Assignment: Queries

Requirements

- Improve the user experience of your Ruby on Rails application when displaying a large number of records. For at least one resource in your application, add functionality to filter, sort, and paginate records listed on the index page for the resource.

- Use Active Record scopes to implement the query logic necessary to filter and sort records, and implement logic in your controller to apply the scopes based on specific request parameters. Implement the UI components (links and/or forms) necessary to allow users to filter and sort records on the index page. Implement at least two different filters, and at least two different sort orders.

- Use the Kaminari gem to implement the logic and UI components necessary to paginate records on the index page.

Steps

1. Within the model for your resource, add scopes to filter records using attributes on the model. The example below queries all Moon records and returns those with an atmosphere.

   ```ruby
   scope :atmosphere, -> { where(:atmosphere => true) }
   ```

2. Your scopes may also accept arguments. For example, the scope below accepts an argument `planet_id` and returns all Moon records associated to a Planet with that ID.

   ```ruby
   scope :planet, -> planet_id { where(:planet_id => planet_id) }
   ```

3. Here is an example demonstrating how to perform a basic wildcard search using a scope that accepts an argument.

   ```ruby
   scope :search, -> search { where("name LIKE ?", "%#{search}%") }
   ```

4. You may also create scopes that return records sorted in a particular order. The example below returns all Moon records sorted alphabetically by name.

   ```ruby
   scope :order_by_name, -> { order("name") }
   ```

5. If you’d like to filter or order based on the attributes of an associated model, you must first include the association. For example, the scope below returns all Moon records sorted by the position of the associated Planet records.

   ```ruby
   scope :order_by_planet_position, -> { includes(:planet).order("planets.position") }
   ```

6. Now test your new scopes. From within the Rails console, enter something like the following, and you’ll get an array of matching records, or an empty array if no records match. If the scope specifies an order, the array will be sorted appropriately.

   ```ruby
   irb> Moon.atmosphere
   irb> Moon.planet(4)
   irb> Moon.search("mars")
   irb> Moon.order_by_name
   irb> Moon.order_by_planet_position
   ```

7. Then, within the controller for your resource, apply scopes based on specific request parameters. If you are using CanCan, use code similar to the following within your `index` action.

   ```ruby
   def index
     if params[:planet].present?
       @moons = @moons.planet(params[:planet])
     end
   end
   ```
8. Finally, add some links to your index.html.erb view template to allow your users to filter and sort the records. Simply use link_to and specify URL parameters that match your controller logic. See below for a few examples.

```ruby
if params[:order] == "name"
  @moons = @moons.order_by_name
end
end
```

8. Finally, add some links to your index.html.erb view template to allow your users to filter and sort the records. Simply use link_to and specify URL parameters that match your controller logic. See below for a few examples.

```html
<p>
  Filter by:
  <%= link_to "Atmosphere", :atmosphere => true %>
  <%= Planet.all.each do |planet| %>
  <%= link_to planet.name, :planet => planet %>
  <% end %>
</p>

<p>
  Sort by:
  <%= link_to "Name", :order => "name" %>
  <%= link_to "Planet", :order => "planet_position" %>
</p>
```

9. In order to paginate records on the index page for your resource, use the Kaminari gem. Add gem 'kaminari' to your Gemfile and run bundle within Terminal to install it.

10. Kaminari implements pagination by providing scopes just like the ones you created. Use code similar to the following to apply the page scope within the controller for your resource.

```ruby
def index
  @moons = @moons.page(params[:page])
end
```

11. Kaminari also provides a convenient helper method to render pagination links within your views. Add a line similar to the following near the end of your index.html.erb view template.

```erb
<%= paginate @moons %>
```

12. That’s it! Since scopes play nicely together, the pagination links will automatically work together with your filtering and sorting links.

**Resources**

- Use the example from class as a reference when completing the steps above.
- [Ruby on Rails Guides: Active Record Query Interface](https://guides.rubyonrails.org/active_record_query_language.html)
- [Kaminari](https://github.com/i11y/kaminari)

**Submission**

- Show your completed assignment to the instructor during class or office hours to receive credit.