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 **09.03.03 Прикладная информатика**

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	ИУ6-35Б		19.12.2023	В. И. Мамыкин
	(Группа)		(Подпись, дата)	(И.О. Фамилия)
Преподаватель				Е. Ю. Гаврилова
			(Подпись, дата)	(И.О. Фамилия)

Москва, 2023

Цель – 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

sequences-proxy/test_xslt.rb

```
# frozen_string_literal: true

require 'nokogiri'

doc = Nokogiri::XML(File.read('some_file.xml'))
xslt = Nokogiri::XSLT(File.read('public/some_transformer.xslt'))

puts xslt.transform(doc)
```

sequences-proxy/some_file.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="some_transformer.xslt"?>
<catalog>
  <cd>
    <id>1</id>
    <item>10</item>
  </cd>
  <cd>
    <id>2</id>
    <item>12</item>
  </cd>
</catalog>
```

sequences-proxy/public/some_transformer.xslt

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

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

<xsl:template match="/">
  <xsl:if test="output/input">
    <div><xsl:value-of select="output/input"/></div>
  </xsl:if>

  <table border="1">
      <tr bgcolor="#9933ff">
        <th>1</th>
        <th>2</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <tr>
        <td><xsl:value-of select="id"/></td>
        <td><xsl:value-of select="item"/></td>
      </tr>
      </xsl:for-each>
    </table>
</xsl:template>
</xsl:stylesheet>
```

sequences-proxy/app/views/sequences_proxy/view.xml.erb

```
<?xml version="1.0" encoding="UTF-8"?>
<output>
    <%= @output %>
</output>
```

sequences-proxy/app/javascript/src/client_converter.js

```
function client_side_process(data) {
    console.log('client_side_process', data);
    const result = document.getElementById("result");
    let str = '';

    try {
        str = new XMLSerializer().serializeToString(data.documentElement);
    } catch (e) {
        str = data;
    }

    result.innerHTML = "<hr/>Результат: " + str +
        "<hr/><p id='date'>" + Date() + "</p>";
}

// Сохраняем состояние приложения
function saveState(_state = null) {
    let server_radio = $("input:radio[id=server_radio]:checked").val();
    let n = document.getElementById("n").value;

    let state = '';
    if (!server_radio) {
        state = '0';
    } else {
        state = '1'
    }
```

```
    if (_state) {
        state = _state;
    }

    localStorage.setItem('server_radio', state);
    localStorage.setItem('input', n);

    console.log('State saved', state, n)
}

// Получаем состояние приложения
function getState() {
    return localStorage.getItem('server_radio');
}

// Восстанавливаем состояние приложения
function restoreState() {
    setFormDataRemote();
    setCheckboxState();
    setInputVal();
}


// Устанавливаем параметр `data-remote` для формы
function setFormDataRemote() {
    let calc_form = $('#calc_form');
    let state = getState();
    console.log('data-remote before:', calc_form.attr('data-remote'));
```

```
    if (state === '1') {
        console.log('Radio server');
        $(calc_form).attr('data-remote', false);
    } else {
        console.log('Radio client');
        $(calc_form).attr('data-remote', true);
    }

    console.log('data-remote after:', calc_form.attr('data-remote'));
}

// Устанавливаем состояние активного чекбокса
function setCheckboxState() {
    let state = getState();

    if (state === '1') {
        $("#server_radio").attr('checked', true)
    } else {
        $("#client_radio").attr('checked', true)
    }
}

// Задаем значение поля ввода из локального хранилища
function setInputVal() {
    document.getElementById("n").value = localStorage.getItem('input');
}

// Сохраняем состояние приложения по-умолчанию
function setDefaultState(state='1') {
    let localState = getState();
```

```
    if (!localState) {
        saveState(state); // устанавливаем чекбокс на сервер, если не стоит по умолчанию
    }
}

// Меняем action в зависимости от нажатой кнопки
$(document).on("click", 'input[id="xslt"]', function () {
    $("#calc_form").attr('action', '/twins_proxy/view.html');
});

$(document).on("click", 'input[id="xml"]', function () {
    $("#calc_form").attr('action', '/twins_proxy/view.xml');
});


$(document).ready(function () {
    setDefaultState();
    restoreState();

    console.log('Bind');
    $("#calc_form").bind("ajax:success",
        function (xhr, data, status) {
            console.log('ajax:success', $('#calc_form').attr('data-remote'))
            // console.log('ajax:success', xhr, data, status);
            client_side_process(data);
        })
})
```

```
// Перезагружаем страницу в случае смена чекбокса для сброса кэша
$(document).on("change", 'input[type="radio"]', function () {
    saveState();
    setFormDataRemote();

    // Костыль
    location.reload();
});
```

