## Part 1 - A New Rails Application

1. rails new simple\_store
2. git commit -m "Your commit message."
3. rails g model Product title:string description:text price:decimal stock\_quantity:integer
4. git commit -m "Your commit message."
5. rails db:migrate

rails server

1. product.rb add validation:

validates :title, presence:true

validates :price, presence:true

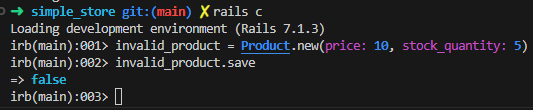
validates :stock\_quantity, presence:true

Test with rails console:

rails console or rails c

invalid\_product = Product.new(price: 10, stock\_quantity: 5)

invalid\_product.save



Test with db/seeds.rb, add the following code, then run rails db:seed

i invalid\_product = Product.new(price: 10, stock\_quantity: 5)

if invalid\_product.save

  puts "Product saved successfully."

else

  puts "Failed to save product: #{invalid\_product.errors.full\_messages.join(", ")}"

end



1. git commit -m "Your commit message."
2. gem “faker” -- add into Gemfile

bundle install

1. seeds.rb:

require "faker"

# set the random seed to get the same data every time

Faker::Config.random = Random.new(42)

676.times do

  Product.create(

    title: Faker::Commerce.product\_name,

    price: Faker::Commerce.price(range: 0..100.0, as\_string: true),

    stock\_quantity: Faker::Number.between(from: 0, to: 100)

  )

end

puts "676 products created successfully."

1. rails db:seed
2. rails generate controller Products index
3. git commit -m “message”
4. modify controller

def index

@prodcut = Product.all

end

1. 1) update app/assets/stylesheets

Add a .scss file procuts.scss

.products-grid {

  display: flex;

  flex-wrap: wrap;

  gap: 20px; // Creates space between grid items

  justify-content: center; // Center grid items horizontally

}

.product-card {

  border: 1px solid #ddd; // Adds a border around each product card

  padding: 20px;

  width: 200px; // Sets a fixed width for each product card

  text-align: center; // Center-aligns the text inside the card

  box-shadow: 0 2px 4px rgba(0,0,0,0.1); // Adds a subtle shadow around each card

}

Then add gem “scssc-rails” into Gemfile;

bundle install

rails assets:precompile

2)modify index.html.erb

<h1>Products</h1>

<div class="products-grid">

  <% @products.each do |product| %>

    <div class="product-card">

      <h2><%= product.title %></h2>

      <p>$<%= product.price %></p>

    </div>

  <% end %>

</div>

1. resources :products
2. git commit -m “”
3. create app\views\products\show.html.erb:

<h1><%= @product.title %></h1>

<p>Price: $<%= @product.price %></p>

<p>Stock Quantity: <%= @product.stock\_quantity %></p>

<%= link\_to 'Back to Products', products\_path %>

add routes.rb

add action of show into controller:

def show

    @product = Product.find(params[:id])

  end

run rail server , visit /products/:id

1. index.html.erb :  
   <td><!-- Example of additional actions you might include -->

          <%= link\_to 'View', product\_path(product), class: 'btn btn-info' %>

        </td>

1. git commit -m “”

## Patr 2 - Loading data from CSV

1. rails g model Category name:string
2. rails c

Product.destroy\_all

1. rails g migration add\_category\_to\_products category:references

rails db:migrate

1. Category.rb:

class Category < ApplicationRecord

has\_many :products

end

product.rb:  
belongs\_to :category

git commit -m “”

1. Create products.csv in db filling in data
2. require "csv" #in seeds.rb
3. check the appendix in instructions which about how to pull data from csv

require "csv"

csv\_file = Rails.root.join("db/products.csv") # path to the csv file containing the data

csv\_data = File.read(csv\_file)# read the file into a string using the File.read method

products = CSV.parse(csv\_data, headers: true, encoding: "ISO-8859-1") # parse the CSV data into an array of rows using the CSV.parse method,headers: true option to tell the parser to treat the first row as headers

new\_products = []

current\_time = Time.current

products.each do |product|

  # create categories

  category = Category.find\_or\_create\_by!(name: product["category"]) # find a category with the name from the CSV file in the database so that we don't create duplicate categories

  # create products

  new\_product = Product.new(

    title: product["name"],

    description: product["description"],

    price: product["price"],

    stock\_quantity: product["stock quantity"],

    category\_id: category.id,

    created\_at: current\_time,

    updated\_at: current\_time

  )# create a new product object with the data from the CSV file

  # if new\_product.save

  #   puts "Product created successfully."

  # else

  #   puts "Failed to create product: #{new\_product.errors.full\_messages.join(", ")}"

  # end

  new\_products << new\_product

end

Product.insert\_all!(new\_products.map(&:attributes))# use the insert\_all! method to insert all the new products into the database in a single query

puts "Products created successfully."# print a message to the console to indicate that the products were created successfully

1. In product controller:

@products = Product.includes(:category).all

 @product = Product.includes(:category).find(params[:id])