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: Отображение XML в HTML средствами сервера и клиента.

Дисциплина: Языки интернет-программирования

Студент ИУ6-32Б _____ Джафаров Э.Э.
(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ Петрова Я.С.
(Подпись, дата) (И.О. Фамилия)

Москва, 2022

Цель работы:

Получить практические навыки формирования данных в формате XML и их визуализации с помощью клиентских и серверных средств с использованием XSLT-преобразований.

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

Решение:

1. Первое приложение

А. Контроллер proxy_controller.rb

```ruby
# frozen_string_literal: true

require 'nokogiri'
require 'open-uri'

# controller for getting input and showing answer
class ProxyController < ApplicationController
  def input; end

  def view
    @power = params[:power].to_i
    @side = params[:route]
    api_response = URI.open("http://localhost:3000/?format=xml&power=#{@power}")

    case @side
    when 'server'
      @result = xslt_transform(api_response).to_html
    when 'client'
      render xml: insert_browser_xslt(api_response).to_xml
    end
  end

  private

  def xslt_transform(data)
    doc = Nokogiri::XML(data)
    xslt = Nokogiri::XSLT(File.read("#{Rails.root}/app/helpers/server.xslt"))
    xslt.transform(doc)
  end

  def insert_browser_xslt(data)
    doc = Nokogiri::XML(data)
    xslt = Nokogiri::XML::ProcessingInstruction.new(doc, 'xml-stylesheet',
                                                    'type="text/xsl" href="/browser_render.xslt"')
    doc.root.add_previous_sibling(xslt)
    doc
  end
end
```

Б. input.html.erb

```erb
<h1>Здрасьте</h1>
<%= form_tag("/proxy/view.x", :method => "get") do %>
<%= label_tag("Введите количество цифр в числе:") %>
</br>
<%= number_field_tag(:power, nil, in: 1..6, step:1)%>
<%= select_tag(:route,options_for_select([ "server", "client" ], "server")) %> <br>
</br>
<%= submit_tag("Yosh") %>
<% end %>
```

В. show,html.erb

```erb
<h1>Страница результата</h1>
<%= render inline: @result %>
<br>
<%= link_to "Вернуться", :root %>
```

Г. routes.rb

```ruby
Rails.application.routes.draw do
  root "proxy#input"
  get 'proxy/view'
  # Define your application routes per the DSL in https://
  # Defines the root path route ("/")
  # root "articles#index"
end
```

Д. server.xslt

```xml
<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:template match="/">
      <table class="table table-hover">
        <tr>
          <th>Индекс</th>
          <th>Число</th>
        </tr>
        <xsl:for-each select="objects/object">
        <tr>
          <td>
            <xsl:value-of select="index"></xsl:value-of>
          </td>
          <td>
            <xsl:value-of select="number"></xsl:value-of>
          </td>
        </tr>
         </xsl:for-each>
        <tr>
          <th>Количество:</th>
          <th><xsl:value-of select="count(objects/object)"/></th>
        </tr>
      </table>
  </xsl:template>
</xsl:stylesheet>
```

E. browser_render.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <head>
        <title>Pr</title>
        <style>
        tr,td,th,table {
          border: 1px solid black;
        }
        </style>
      </head>
      <body>
        <h1>Страница результата</h1>
        <table class="t table-hover">

            <tr>
              <th>Индекс</th>
              <th>Число</th>
            </tr>
            <xsl:for-each select="objects/object">
            <tr>
              <td>
                <xsl:value-of select="index"></xsl:value-of>
              </td>
              <td>
                <xsl:value-of select="number"></xsl:value-of>
              </td>
            </tr>
             </xsl:for-each>
            <tr>
              <th>Количество:</th>
              <th><xsl:value-of select="count(objects/object)"/></th>
            </tr>
        </table>
        <br/>
        <a href="/">Вернуться</a>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Ж. proxy_spec.rb:

```ruby
require 'rails_helper'

RSpec.describe "Proxy", type: :request do

  describe "GET /calc" do
    it "returns http success" do
      get "/"
      expect(response).to have_http_status(:success)
    end

    it "returns http success" do
      get "/proxy/view"
      expect(response).to have_http_status(:success)
    end

    it "returns different results" do
      get '/proxy/view.x?power=3&route=server&commit=FYosh'
      response_first = @response.body
      get '/proxy/view.x?power=4&route=server&commit=Yosh'
      response_second = @response.body
      expect(response_first).not_to be eq(response_second)
    end

    # при запросе через клиента файл browser_render подгружается и браузер имеет к нему доступ
    it "returns correct result" do
      get '/proxy/view.x?power=4&route=client&commit=Yosh'
      expect(body.include?('/browser_render.xslt')).to be true
    end

    # при запросе через сервер файл browser_render не подгружается
    it "returns correct result" do
      get '/proxy/view.x?power=3&route=server&commit=Yosh'
      expect(body.include?('/browser_render.xslt')).to be false
    end
  end

end
```

