Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.01 **Информатика и вычислительная техника**

О Т Ч Е Т

по лабораторной работе № 1 0

Название: Формирование и отображение XML в HTML средствами сервера и клиента.

Дисциплина: Языки интернет программирования

Студент	ИУ6-53Б		Т.Р. Сапарбаев
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			Р.С. Самарев
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2020

Задание

Модифицировать код ЛР 8 таким образом, чтобы по запросу с указанными параметрами выдавался результат в формате XML (средствами стандартной сериализации ActiveSupport).

- Проверить формирование XML и сохранить в файл для отладки XSLT и второго приложения.
- Написать функциональный тест, проверяющий формат выдаваемых данных при запросе RSS.

Разработать XSLT-программу преобразования полученной XML в HTML.
Добавить в проверяемый XML-файл строку привязки к преобразованию <?xml-stylesheet type="text/xsl" href="some_transformer.xslt"?>. Проверить корректность отображения браузером результата преобразования.

Проверить на автономной Ruby-программе корректность преобразования, используя следующий фрагмент кода:

```
require 'nokogiri'
doc = Nokogiri::XML(File.read('some_file.xml'))
xslt = Nokogiri::XSLT(File.read('some_transformer.xslt'))
puts xslt.transform(doc)
```

Разработать второе приложение, являющееся посредником между клиентом и первым приложением, задачей которого является преобразование XML в HTML или передача в неизменном виде браузеру для отображения браузером. Приложение должно запускаться с указанием номера порта TCP, отличным от номера порта первого приложения (например rails server -p 3001)!

- Подготовить каркас приложения, а также форму формирования запроса, форму отображения результата и соответствующие действия контролера.
- Добавить в контроллер преобразование XML в HTML с помощью ранее разработанного XSLT-файла.
- Подключить запрос XML с первого приложения и проверить работу приложений в связке.
- Написать функциональный тест, проверяющий что при различных входных данных результат генерируемой страницы различен.
- Доработать код контроллера и представлений данного приложения для выдачи браузеру XML-потока в неизменном виде (организовать возможность выбора формата выдачи для пользователя).
- Проверить, что браузер получает XML первого приложения в неизменном виде.
- Доработать код контроллера приложения таким образом, чтобы XML-поток первого приложения получал дополнительную строку, указывающую xsl. Модифицировать форму запроса параметров таким образом, чтобы браузер получал в ответ XML. При этом разместить XSLT-файл в директории public.
- Проверить, что браузер производит преобразование XML->HTML в соответствии с xlt.
- Реализовать функциональные тесты второго приложения. Проверить результаты, формируемые приложением, на соответствие выбранному формату выдачи.

Итоговая форма ввода параметра должна содержать кнопки или селектор, позволяющие проверить два варианта преобразования:

- Серверное xml+xslt->html
- Клиентское xml+xslt->html

Контроллер основного приложения (home_controller.rb):

```ruby
# frozen_string_literal: true

# HomeController
class HomeController < ApplicationController
  before_action :check_params, only: %(output)

  def input; end

  def output
    @result = (params[:txt].to_i + 1).times.filter do |i|
      square = (i**2).to_s
      square == square.reverse
    end
    render xml: convert_to_xml(@result)
  end

  # rubocop:disable Metrics/MethodLength
  def convert_to_xml(arr)
    Nokogiri::XML::Builder.new do
      result do
        arr&.each_with_index do |elem, i|
          element do
            index i + 1
            value elem
            square elem * elem
          end
        end
      end
    end
  end
  # rubocop:enable Metrics/MethodLength

  protected

  def check_params
    render :output unless Integer(params[:txt], exception: false)
  end
end
```

Тесты основного приложения:

```ruby
require './test/test_helper'

class HomeControllerTest < ActionDispatch::IntegrationTest
  test "should get input" do
    get input_url
```

```
    assert_response :success
  end

  test "should get output" do
    get output_url, :params => { :txt => '1' }
    assert_response :success
  end

  test 'rss request' do
    get output_url, :params => { :txt => '10', :format => :rss }
    assert response.body.include?('<?xml version="')
  end

  test 'different requests should return different responses' do
    get output_url, :params => { :txt => '10' }
    res1 = response.body
    get output_url, :params => { :txt => '100' }
    res2 = response.body
    assert_not_equal res1, res2
  end

end
```

