МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Кафедра програмних засобів

Звіт

з лабораторної роботи №6

з дисципліни: «Крос-платформне програмування»

за темою: «Створення асинхронного веб-інтерфейсу за допомогою Rails та Javascript. Selenium IDE»

Виконав:

ст. гр. КНТ-117 М. І. Басанець

Прийняв:

старший викладач Є. М. Федорченко

2020

1 МЕТА РОБОТИ

Поглиблення теоретичних відомостей про принципи роботи асинхронного веб-інтерфейсу.

Отримання практичних навичок створення веб-додатки з використанням засобів Ruby on Rails та технології AJAX.

2 зАВДАННЯ ДО РОБОТИ

1. Використайте веб-додаток на ruby on rails, розробивши калькулятор для підрахунку вартості якоїсь необхідної речі (яблуко, кофта, м’яч і т.д.).
2. Виконайте асинхронний запит за допомогою Ruby on Rails.
3. Виконайте асинхронний запит за допомогою Javascript.
4. Відлагодьте веб-додаток.
5. За допомогою Selenium запишіть тест вашого веб-додатка.

3 Результати виконання завдання

Для виконання лабораторної роботи було створено дуже спрощений аналог інтернет магазину лише з однією моделлю та нагенерованими сторінками.

Результати доданих продуктів зображено на рисунку 3.1

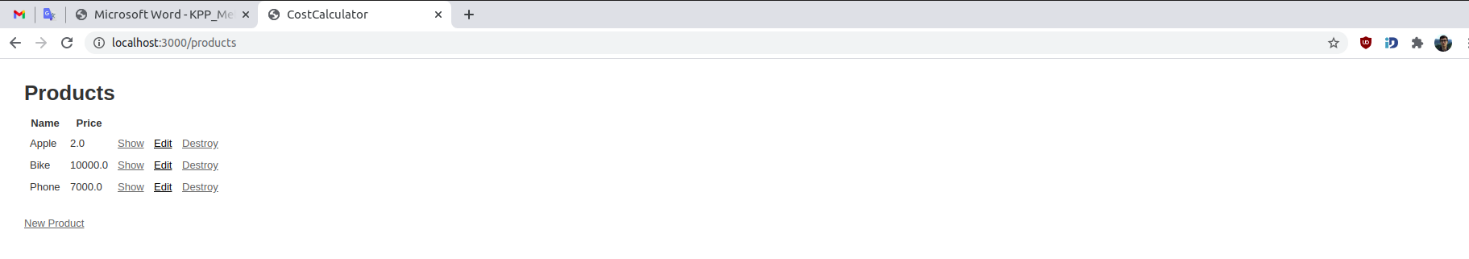


Рисунок 3.1 – Результати доданих продуктів

Код програми:

Файл products\_controller.rb:

class ProductsController < ApplicationController

before\_action :set\_product, only: [:show, :edit, :update, :destroy]

# GET /products

# GET /products.json

def index

@products = Product.all

end

# GET /products/1

# GET /products/1.json

def show

end

# GET /products/new

def new

@product = Product.new

end

# GET /products/1/edit

def edit

end

# POST /products

# POST /products.json

def create

@product = Product.new(product\_params)

respond\_to do |format|

if @product.save

format.html { redirect\_to @product, notice: 'Product was successfully created.' }

format.json { render :show, status: :created, location: @product }

else

format.html { render :new }

format.json { render json: @product.errors, status: :unprocessable\_entity }

end

end

end

# PATCH/PUT /products/1

# PATCH/PUT /products/1.json

def update

respond\_to do |format|

if @product.update(product\_params)

format.html { redirect\_to @product, notice: 'Product was successfully updated.' }

format.json { render :show, status: :ok, location: @product }

else

format.html { render :edit }

format.json { render json: @product.errors, status: :unprocessable\_entity }

end

end

end

# DELETE /products/1

# DELETE /products/1.json

def destroy

@product.destroy

respond\_to do |format|

format.html { redirect\_to products\_url, notice: 'Product was successfully destroyed.' }

format.json { head :no\_content }

end

end

def calculate\_price

product\_id = params[:product][:id]

product\_quantity = params[:quantity]

