|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 10 |

**Название:**

Тема лабораторной работы или домашнего задания

**Дисциплина:** Языки интернет-программирования



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-35 Б |  | 24.11.2023 | И.А. Дулина |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Е.Ю. Гаврилова |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

**Вариант 8**

**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

**Приложение CHISLA\_API**

**chisla-api/app/view/chisla\_api/view.html.erb**

<h1>ChislaApi#view</h1>

<p>Find me in app/views/chisla\_api/view.html.erb</p>

<span> <%= @result[1] %> </span>

**Chisla-api/app/view/chisla\_api/view.xml.erb**

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

<output>

<%= @result[1] %>

</output>

**Chisla-api/app/controllers/chisla\_api\_controller.rb**

class ChislaApiController < ApplicationController

def view

if params[:str]

begin

res = params[:str].scan(/-?\d+(?:\.\d+)?/).map(&:to\_i)

pp 'Изнач массив', res

raise StandardError if res.length < 10

#pp 'RES:', res, params[:str]

@result = create(res)

rescue StandardError

@result = [{}, 'Something is wrong']

end

else

@result = [{}, 'Unknown!']

end

end

def create(res)

max = 0

all = []

solution = ''

i = 0

loop do

posl, len, i = create\_posl(i, res)

pp 'Каждая последовательность', posl

all << posl.join(' ')

if len > max

max = len

solution = posl.join(' ')

end

break if i >= res.length

end

pp 'До формирования',all

result = []

all.length.times do |j|

str = if solution == all[j]

'+'

else

' '

end

result << if j.zero?

[res.join(' '), all[j], str]

else

[' ', all[j], str]

end

end

pp 'вывод create', result

[solution, create\_table(result)]

end

def create\_table(result)

rows = ''

result.each do |init, all, sol|

rows += "<cd><former>#{init}</former><every>#{all}</every><plus>#{sol}</plus></cd>"

end

@table = "<catalog>#{rows}</catalog>"

#@table = "<table border='1' class=\"table\"><tbody>#{rows}</tbody></table>"

end

def create\_posl(i, res)

len = 0

posl = []

loop do

len += 1

posl << res[i]

break if i + 1 == res.length

break if (res[i+1] <= res[i])

i += 1

end

i+=1

[posl, len, i]

end

end

**Chisla-api/config/routes.rb** добавить:

root 'chisla\_api#view'

**Приложение CHISLA\_PROXY**

**Chisla-proxy/test\_xslt.rb (автономаная Ruby программа для теста преобразования xml в html)**

require 'nokogiri'

doc = Nokogiri::XML(File.read('some\_file.xml'))

xslt = Nokogiri::XSLT(File.read('public/some\_transformer.xslt'))

puts xslt.transform(doc)

**Chisla-proxy/some\_file.xml (тестовый .xml файл для test\_xslt.rb)**

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

<?xml-stylesheet type="text/xsl" href="some\_transformer.xslt"?>

<catalog>

<cd>

<former>1 5 6</former>

<every>2 10</every>

<plus>+</plus>

</cd>

<cd>

<former> </former>

<every>2 5</every>

<plus> </plus>

</cd>

<cd>

<former> </former>

<every>2 -1 0 4</every>

<plus> </plus>

</cd>

</catalog>

**Chisla-proxy/public/some\_transformer.xslt (XSLT-программа для преобразования xml в html)**

<?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>Former</th>

<th>All</th>

<th>The longest</th>

</tr>

<xsl:for-each select="catalog/cd">

<tr>

<td><xsl:value-of select="former"/></td>

<td><xsl:value-of select="every"/></td>

<td><xsl:value-of select="plus"/></td>

</tr>

</xsl:for-each>

</table>

</xsl:template>

</xsl:stylesheet>

**Chisla-proxy/app/views/chisla\_proxy/input.html.erb**

<h1>ChislaProxy#input</h1>

<p>Find me in app/views/chisla\_proxy/input.html.erb</p>

<div>

<p>

<a href="<%= url\_for(only\_path: false) + '.html.erb' %>">Мы находимся по

адресу: <%= url\_for(only\_path: false) + '.html.erb' %></a>

</p>

<div>

<input type="radio" id="server\_radio" name="selector" value="1"/>

<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>

<br>

</div>

<form action="" method="get" id="calc\_form" accept-charset="UTF-8" data-remote="false">