Контроллер proxy сервера (home_controller.rb):

```
# frozen_string_literal: true

require 'nokogiri'
require 'open-uri'

# Home Controller Class
class HomeController < ApplicationController
  before_action :check_params, only: %(output)

  XSLT_FILE = '/xslt_for_response.xslt'
  MAIN_SERV = 'http://localhost:3000'

  def input; end

  # rubocop:disable Security/Open
  def output
    response = Nokogiri::XML(URI.open("#{MAIN_SERV}/home/out-
put?txt=#{params[:txt]}"))

    handle = params[:process]
    case handle
    when 'Сервер'
      render inline: transform_xml(response, "#{Rails.root}/public#{XSLT_FILE}")
    when 'Клиент'
```

```
      render xml: add_instr(response, XSLT_FILE)
    when 'Не обрабатывать'
      render xml: response
    end
  end
  # rubocop:enable Security/Open

  protected

  def check_params
    render :output unless Integer(params[:txt], exception: false)
  end

  def add_instr(doc, _href_xslt)
    ins = Nokogiri::XML::ProcessingInstruction
          .new(doc, 'xml-stylesheet', "type=\"text/xsl\" href=\"#{XSLT_FILE}\"")
    doc.root.add_previous_sibling ins
    doc
  end

  def transform_xml(doc, href_xslt)
    xslt = Nokogiri::XSLT(File.read(href_xslt))
    xslt.transform(doc)
  end
end
```

Тест для проверки формата ответа сервера:

```
require 'test_helper'

class ComparisonTest < ActionDispatch::IntegrationTest
  test 'check server processing request' do
    get output_url, :params => { :txt => '100', :process => 'Клиент' }
    content = '<?xml-stylesheet type="text/xsl" href="/xslt_for_response.xslt"?>'
    assert response.body.include?(content)
  end

  test 'check response with server processing' do
    get output_url, :params => { :txt => '100', :process => 'Сервер' }
    content = ['<html>', '<body>', '<table', '<th', '<tr']
    res = response.body
    assert content.all? { |str| res.include?(str) }
  end

  test 'check response without any handlings' do
    get output_url, :params => { :txt => '100', :process => 'Не обрабатывать' }
    content = ['<?xml version="1.0"?>', '<result>']
    not_incl = '<?xml-stylesheet type="text/xsl" href="/xslt_for_re-
sponse.xslt"?>'
```

```
    res = response.body
    assert content.all? { |str| res.include?(str) } && res.exclude?(not_incl)
  end
end
```

Пример работы приложения:

