***Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования***

|  |  |
| --- | --- |
|  | ***«Московский государственный технический университет  имени Н.Э. Баумана»***  ***(МГТУ им. Н.Э. Баумана)*** |

ФАКУЛЬТЕТ *Информатика и системы управления*

КАФЕДРА *Компьютерные системы и сети*

**Отчет**

**по лабораторной работе № 10**

**Дисциплина: Языки интернет-программирования**

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

Вариант 10

Студент гр.  **ИУ6-33Б \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Захаров К.А**

(Подпись, дата) (И.О. Фамилия)

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Самарев Р.С.**

(Подпись, дата) (И.О. Фамилия)

Москва, 2019

**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.

**Lab10-api**

Измененные файлы:

config/routes.rb  
app/controllers/xml\_controller.rb  
test/controllers/xml\_controller\_test.rb

**xml\_controller.rb**

**require 'prime'**

**class XmlController < ApplicationController**

**before\_action :parse\_params, only: :index**

**def parse\_params**

**@input = params[:input].to\_i**

**end**

**def index**

**arr\_hash = mersen\_nums(@input)**

**data = if arr\_hash == nil**

**{ message: "Неверные параметры запроса (input = '#{@input}')" }**

**else**

**arr\_hash**

**end**

**respond\_to do |format|**

**format.xml { render xml: data.to\_xml}**

**format.rss { render xml: data.to\_xml}**

**end**

**end**

**def mersen\_nums(input)**

**if (@input =~ /-/ || @input == "" || @input == '0')**

**return**

**end**

**n = input.to\_i**

**p = (Math.log(n) / Math.log(2)).to\_i**

**p += 1 if Prime.prime?(n)**

**i = 1**

**arr\_hash = []**

**arr = (1..p).to\_a.keep\_if { |num| Prime.prime?(2\*\*num - 1) }.map do |el|**

**arr\_hash.push(id: i, p: el, M: (2\*\*el - 1))**

**i += 1**

**end**

**arr\_hash**

**end**

**end**

**xml\_controller\_test.rb**

**require 'test\_helper'**

**class XmlControllerTest < ActionDispatch**

**def setup; end**

**test 'check rss format' do**

**get '/', params: { input: 127, format: :rss }**

**assert\_response :success**

**assert\_includes @response.headers['Content-Type'], 'application/rss'**

**end**

**test 'check xml format' do**

**get '/', params: { input: 127, format: :xml }**

**assert\_response :success**

**assert\_includes @response.headers['Content-Type'], 'application/xml'**

**end**

**end**

**routes.rb**

**Rails.application.routes.draw do**

**root 'xml#index'**

**# For details on the DSL available within this file, see http://guides.rubyonrails.org/routing.html**

**end**

**Lb10-proxy**

Измененные файлы:

config/routes.rb  
app/views/proxy/input.html.erb

app/views/proxy/output.html.erb  
app/controllers/proxy\_controller.rb  
test/controllers/proxy\_controller\_test.rb