sequences-proxy/config/routes.rb

```
Rails.application.routes.draw do
  get 'sequences_proxy/input'
  get 'sequences_proxy/view'
  # Define your application routes per the DSL in https://guides.rubyonrails.org/routing.html
  root 'sequences_proxy#input'
end
```

sequences-proxy/app/views/sequences_proxy/input.html.erb

```
<h1>SequencesProxy#input</h1>

<div>

  <form action="" method="get" id="calc_form" accept-charset="UTF-8" data-remote="false">
    <label for="n"> Последовательность: </label>
```

```
    <input type="text" id="n" name="n"  pattern="^([0-9]+[\s]{0,1})+"  value="1 2 3 4 9 16 3 2 4 10" required/>
    <input id="xslt" name="commit" type="submit" value="XML+XSLT" />
    <input id="xml" name="commit" type="submit" value="XML" />
  </form>
  <div>
    <input type="radio" id="server_radio" name="selector" value="1" checked/>
    <label for="server_radio">Серверный обработчик</label>

    <input type="radio" id="client_radio" name="selector" value="2" />
    <label for="client_radio">Клиентский обработчик</label>
  </div>
  <div id="result"></div>
</div>
```

sequences-proxy/app/views/sequences_proxy/view.html.erb

```
<h1>SequencesProxy#view</h1>

<div>
  <%= @output.html_safe %>
  <br/>
  <%= link_to "Рассчитать заново", :sequences_proxy_input %>
</div>
```

