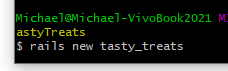
Tasty Treats

Solution Guide

# Step 1: Create a new Rails project called recipe\_app.

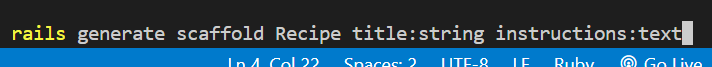
To create a new Rails project, run the following command in your terminal:

rails new recipe\_app



# Step 2: Generate a Recipe model, controller and views with the following attributes:

To generate a Recipe model with the specified attributes, run the following command inside the project folder:



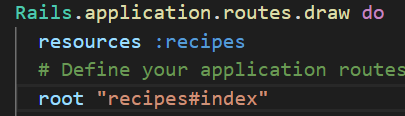
# Step 3: Apply Migration

rails db:migrate in terminal



# Step 4: Change the root route to go to the recipes#index action.

To change the root route to go to the recipes#index action, update the config/routes.rb file as follows:



# Step 5: Generate an Ingredient model, controller and views with the following attributes:

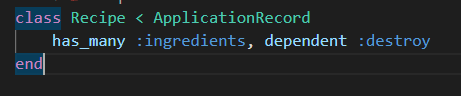
To generate an Ingredient model with the specified attributes, run the following command:

rails generate scaffold Ingredient name:string quantity:integer

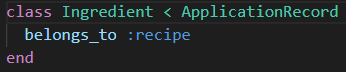
recipe:references

# Step 6: Add a has\_many relationship between Recipe and Ingredient.

To add a has\_many relationship between Recipe and Ingredient, update the app/models/recipe.rb file as follows:



and update the app/models/ingredient.rb file as follows:



# Step 7: Update the Recipe show view to display a list of ingredients for that recipe.

To update the Recipe show view to display a list of ingredients for that recipe, update the app/views/recipes/show.html.erb file as follows:

<h2>Ingredients</h2>

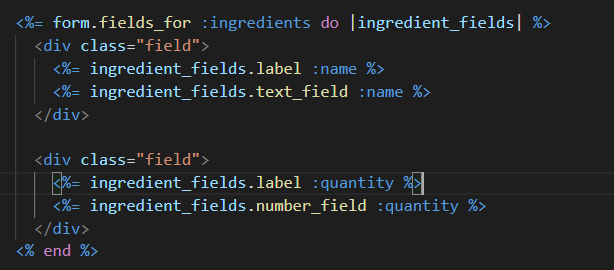
<ul>

  <% @recipe.ingredients.each do |ingredient| %>

    <li><%= ingredient.name %> (<%= ingredient.quantity %>)</li>

  <% end %>

</ul>

Step 8: Update Recipe Form

Add this to the end of the form for javascript

<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>

# Step 9: 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

# Step 10: Change Recipe controller to permit ingredient attributes

Finally, update the app/controllers/recipes\_controller.rb file to permit the ingredients\_attributes parameter when creating and updating recipes:

# Only allow a list of trusted parameters through.

    def recipe\_params

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

    end

# Step 11: Change Recipe controller to instantiate ingredients as part of recipe

# GET /recipes/new

  def new

    @recipe = Recipe.new

    @recipe.ingredients.build

  end