**input.html.erb**

**<h1>Proxy#input</h1>**

**<p>Find me in app/views/proxy/input.html.erb</p>**

**<%= form\_tag("/proxy/output", :method => "get") do %>**

**<%= label\_tag("Input border value") %>**

**<%= number\_field\_tag(:input) %>**

**<%= submit\_tag("result") %>**

**<br/>**

**<input type="radio" class="form-check-input" name="side" value="server" id="server-side" checked>**

**Рендеринг на стороне сервера этого приложения**

**<br/>**

**<input type="radio" class="form-check-input" name="side" value="client" id="client-side">**

**Отображение сырого XML на стороне браузера**

**<br/>**

**<input type="radio" class="form-check-input" name="side" value="client-with-xslt" id="client-side-xslt">**

**Рендеринг на стороне браузера с помощью XSLT**

**<% end %>**

**output.html.erb**

**<h1>Proxy#output</h1>**

**<p>Find me in app/views/proxy/output.html.erb</p>**

**<%= render inline: @result %>**

**proxy\_controller.rb**

**require 'open-uri'**

**class ProxyController < ApplicationController**

**before\_action :parse\_params, only: :output**

**before\_action :prepare\_url, only: :output**

**def input**

**end**

**def output**

**begin**

**api\_response = open(@url)**

**rescue OpenURI::HTTPError**

**redirect\_to :index**

**return**

**end**

**if @side == 'server'**

**@result = xslt\_transform(api\_response).to\_html**

**elsif @side == 'client-with-xslt'**

**render xml: insert\_browser\_xslt(api\_response).to\_xml**

**else**

**render xml: api\_response**

**end**

**end**

**BASE\_API\_URL = 'http://localhost:3000/?format=xml'.freeze**

**XSLT\_SERVER\_TRANSFORM = "#{Rails.root}/public/server\_transform.xslt".freeze**

**XSLT\_BROWSER\_TRANSFORM = '/browser\_transform.xslt'.freeze**

**private**

**def parse\_params**

**@input = params[:input]**

**@side = params[:side]**

**end**

**def prepare\_url**

**@url = BASE\_API\_URL + "&input=#{@input}"**

**end**

**# Преобразование XSLT на сервере (код Самарева).**

**def xslt\_transform(data, transform: XSLT\_SERVER\_TRANSFORM)**

**doc = Nokogiri::XML(data)**

**xslt = Nokogiri::XSLT(File.read(transform))**

**xslt.transform(doc)**

**end**

**def insert\_browser\_xslt(data, transform: XSLT\_BROWSER\_TRANSFORM)**

**doc = Nokogiri::XML(data)**

**xslt = Nokogiri::XML::ProcessingInstruction.new(doc,**

**'xml-stylesheet',**

**'type="text/xsl" href="' + transform + '"')**

**doc.root.add\_previous\_sibling(xslt)**

**doc**

**end**

**end**

**proxy\_controller\_test.rb**

**require 'test\_helper'**

**require 'selenium-webdriver'**

**class ProxyControllerTest < ActionDispatch::IntegrationTest**

**def setup; end**

**test "should get input" do**

**get proxy\_input\_url**

**assert\_response :success**

**end**

**test "should get output" do**

**get proxy\_output\_url**

**assert\_response :success**

**end**

**test "should return" do**

**get proxy\_output\_url params: { input: 127, side: 'client\_with\_xslt'}**

**result1 = response.parsed\_body**

**get proxy\_output\_url params: { input: 10, side: 'client\_with\_xslt'}**

**result2 = response.parsed\_body**

**assert\_not\_equal result1, result2**

**end**

**test "should return HTML" do**

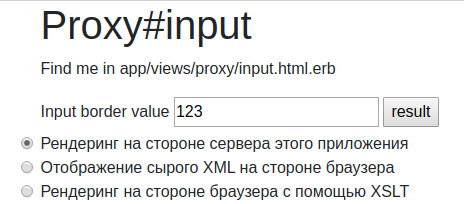
**get proxy\_output\_url, params: { input: 127, side: 'server'}**

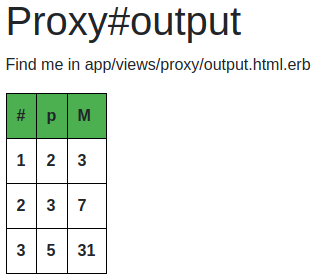
**assert\_select 'table', input: 10**

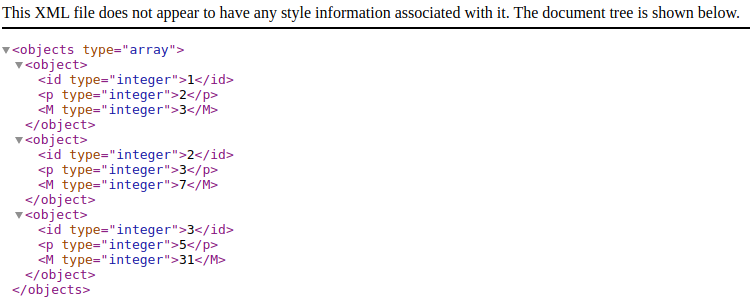
**end**

**end**

**Работа программы:**

****

**Рендеринг на стороне сервера  
**

**Отображение сырого XML на стороне браузера**

**Рендеринг на стороне браузера с помощью XSLT**

**# p M**

**1 2 3**

**2 3 7**

**3 5 31**

**Проверка rubocop:**

**keller@keller-VirtualBox:~/Desktop/lb10-proxy$ rubocop**

**Inspecting 41 files**

**41 files inspected, no offenses detected**

**Автоматически тесты:**

**/home/keller/Desktop/lb10-proxy/db/schema.rb doesn't exist yet. Run `rails db:migrate` to create it, then try again. If you do not intend to use a database, you should instead alter /home/keller/Desktop/lb10-proxy/config/application.rb to limit the frameworks that will be loaded.**

**Started**

**Finished in 1.79107s**

**4 tests, 4 assertions, 0 failures, 0 errors, 0 skips**

**Вывод:** Модифицировали код ЛР 8 таким образом, чтобы по запросу с указанными параметрами выдавался результат в формате XML. Разработали XSLT-программу преобразования полученной XML в HTML. Программа протестирована и корректно работает.