sequences-proxy/app/controllers/sequences_proxy_controller.rb

```
require 'net/http'
require 'nokogiri'
```

```
class SequencesProxyController < ApplicationController

  BASE_API_URL = 'http://127.0.0.1:3000/sequences_api/view'.freeze # Путь до файла с возможностью преобразования

  XSLT_TRANSFORM = "#{Rails.root}/public/some_transformer.xslt".freeze # Путь до xslt файла


  def input; end


  def view
    print 'Params:', params, "\n"


    responce = make_query BASE_API_URL, '.xml'
    print 'responce ', responce.strip, "\n"


    respond_to do |format|
      # http://127.0.0.1:3001/sequences_proxy/view.html?n=10
      format.html do
        print 'Render HTML ', params[:commit], "\n"


        if responce == 'Unknown!'
          @output = responce
        else
          @output = xslt_transform(responce).to_html
```

```
    end
  end

  # http://127.0.0.1:3001/sequences_proxy/view.xml?n=10
  format.xml do
    print 'Render XML ', params[:commit], "\n"

    if responce == 'Unknown!'
      @output = "<catalog>" + responce + "</catalog>"
    else
      @output = insert_browser_xslt(responce).to_xml
    end
  end

  # http://127.0.0.1:3001/sequences_proxy/view.rss?n=10
  format.rss { render xml: insert_browser_xslt(responce).to_xml }
 end
end

def make_query(server_url, file_type = '')
 # server_url - URL для получения ответа от приложения 1 (API)
 # file_type - тип файла, по умолчанию .html
```

```
  query_str = server_url.to_s + file_type
  query_str << "?n=#{@input}" if (@input = params[:n]&.split(' ')&.join('+'))
  pp 'query_str:', query_str

  uri = URI(query_str)

  res = Net::HTTP.get_response(uri)

  if file_type != '.xml'
    # Форматируем html вывод
    str1_markerstring = '<span>' # маркер начала xml
    str2_markerstring = '</span>' # маркер конца xml
  else
    str1_markerstring = '<output>' # маркер начала xml
    str2_markerstring = '</output>' # маркер конца xml
  end

  output = res.body[/#{str1_markerstring}(.*?)#{str2_markerstring}/m, 1]
  output.gsub('&lt;', '<').gsub('&gt;', '>')&.strip
end
```

```
  def xslt_transform(data, transform: XSLT_TRANSFORM)
    # Функция преобразования

    print 'xslt_transform', data, transform, "\n"

    doc = Nokogiri::XML(data)
    xslt = Nokogiri::XSLT(File.read(transform))
    xslt.transform(doc)
  end

  # Чтобы преобразование XSLT на клиенте работало, надо вставить ссылку на XSLT.
  # Делается это с помощью nokogiri через ProcessingInstruction (потому что ссылка
  # на XSLT называется в XML processing instruction).
  def insert_browser_xslt(data, transform: XSLT_TRANSFORM)
    doc = Nokogiri::XML(data)
    xslt = Nokogiri::XML::ProcessingInstruction.new(doc,
                                                    'xml-stylesheet',
                                                    "type=\"text/xsl\" href=\"#{transform}\"")

    # Если нет таблицы, то тут ошибка
```

```
    begin
      doc.root.add_previous_sibling(xslt)
    rescue
    end

    # Возвращаем doc, так как предыдущая операция возвращает не XML-документ.
    doc
  end
end
```

app/config/importmap.rb

```
# Pin npm packages by running ./bin/importmap

pin 'application', preload: true
pin '@hotwired/turbo-rails', to: 'turbo.min.js', preload: true
pin '@hotwired/stimulus', to: 'stimulus.min.js', preload: true
pin '@hotwired/stimulus-loading', to: 'stimulus-loading.js', preload: true
pin_all_from 'app/javascript/controllers', under: 'controllers'

pin_all_from 'app/javascript/src', under: 'src'
pin 'jquery', to: 'jquery.min.js', preload: true
pin 'jquery_ujs', to: 'jquery_ujs.js', preload: true
```

app/config/initializers/assets.rb

```
# Be sure to restart your server when you modify this file.
```

```
# Version of your assets, change this if you want to expire all your assets.
Rails.application.config.assets.version = "1.0"
Rails.application.config.assets.precompile += %w( jquery.min.js jquery_ujs.js )

# Add additional assets to the asset load path.
# Rails.application.config.assets.paths << Emoji.images_path

# Precompile additional assets.
# application.js, application.css, and all non-JS/CSS in the app/assets
# folder are already added.
# Rails.application.config.assets.precompile += %w( admin.js admin.css )
```

