
`user.microposts.create!(arg)`	Creates a micropost associated with `user` (exception on failure)
`user.microposts.build(arg)`	Returns a new Micropost object associated with `user`
`user.microposts.find_by(id: 1)`	Finds the micropost with id `1` and `user_id` equal to `user.id`

NOTE: Use bootstrap to add in relative times in fixtures: **created_at: <%= 10.minutes.ago %>**

Using -> (stabby lambda) to call an anonymous function in microposts model to sort in descending order

```
default_scope -> { order(created_at: :desc) }
```

Adding this to the model allows a dependent micropost to be destroyed when the user itself is destroyed

```
has_many :microposts, dependent: :destroy
```

13.2 Showing microposts

- Uses ordered lists and pagination – be sure to check if any posts exist!
- Testing – add 30 microposts for a user

```
<% 30.times do |n| %>
micropost_<%= n %>:
```

```
    content: <%= Faker::Lorem.sentence(5) %>
    created_at: <%= 42.days.ago %>
    user: michael
  <% end %>
```

13.3 Manipulating microposts

Since the **logged_in_user** method will used in both Users and Microposts this can be moved to the application controller

HTML to render different code depending on if a user is logged in

```
<% if logged_in? %>
...
<% else %>
```

Calls shared/_error_messges partial and passes it f.object

```
<%= render 'shared/error_messages', object: f.object %>
```

```
<% if object.errors.any? %>
  <div id="error_explanation">
    <div class="alert alert-danger">
    The form contains <%= pluralize(object.errors.count, "error") %>.
    </div>
    <ul>
  <% object.errors.full_messages.each do |msg| %>
      <li><%= msg %></li>
    <% end %>
    </ul>
  </div>
<% end %>
```

The **where** method can be used to perform Active Record selections.
Using escaped query using '?' prevents SQL injection.

```
Micropost.where("user_id = ?", id)
```

Redirect back to the previous URL

```
request.referrer || root_url
```

13.4 Micropost images: Upload and resize images using CarrierWave

Other Notes:

- It is possible to pass variables to Rails partials.
- We can enforce secure operations by always creating and destroying dependent objects through their association.
- Fixtures support the creation of associations.