Ruby on Rails Exercise

# Tasty Recipes

1. Create a new Rails project called recipe\_app.
2. Generate a Recipe model, controller and views scaffolding with the following attributes:
   1. title (string)
   2. instructions (text)
3. Apply Migration
4. Change the root route to go to the recipes#index action.
5. Generate an Ingredient model, controller and views with the following attributes:
   1. name (string)
   2. quantity (integer)
   3. recipe (references)
6. Add a has\_many relationship between Recipe and Ingredient.
7. Update the Recipe show view to display a list of ingredients for that recipe.
8. Update the Recipe form to allow users to add ingredients to the recipe when creating or editing it.
   1. Add before Submit button on \_form for Recipe (you can copy past from the code block below)

<%= form.label :title%>

  <%= form.text\_field :title %>

  <%= form.label :instructions %>

  <%= form.text\_field :instructions %>

  <div id="ingredients">

    <%= form.fields\_for :ingredients do |ingredient\_fields| %>

      <div class="ingredient">

        <div class="field">

          <%= ingredient\_fields.label :name %>

          <%= ingredient\_fields.text\_field :name %>

        </div>

        <div class="field">

          <%= ingredient\_fields.label :quantity %>

          <%= ingredient\_fields.number\_field :quantity %>

        </div>

        <%= link\_to "Remove", "#", class: "remove\_fields" %>

      </div>

    <% end %>

  </div>

<%= link\_to "Add Ingredient", "#", class: "add\_fields", data: { target: "#ingredients" } %>

* 1. Add this at the end of \_form for JavaScript magic

<script>

  document.addEventListener("click", function(event) {

    var target = event.target;

    if (target.classList.contains("add\_fields")) {

      event.preventDefault();

      var targetSelector = “#ingredients”

      var targetNode = document.querySelector(targetSelector);

      var firstField = targetNode.querySelector(".ingredient:first-of-type");

      var newField = firstField.cloneNode(true);

      var inputFields = newField.querySelectorAll("input");

      inputFields.forEach(function(field) {

        field.value = "";

        field.name = field.name.replace(/\d+/, function(match) {

          return parseInt(match) + 1;

        });

      });

      targetNode.appendChild(newField);

    } else if (target.classList.contains("remove\_fields")) {

      event.preventDefault();

      var field = target.closest(".ingredient");

      field.parentNode.removeChild(field);

    }

  });

</script>

1. Change Recipe model to allow for nested attributes

class Recipe < ApplicationRecord

    has\_many :ingredients, dependent: :destroy

    accepts\_nested\_attributes\_for :ingredients, allow\_destroy: true, reject\_if: :all\_blank

end

1. Change Recipe controller to permit ingredient attributes

# Only allow a list of trusted parameters through.

    def recipe\_params

      params.fetch(:recipe, {}).permit(:title, :instructions, ingredients\_attributes: [:id, :name, :quantity, :\_destroy])

    end

1. Change Recipe Controller to instantiate ingredients as part of @recipe

# GET /recipes/new

  def new

    @recipe = Recipe.new

    @recipe.ingredients.build

  end