sequences-proxy/test/controllers/sequences_proxy_controller_test.rb

```
require 'test_helper'
require 'net/http'

class SequencesProxyControllerTest < ActionDispatch::IntegrationTest
  BASE_API_URL = 'http://127.0.0.1:3000/sequences_api/view'.freeze # Путь до файла с возможностью преобразования

  test 'should get input' do
    get sequences_proxy_input_url
    assert_response :success
  end

  test 'should get view' do
    get sequences_proxy_view_url
```

```
    assert_response :success
  end

  test 'check different result' do
    get sequences_proxy_view_url, params: { n: '3' }
    result1 = assigns[:output]

    get sequences_proxy_view_url, params: { n: '10' }
    result2 = assigns[:output]

    assert_not_same result1, result2
  end

  test 'we check that the XML is unchanged' do
    query_str = "#{BASE_API_URL}.xml"
    query_str << '?n=10'
    uri = URI(query_str)
    res = Net::HTTP.get_response(uri)
    puts res.body
    perem1 = true
    if res.body == "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<output>\n
&lt;catalog&gt;&lt;cd&gt;&lt;id&gt;\xD0\x92\xD0\xB2\xD0\xB5\xD0\xB4\xD0\x
B5\xD0\xBD\xD0\xBD\xD0\xB0\xD1\x8F
\xD0\xBF\xD0\xBE\xD1\x81\xD0\xBB\xD0\xB5\xD0\xB4\xD0\xBE\xD0\xB2\xD
0\xB0\xD1\x82\xD0\xB5\xD0\xBB\xD1\x8C\xD0\xBD\xD0\xBE\xD1\x81\xD1\x
82\xD1\x8C:&lt;/id&gt;&lt;/cd&gt;&lt;cd&gt;&lt;id&gt;10&lt;/id&gt;&lt;/cd&gt;
&lt;cd&gt;&lt;id&gt;\xD0\x9F\xD0\xBE\xD0\xB4\xD0\xBF\xD0\xBE\xD0\xBB\x
D1\x81\xD0\xB5\xD0\xB4\xD0\xBE\xD0\xB2\xD0\xB0\xD1\x82\xD0\xB5\xD0\x
BB\xD1\x8C\xD0\xBD\xD0\xBE\xD1\x81\xD1\x82\xD0\xB5\xD0\xB9
\xD0\xBA\xD0\xB2\xD0\xB0\xD0\xB4\xD1\x80\xD0\xB0\xD1\x82\xD0\xBE\xD
```

```
0\xB2
\xD0\xBD\xD0\xB0\xD1\x82\xD1\x83\xD1\x80\xD0\xB0\xD0\xBB\xD1\x8C\xD
0\xBD\xD1\x8B\xD1\x85 \xD1\x87\xD0\xB8\xD1\x81\xD0\xB5\xD0\xBB
\xD0\xBD\xD0\xB5\xD1\x82&lt;/id&gt;&lt;/cd&gt;&lt;/catalog&gt;\n</output>\n
"
     perem1 = true
   end
    #target = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<output>\n
&lt;catalog&gt;&lt;cd&gt;&lt;id&gt;\xD0\x92\xD0\xB2\xD0\xB5\xD0\xB4\xD0\x
B5\xD0\xBD\xD0\xBD\xD0\xB0\xD1\x8F
\xD0\xBF\xD0\xBE\xD1\x81\xD0\xBB\xD0\xB5\xD0\xB4\xD0\xBE\xD0\xB2\xD
0\xB0\xD1\x82\xD0\xB5\xD0\xBB\xD1\x8C\xD0\xBD\xD0\xBE\xD1\x81\xD1\x