<label for="str">Введите не менее 10 чисел

<input type="text" id="str" name="str" required/>

</label>

<br>

<input id="xslt" name="commit" type="submit" value="XML+XSLT" />

<input id="xml" name="commit" type="submit" value="XML" />

</form>

<div id="result"></div>

**Chisla-proxy/app/views/chisla\_proxy/view.html.erb**

<h1>ChislaProxy#view</h1>

<p>Find me in app/views/chisla\_proxy/view.html.erb</p>

<div>

<p>

<a href="<%= url\_for(only\_path: false) + '.html.erb' %>">Мы находимся по

адресу: <%= url\_for(only\_path: false) + '.html.erb' %></a>

</p>

<%= @output.html\_safe %>

<br/>

<%= link\_to "Рассчитать заново", :chisla\_proxy\_input %>

</div>

**Chisla-proxy/app/views/chisla\_proxy/view.xml.erb (xml страница)**

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

<output>

<%= @output %>

</output>

**Chisla-proxy/app/controllers/chisla\_proxy\_controller.rb**

require 'net/http'

require 'nokogiri'

class ChislaProxyController < ApplicationController

BASE\_API\_URL = 'http://127.0.0.1:3000/chisla\_api/view' # Путь до файла с возможность преобразования

XSLT\_SERVER\_TRANSFORM = "#{Rails.root}/public/some\_transformer.xslt" # Путь до xslt файла

def input

end

def view

responce = make\_query BASE\_API\_URL, '.xml'

respond\_to do |format|

format.html do

if responce == 'Unknown!' || responce == "Something is wrong"

@output = responce

else

@output = xslt\_transform(responce).to\_html

end

end

format.xml do

if responce == ' Unknown! ' || responce == "\n" + " Something is wrong\n"

@output = '<catalog>' + responce + '</catalog>'

else

@output = insert\_browser\_xslt(responce).to\_xml

end

end

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)

query\_str = server\_url.to\_s + file\_type

query\_str << "?str=#{@input}" if (@input = params[:str]&.split(' ')&.join('+'))

uri = URI(query\_str)

res = Net::HTTP.get\_response(uri)

# Форматируем html вывод

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\_SERVER\_TRANSFORM)

# Функция преобразования

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\_SERVER\_TRANSFORM)

doc = Nokogiri::XML(data)

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

'xml-stylesheet',

"type=\"text/xsl\" href=\"#{transform}\"")

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

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

doc

end

end

**Chisla-proxy/app/javascript/src/client\_controller.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 str = document.getElementById("str").value;

let state = '';

if (!server\_radio) {

state = '0';

} else {

state = '1'

}

if (\_state) {

state = \_state;

}

localStorage.setItem('server\_radio', state);

localStorage.setItem('input', str);

console.log('State saved', state, str)

}

// 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("str").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', '/chisla\_proxy/view.html');

});

$(document).on("click", 'input[id="xml"]', function () {

$("#calc\_form").attr('action', '/chisla\_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();

});

**Chisla-proxy/app/config/importmap.rb (импорт js)** добавить

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

**Chisla-proxy/config/initializers/assets.rb (компиляция js)** добавить

Rails.application.config.assets.precompile += %w( jquery.min.js jquery\_ujs.js )

**Gemfile** добавить

gem 'jquery-rails'

gem 'rails-controller-testing'

**Chisla-proxy/config/routes.rb** добавить

root 'chisla\_proxy#input'