@product = Product.find(product\_id)

@calculated\_price = @product.price \* product\_quantity.to\_i

respond\_to do |format|

format.html { redirect\_to @product, notice: "Total price for " + product\_quantity + ' ' + @product.name + ' ' + 'is $' + @calculated\_price.to\_i.to\_s }

end

end

private

# Use callbacks to share common setup or constraints between actions.

def set\_product

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

end

# Only allow a list of trusted parameters through.

def product\_params

params.require(:product).permit(:name, :price)

end

end

Файл моделі product.rb:

class Product < ApplicationRecord

end

Файл представлення з вбудованим AJAX:

<p id="notice"><%= notice %></p>

<p>

<strong>Name:</strong>

<%= @product.name %>

</p>

<p>

<strong>Price:</strong>

<%= @product.price %>

</p>

<%= form\_for(@product, :url => calculate\_price\_path(:product\_quantity => :quantity), :method => :post, remote: true) do |f| %>

<p>

<%= f.hidden\_field :id, :readonly => true %>

<strong>Quantity to inquire for: </strong>

<%= text\_field\_tag :quantity, "0" %>

</p>

<%= submit\_tag "Calculate price" %>

<% end %>

<%= link\_to 'Edit', edit\_product\_path(@product) %> |

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

Файл products\_controller\_test.rb з тестами для контроллеру:

require 'test\_helper'

class ProductsControllerTest < ActionDispatch::IntegrationTest

setup do

@product = products(:one)

end

test "should get index" do

get products\_url

assert\_response :success

end

test "should get new" do

get new\_product\_url

assert\_response :success

end

test "should create product" do

assert\_difference('Product.count') do

post products\_url, params: { product: { name: @product.name, price: @product.price } }

end

assert\_redirected\_to product\_url(Product.last)

end

test "should show product" do

get product\_url(@product)

assert\_response :success

end

test "should get edit" do

get edit\_product\_url(@product)

assert\_response :success

end

test "should update product" do

patch product\_url(@product), params: { product: { name: @product.name, price: @product.price } }

assert\_redirected\_to product\_url(@product)

end

test "should destroy product" do

assert\_difference('Product.count', -1) do

delete product\_url(@product)

end

assert\_redirected\_to products\_url

end

end

Файл product\_test.rb з тестами для моделі:

require 'test\_helper'

class ProductTest < ActiveSupport::TestCase

# test "the truth" do

# assert true

# end

end

На рисунку 3.2 зображено форму до асинхронного запиту (без результату)

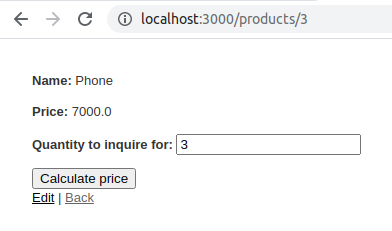


Рисунок 3.2 – Результат роботи до асинхронного запиту

На рисунку 3.3 зображено форму після асинхронного запиту (з результатом)