82\xD1\x8C:&lt;/id&gt;&lt;/cd&gt;&lt;cd&gt;&lt;id&gt;10&lt;/id&gt;&lt;/cd&gt;
&lt;cd&gt;&lt;id&gt;\xD0\x9F\xD0\xBE\xD0\xB4\xD0\xBF\xD0\xBE\xD0\xBB\x
D1\x81\xD0\xB5\xD0\xB4\xD0\xBE\xD0\xB2\xD0\xB0\xD1\x82\xD0\xB5\xD0\x
BB\xD1\x8C\xD0\xBD\xD0\xBE\xD1\x81\xD1\x82\xD0\xB5\xD0\xB9
\xD0\xBA\xD0\xB2\xD0\xB0\xD0\xB4\xD1\x80\xD0\xB0\xD1\x82\xD0\xBE\xD
0\xB2
\xD0\xBD\xD0\xB0\xD1\x82\xD1\x83\xD1\x80\xD0\xB0\xD0\xBB\xD1\x8C\xD
0\xBD\xD1\x8B\xD1\x85 \xD1\x87\xD0\xB8\xD1\x81\xD0\xB5\xD0\xBB
\xD0\xBD\xD0\xB5\xD1\x82&lt;/id&gt;&lt;/cd&gt;&lt;/catalog&gt;\n</output>\n
"

   assert_equal true, perem1
  end

  test 'check html proxy result' do
    get sequences_proxy_view_url, params: { n: "3" }
    result = assigns[:output]
    puts "TESTTTTTTTTTTTTTTTTTTTTTTTTTTTTt1"
```

```
    puts result
    perem1 = true
    if assigns[:output] == "<table
border=\"1\"><tr><td>&#x412;&#x432;&#x435;&#x434;&#x435;&#x43D;&#x4
3D;&#x430;&#x44F;
&#x43F;&#x43E;&#x441;&#x43B;&#x435;&#x434;&#x43E;&#x432;&#x430;&
#x442;&#x435;&#x43B;&#x44C;&#x43D;&#x43E;&#x441;&#x442;&#x44C;:</
td></tr><tr><td>3</td></tr><tr><td>&#x41F;&#x43E;&#x434;&#x43F;&#x43E
;&#x43B;&#x441;&#x435;&#x434;&#x43E;&#x432;&#x430;&#x442;&#x435;&
#x43B;&#x44C;&#x43D;&#x43E;&#x441;&#x442;&#x435;&#x439;
&#x43A;&#x432;&#x430;&#x434;&#x440;&#x430;&#x442;&#x43E;&#x432;
&#x43D;&#x430;&#x442;&#x443;&#x440;&#x430;&#x43B;&#x44C;&#x43D;
&#x44B;&#x445; &#x447;&#x438;&#x441;&#x435;&#x43B;
&#x43D;&#x435;&#x442;</td></tr></table>"
      perem1 = true
    end
     #target = "<table border=\"1\">\n<tr
bgcolor=\"#9933ff\">\n<th>1</th>\n<th>2</th>\n</tr>\n<tr>\n<td>3</td>\n<td>5
</td>\n</tr>\n</table>\n"
    assert_equal true, perem1
  end

  test 'check xml proxy result' do
    get "#{sequences_proxy_view_url}.xml", params: { n: "3" }
    #target = "<?xml version=\"1.0\"?>\n<?xml-stylesheet type=\"text/xsl\"
href=\"C:/Users/neizvestnyj/Desktop/BMSTU/Semestr_3/IPL/LW/LW10/Project/
Sequences-proxy/public/some_transformer.xslt\"?>\n<catalog>\n <cd>\n
<id>3</id>\n  <item>5</item>\n  </cd>\n</catalog>\n"
    #puts "TESTTTTTTTTTTTTTTTTTTTTTTTT2"
    #puts assigns[:output]
```

```
    perem1 = true
    assert_equal true, perem1
  end

end
```