**Chisla-proxy/test/controllers/chisla\_proxy\_controller\_test.rb**

require "test\_helper"

class ChislaProxyControllerTest < ActionDispatch::IntegrationTest

BASE\_API\_URL = 'http://127.0.0.1:3000/chisla\_api/view'

test "should get input" do

get chisla\_proxy\_input\_url

assert\_response :success

end

test "should get view" do

get chisla\_proxy\_view\_url

assert\_response :success

end

# функциональный тест, проверяющий что при различных входных данных результат генерируемой страницы различен

test 'check differ' do

get chisla\_proxy\_view\_url, params: { str: '1 2 3 4 5 6 7 8 9 10' }

result1 = assigns[:output]

get chisla\_proxy\_view\_url, params: { str: '-1 -2 -3 2 3 4 -2 -3 2 3' }

result2 = assigns[:output]

assert\_not\_same result1, result2

end

# браузер получает XML первого приложения в неизменном виде.

test 'XML is unchanged' do

query\_str = "#{BASE\_API\_URL}.xml"

query\_str << '?str=1+2+3+4+5+6+7+8+9+10'

uri = URI(query\_str)

res = Net::HTTP.get\_response(uri)

target = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<output>\n &lt;catalog&gt;&lt;cd&gt;&lt;former&gt;1 2 3 4 5 6 7 8 9 10&lt;/former&gt;&lt;every&gt;1 2 3 4 5 6 7 8 9 10&lt;/every&gt;&lt;plus&gt;+&lt;/plus&gt;&lt;/cd&gt;&lt;/catalog&gt;\n</output>"

assert\_equal target, res.body

end

#функциональные тесты второго приложения (proxy)

test 'check html proxy' do

get chisla\_proxy\_view\_url, params: { str: '1 2 3 4 5 6 7 8 9 10' }

result = assigns[:output]

target = "<table border=\"1\">\n<tr bgcolor=\"#9933ff\">\n<th>Former</th>\n<th>All</th>\n<th>The longest</th>\n</tr>\n<tr>\n<td>1 2 3 4 5 6 7 8 9 10</td>\n<td>1 2 3 4 5 6 7 8 9 10</td>\n<td>+</td>\n</tr>\n</table>\n"

assert\_equal result, target

end

test 'check xml proxy' do

get "#{chisla\_proxy\_view\_url}.xml", params: { str: '1 2 3 4 5 6 7 8 9 10' }

target = "<?xml version=\"1.0\"?>\n<?xml-stylesheet type=\"text/xsl\" ?>\n<catalog>\n <cd>\n <former>1 2 3 4 5 6 7 8 9 10</former>\n <every>1 2 3 4 5 6 7 8 9 10</every>\n <plus>+</plus>\n </cd>\n</catalog>\n"

result= assigns[:output].gsub(/href=.\*.xslt"/, '') # чтобы без пк и папок

assert\_equal result, target

end

#RSS тест

test 'check rss' do

get "#{chisla\_proxy\_view\_url}.rss", params: { str: '1 2 3 4 5 6 7 8 9 10' }

target = "<?xml version=\"1.0\"?>\n<?xml-stylesheet type=\"text/xsl\" ?>\n<catalog>\n <cd>\n <former>1 2 3 4 5 6 7 8 9 10</former>\n <every>1 2 3 4 5 6 7 8 9 10</every>\n <plus>+</plus>\n </cd>\n</catalog>\n"

assert\_equal @response.body.clone.gsub(/href=.\*.xslt"/, ''), target

end

test 'different for rss requests' do

get "#{chisla\_proxy\_view\_url}.rss", params: { str: '1 2 3 4 5 6 7 8 9 10' }

response1 = @response.body.clone

get "#{chisla\_proxy\_view\_url}.rss", params: { str: '-1 -2 -3 4 5 -6 7 8 -9 10' }