3. Результат теста:

```
k/IPL-Laboratory10/pr$ rake spec
/usr/local/bin/ruby -I/usr/local/lib/ruby/gems/3.1.0/gems/rspec-c
b/ruby/gems/3.1.0/gems/rspec-support-3.12.0/lib /usr/local/lib/ru
e-3.12.0/exe/rspec --pattern spec/\*\*\{,/\*/\*\*\}/\*_spec.rb
.....

Finished in 2.87 seconds (files took 1.34 seconds to load)
5 examples, 0 failures
```

И. Результат

1 Через сервер:

Здрасьте

Введите количество цифр в числе:

4 server

Yosh

Страница результата

Индекс	Число
0	1634
1	8208
2	9474
Количество:	**3**

Вернуться

---

2 Через клиента:

Здрасьте

Введите количество цифр в числе:

4 client

Yosh

Страница результата

Индекс	Число
0	1634
1	8208
2	9474
Количество:	**3**

Вернуться

К. Проверка Rubocop:

```
k/IPL-Laboratory10/pr$ rubocop -A app/controllers/proxy_controller.rb
Inspecting 1 file
.

1 file inspected, no offenses detected
```

2. Второе приложение

A. calc_controller.rb

```ruby
# frozen_string_literal: true

# class for calculating result and returning it in xml and rss format
class CalcController < ApplicationController
  def find(number, pow)
    summa = 0
    pow.times do
      tri = number % 10
      number /= 10
      summa += tri**pow
    end
    summa
  end

  def calc
    @result = solve(params[:power].to_i)

    respond_to do |format|
      format.xml { render xml: @result.to_xml }
      format.rss { render xml: @result.to_xml }
    end
  end

  def solve(pow)
    @result = []
    ((10**(pow - 1))...(10**pow)).step(1) do |num|
      @result.push(:index => (num - 1), :number => num)
    end
    index = 0
    @result = @result.select { |num| num[:number] == find(num[:number], pow) }
    @result.each do |elem|
      elem[:index] = index
      index += 1
    end
  end

end
```

Б. routes.rb

```ruby
Rails.application.routes.draw do
  root 'calc#calc'
  # Define your application routes per the
  # Defines the root path route ("/")
  # root "articles#index"
end
```

В. Тест

```ruby
require 'rails_helper'

RSpec.describe "Calcs", type: :request do

  # при запросе xml мы получаем успех
  describe "GET /calc" do
    it "returns http success" do
      get "/?format=xml"
      expect(response).to have_http_status(:success)
    end

    #  проверяем, что при запросе rss мы получаем ответ в том же формате
    it "returns correct result" do
      get "/?format=rss&power=5"
      expect(response).to have_http_status(:success)
      expect(@response.headers['Content-Type']).to include('application/rss')
    end
  end

end
```

Г. Результат теста

```
rk/IPL-Laboratory10/website$ rake spec
/usr/local/bin/ruby -I/usr/local/lib/ruby/gems/3.1.0/gems/rspec-co
ib/ruby/gems/3.1.0/gems/rspec-support-3.12.0/lib /usr/local/lib/ru
ore-3.12.0/exe/rspec --pattern spec/\*\*\{,/\*/\*\*\}/\*_spec.rb
..

Finished in 0.2662 seconds (files took 1.26 seconds to load)
2 examples, 0 failures
```

Д. Проверка rubocop

```
rk/IPL-Laboratory10/website$ rubocop -A app/controllers/calc_controller.rb
Inspecting 1 file
.

1 file inspected, no offenses detected
```

Вывод:

Я получил практические навыки формирования данных в формате XML и их визуализации с помощью клиентских и серверных средств с использованием XSLT-преобразований.