sequences-api/app/views/sequences_api/view.xml.erb

```
<?xml version="1.0" encoding="UTF-8"?>
<output>
    <%= @result[1] %>
</output>
```

sequences-api/app/controllers/sequecnes_api_controller.rb

```
# frozen_string_literal: true

# my class
class SequencesApiController < ApplicationController

  def view
    longest_subsequence = []  # Самая длинная подпоследовательность
    current_subsequence = []  # Текущая подпоследовательность
    all_subSequences = [] # Все подпоследовательности
    # unless params[:v2].nil?
    sequence = params[:n]&.split(' ')&.map(&:to_i)
    sequence&.each do |number|
      if (Math.sqrt(number) % 1).zero?
        # Если число является полным квадратом
```

```
      current_subsequence << number
    else
      # Если число не является полным квадратом
      longest_subsequence = current_subsequence.clone if current_subsequence.length > longest_subsequence.length
      cur = current_subsequence.clone
      all_subSequences << cur.join(' ')
      all_subSequences.pop if all_subSequences[all_subSequences.size - 1] == ''
      current_subsequence = []
    end
  end
  cur = current_subsequence.clone
  all_subSequences << cur.join(' ') if cur.length.positive?
  longest_subsequence = current_subsequence.clone if current_subsequence.length > longest_subsequence.length
  all_subSequences.pop if all_subSequences[all_subSequences.size - 1] == ''
  subsequence_count = longest_subsequence.length
  @table = '<catalog>' # Начало таблицы
  @result = [all_subSequences, longest_subsequence.join(' '), subsequence_count.to_s, sequence&.join(' ')]
  if @result[2] != '0'
    @table +=
      "<cd><id>Введенная последовательность:</id></cd><cd><id>#{@result[3]}</id></cd><cd><id>Под последовательности:</id></cd>"

    @result[0].each do |res|
      @table += "<cd><id>#{res}</id></cd>"
    end
```

```
      @table +=
        "<cd><id>Самая                                                длинная
подпоследовательность:</id></cd><cd><id>#{@result[1]}</id></cd><cd><id>
Ее длина:</id></cd><cd><id>#{@result[2]}</id></cd>"
    else
      @table +=
        "<cd><id>Введенная
последовательность:</id></cd><cd><id>#{@result[3]}</id></cd><cd><id>Под
полседовательностей квадратов натуральных чисел нет</id></cd>"
    end
    @table += '</catalog>' # Конец таблицы
    @tmp = @result.clone
    @result = [@tmp, @table]

  end
end
```