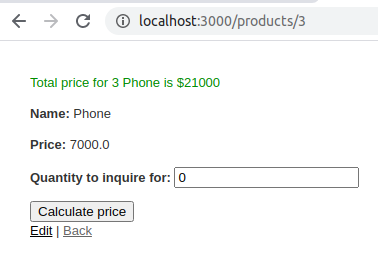


Рисунок 3.3 – Результат роботи після асинхронного запиту

На рисунку 3.4 зображено результати тестування проетку з використанням Selenium

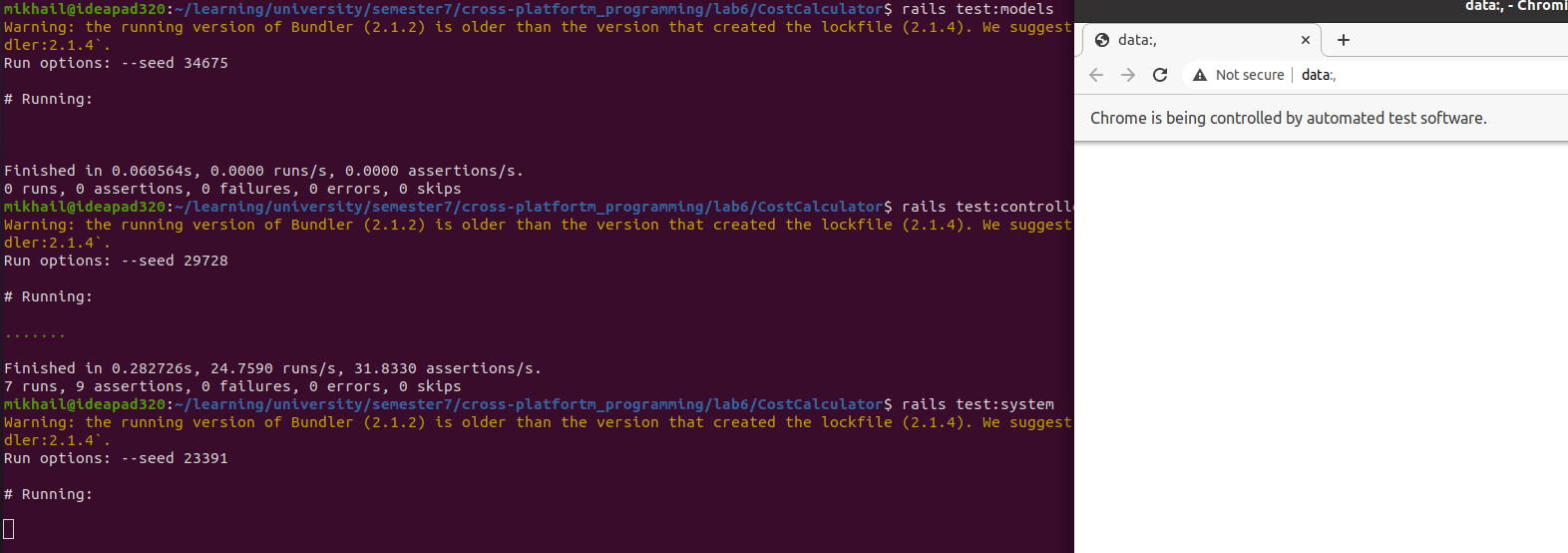


Рисунок 3.4 – Результати виконання тестування

4 висновки

При виконанні даної лабораторної роботи на практиці засвоїв основні принципи та засади роботи з ruby, javascript, selenium та тестуванням веб-додатків.

Відповіді на контрольні запитання:

1. Що таке Selenium IDE?

Selenium IDE - інструмент для автоматизації роботи в web-браузері. Selenium надає застосунок запису/відтворення, що дозволяє створювати тести вебзастосунків без вивчення мов програмування.

2. Що таке JSON?

JSON - це текстовий формат обміну даними між комп'ютерами. Виступає як аналог до формату XML.

3. Що таке Ajax-helpers?

AJAX-helpers - допоміжні методи для view, написані на Ruby, що допомагають генерувати HTML код. За допомогою цих елементів можна легко використовувати AJAX у Ruby. Один з прикладів AJAX-helpers - form\_for, що генерує форму для заповнення.

4. Навіщо потрібен метод respond\_to?

Метод respond\_to використовується для відправки повідомлень від контролера до відображення у заданому форматі.

5. Що таке асинхронна обробка?

Асинхронна обробка - форма обробки вводу/виводу, яка дозволяє виконувати інші процеси ще до того як закінчиться передавання даних