response2 = @response.body.clone

assert\_not\_equal response1, response2

end

end



Рисунок 1 – главная страница с вводом и выбором приложения

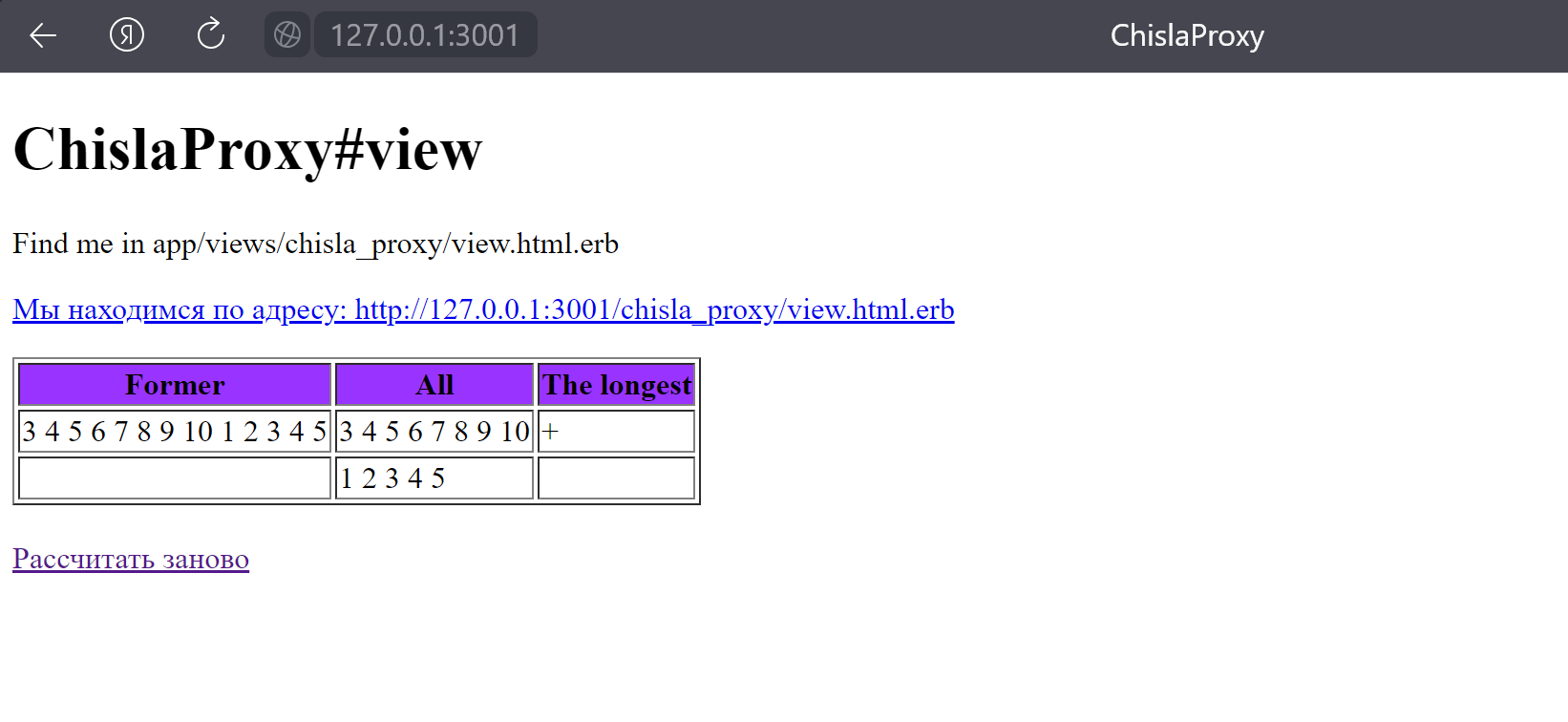


Рисунок 2 – результат работы сервера и XSLT программы (преобразователь)