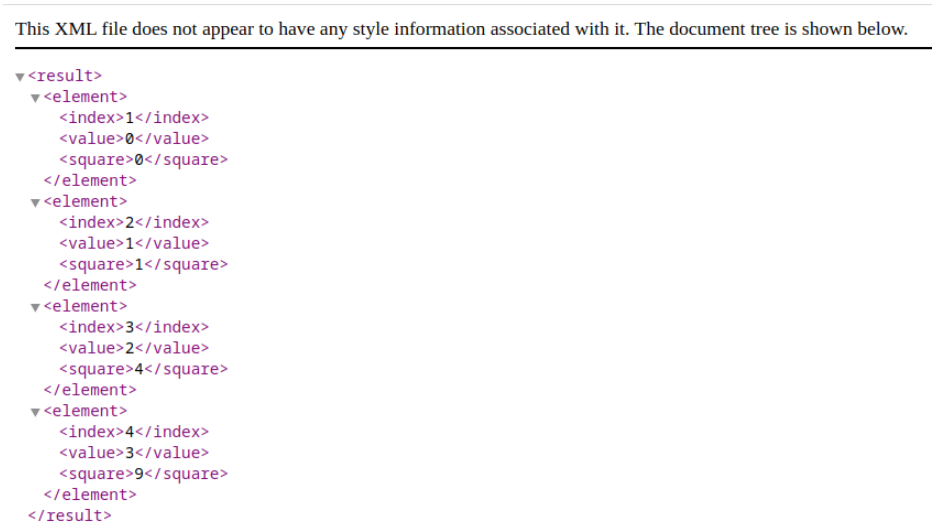

Рисунок 1 – XML ответ основного сервера

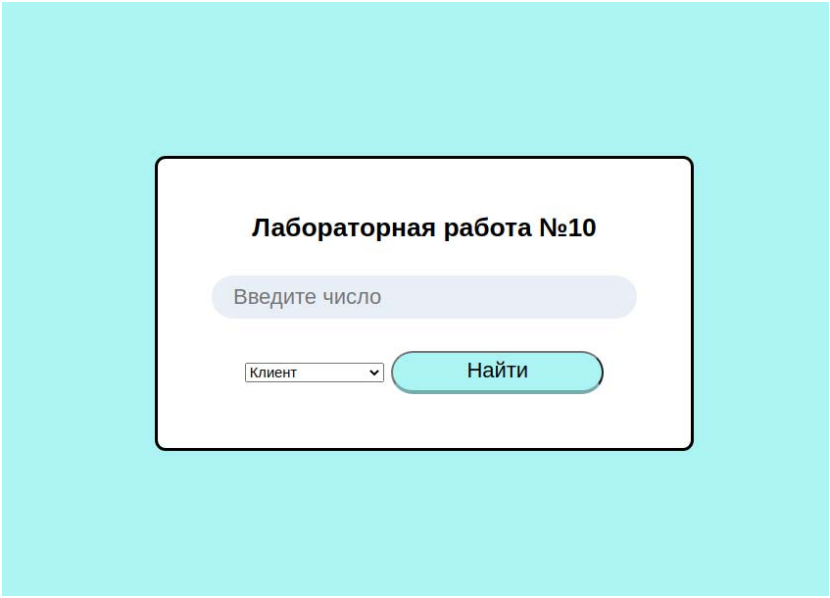

Рисунок 2 – Главная страница proxy сервера

Рисунок 3 – XML обработан на стороне клиента

Тесты основного приложения:

```
saparbi@saparbi-TM1703:~/Desktop/YAIP/LR10/main$ rails test
Running via Spring preloader in process 27507
Run options: --seed 17285

# Running:

....

Finished in 0.163717s, 24.4324 runs/s, 24.4324 assertions/s.
4 runs, 4 assertions, 0 failures, 0 errors, 0 skips
```

Рисунок 4 – Сравнение ответов на разные запросы

Тесты proxy сервера:

```
saparbi@saparbi-TM1703:~/Desktop/YAIP/LR10/main/app/controllers$ rubocop
warning: parser/current is loading parser/ruby27, which recognizes
warning: 2.7.2-compliant syntax, but you are running 2.7.1.
warning: please see https://github.com/whitequark/parser#compatibility-with-ruby-mri.
Inspecting 2 files
..

2 files inspected, no offenses detected
```

Рисунок 5 – Проверка на разные форматы выдачи

Отчеты Rubocop:

```
saparbi@saparbi-TM1703:~/Desktop/YAIP/LR10/main/app/controllers$ rubocop
warning: parser/current is loading parser/ruby27, which recognizes
warning: 2.7.2-compliant syntax, but you are running 2.7.1.
warning: please see https://github.com/whitequark/parser#compatibility-with-ruby-mri.
Inspecting 2 files
..

2 files inspected, no offenses detected
```

```
saparbi@saparbi-TM1703:~/Desktop/YAIP/LR10/main/app/controllers$ rubocop
warning: parser/current is loading parser/ruby27, which recognizes
warning: 2.7.2-compliant syntax, but you are running 2.7.1.
warning: please see https://github.com/whitequark/parser#compatibility-with-ruby-mri.
Inspecting 2 files
..

2 files inspected, no offenses detected
```

Рисунок 6 – Отчёт Rubocop

Вывод: в ходе выполнения лабораторной работы была добавлена поддержка асинхронной подгрузки результатов с помощью технологии AJAX. Приложение было протестировано с помощью Selenium WebDriver и проверено на соответствие стилю программой Rubocop.