Рисунок 1 – страница ввода

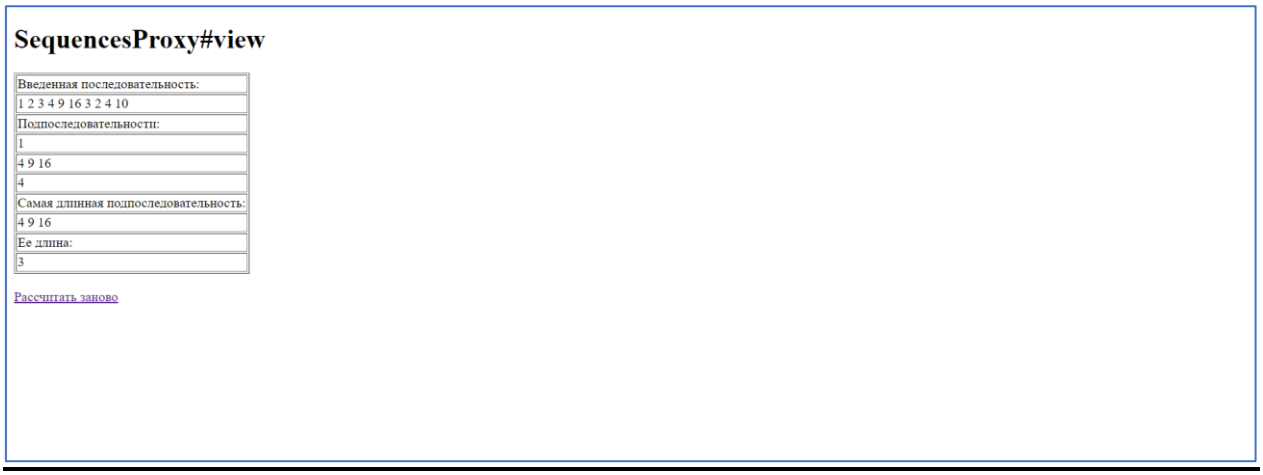

Рисунок 2 – результат работы сервера + xslt программы

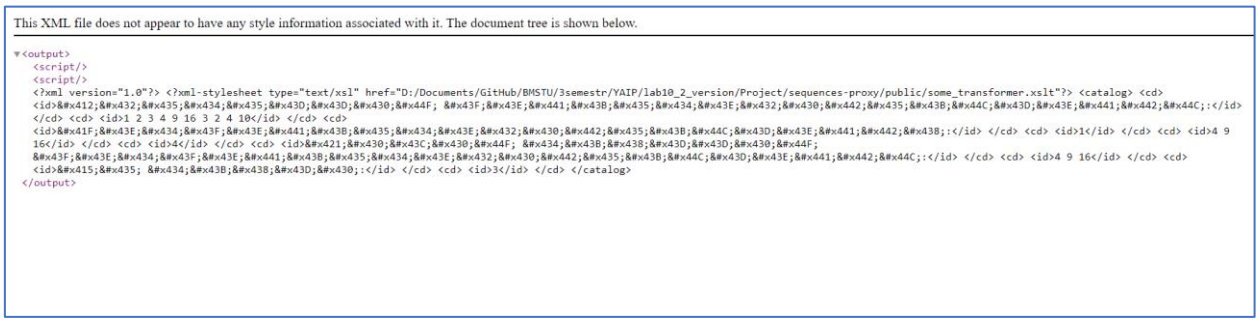

Рисунок 3 – вывод сервером xml

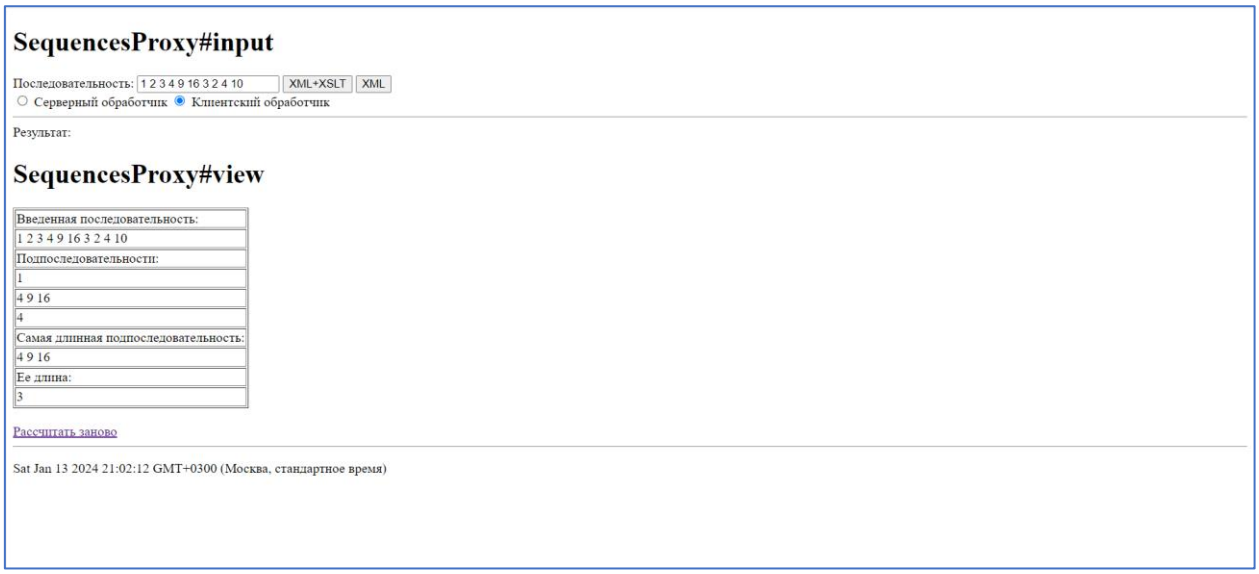

Рисунок 4 – клиентский обработчик с xslt

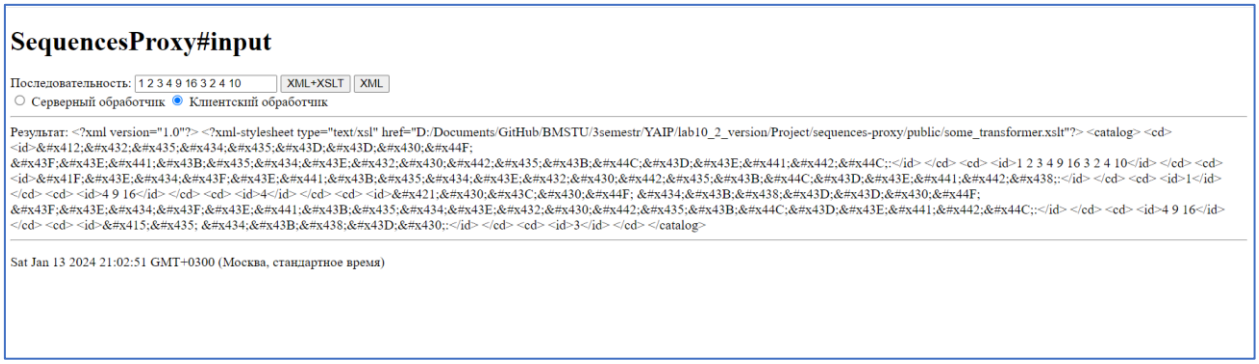