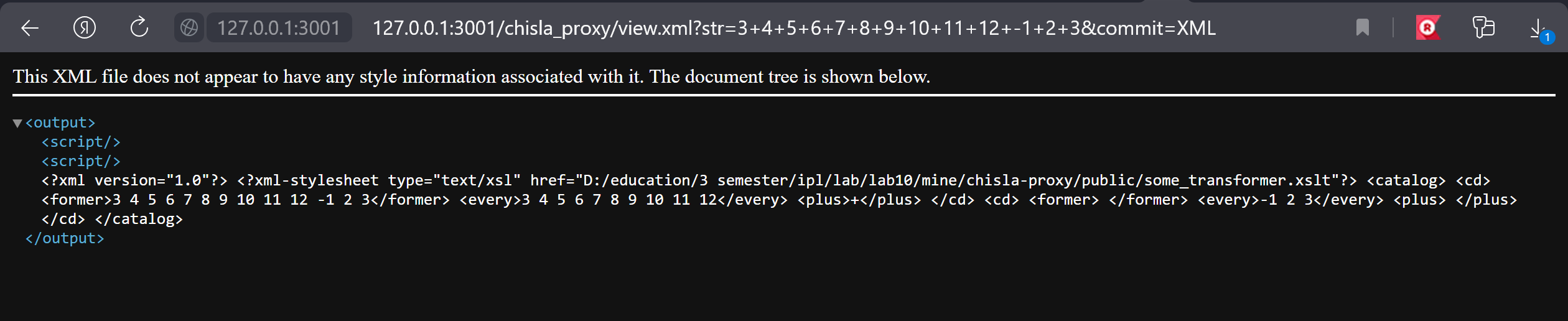


Рисунок 3 – результат работы сервера по выводу XML



Рисунок 4 – результат работы клиентского обработчика и XSLT программы

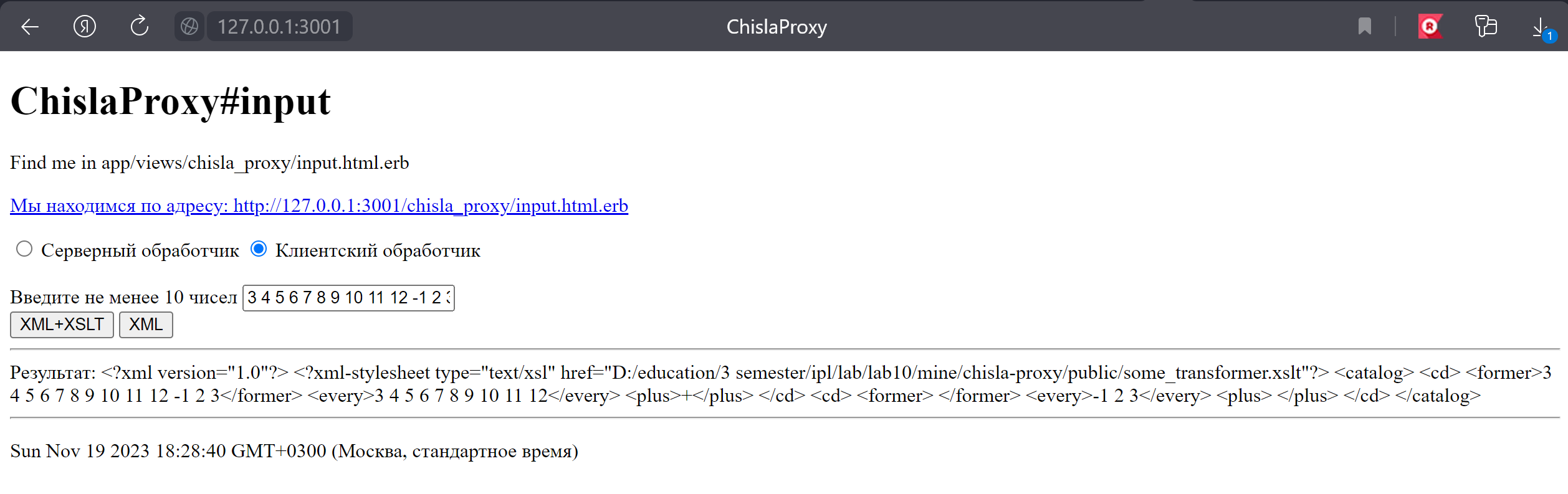


Рисунок 5 – результат работы клиентского обработчика по выводу XML

Рисунок 6 – результат работы тестов

**Вывод**: было изучено формирование данных в формате XML и их визуализации с помощью клиентских и серверных средств с использованием XSLT-преобразований, было сделано веб-приложения с выбором клиентской или серверной стороны и с выводом результата либо в HTML, либо в XML формате.