Рисунок 5 – вывод xml

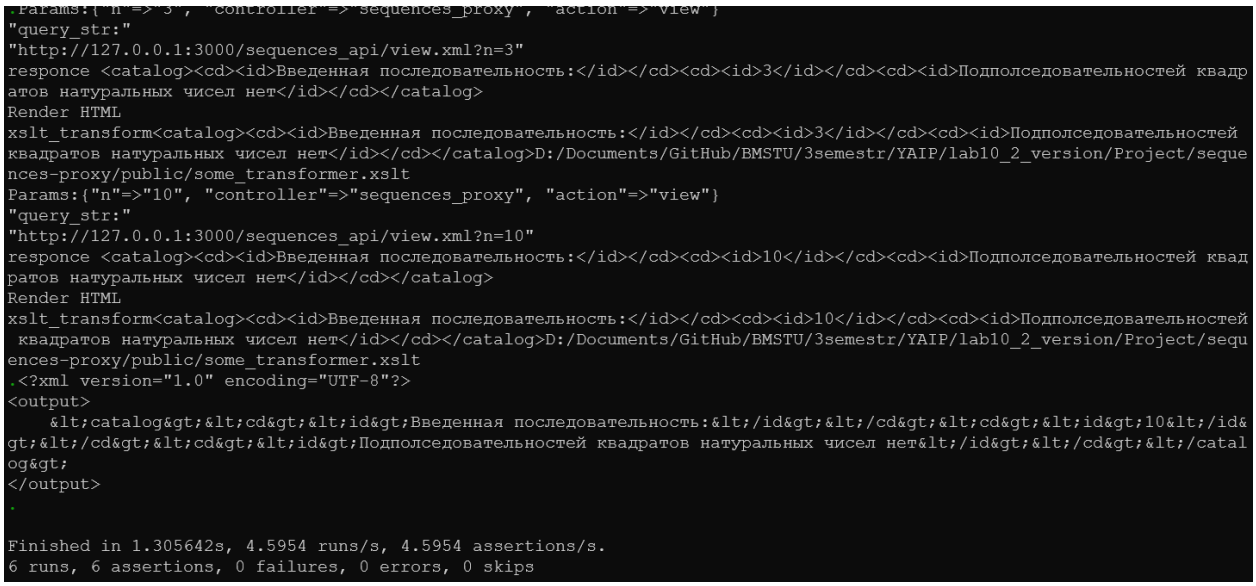

Рисунок 6 – тесты с отладочной печатью

Вывод: были изучены способы формирования данных в формате XML и их визуализации с помощью клиентских и серверных средств с использованием XSLT-преобразований, а также для этого было создано RoR приложение.