|  |  |
| --- | --- |
| 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:**

Формирование и отображение XML в HTML средствами сервера и клиента.

**Дисциплина:** Языки Интернет-программирования



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-35Б |  | 27.10.2023 | И.А. Буханцев |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Е.Ю. Гаврилова |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

**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

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

**twins-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**)**

**twins-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>**

**twins-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>**

**twins-proxy/app/views/proxy\_view/view.xml.erb**

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

**<output>**

<%= @output %**>**

**</output>**

**twins-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();

});

**twins-proxy/config/routes.rb**

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

get 'twins\_proxy/input'

get 'twins\_proxy/view'

*# Define your application routes per the DSL in https://guides.rubyonrails.org/routing.html*

root 'twins\_proxy#input'

**end**

**twins-proxy/app/views/proxy\_view/input.html.erb**

<**h1**>TwinsProxy#input</**h1**>

<**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 обработчки</**label**>

</**div**>

<**br**>

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

<**label** for="n">

<**input** type="number" id="n" name="n" min="0" minlength="1" oninput="this.value = Math.abs(this.value);" required />

</**label**>

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

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

</**form**>

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

</**div**>

**twins-proxy/app/views/proxy\_view/view.html.erb**

<**h1**>TwinsProxy#view</**h1**>

<**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", :twins\_proxy\_input %>

</**div**>

**twins-proxy/app/controllers/twins\_proxy\_controller.rb**

**require** 'net/http'

**require** 'nokogiri'

**class** TwinsProxyController **<** ApplicationController

BASE\_API\_URL = 'http://127.0.0.1:3000/twins\_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'

**respond\_to** **do** **|format|**

*# http://127.0.0.1:3001/twins\_proxy/view.html?n=10*

**format**.html **do**

**print** 'Render HTML ', params**[:commit]**, "\n"

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

**end**

*# http://127.0.0.1:3001/twins\_proxy/view.xml?n=10*

**format**.xml **do**

**print** 'Render XML ', params**[:commit]**, "\n"

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

**end**

*# http://127.0.0.1:3001/twins\_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;', '>'**)**

**end**

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

*# Функция преобразования*

**print** 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}\""**)**

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

*# 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 )*

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

**require** 'test\_helper'

**require** 'net/http'

**class** TwinsProxyControllerTest **<** **ActionDispatch::IntegrationTest**

BASE\_API\_URL = 'http://127.0.0.1:3000/twins\_api/view'.freeze *# Путь до файла с возможностью преобразования*

test 'should get input' **do**

get twins\_proxy\_input\_url

assert\_response **:success**

**end**

test 'should get view' **do**

get twins\_proxy\_view\_url

assert\_response **:success**

**end**

test 'check different result' **do**

get twins\_proxy\_view\_url, params: **{** n: 3 **}**

result1 = assigns**[:output]**

get twins\_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**)**

target = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<output>\n &lt;catalog&gt;&lt;cd&gt;&lt;id&gt;11&lt;/id&gt;&lt;item&gt;13&lt;/item&gt;&lt;/cd&gt;&lt;cd&gt;&lt;id&gt;17&lt;/id&gt;&lt;item&gt;19&lt;/item&gt;&lt;/cd&gt;&lt;/catalog&gt;\n</output>\n"

**assert\_equal** target, res.body

**end**

test 'check html proxy result' **do**

get twins\_proxy\_view\_url, params: **{** n: 3 **}**

result = assigns**[:output]**

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** result, target

**end**

test 'check xml proxy result' **do**

get "#{twins\_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/twins-proxy/public/some\_transformer.xslt\"?>\n<catalog>\n <cd>\n <id>3</id>\n <item>5</item>\n </cd>\n</catalog>\n"

**assert\_equal** assigns**[:output]**, target

**end**

test 'check rss' **do**

get "#{twins\_proxy\_view\_url}.rss", 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/twins-proxy/public/some\_transformer.xslt\"?>\n<catalog>\n <cd>\n <id>3</id>\n <item>5</item>\n </cd>\n</catalog>\n"

**assert\_equal** **@response**.body.**clone**, target

**end**

test 'should get different responds for different rss requests' **do**

get "#{twins\_proxy\_view\_url}.rss", params: **{** n: 3 **}**

response1 = **@response**.body.**clone**

get "#{twins\_proxy\_view\_url}.rss", params: **{** n: 10 **}**

response2 = **@response**.body.**clone**

**assert\_not\_equal** response1, response2

**end**

**end**

**twins-api/app/views/twins\_api/view.xml.erb**

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

**<output>**

<%= @result[1] %**>**

**</output>**

**twins-api/app/controllers/twins\_api\_controller.rb**

**class** TwinsApiController **<** ApplicationController

**def** view

n = params**[:n]**.**to\_i**

*# Возвращаем результат*

**@result** = twins**(**n**)**

**end**

**def** is\_prime**(**num**)**

**return** **false** **if** num **<**= 1

**Math**.sqrt**(**num**)**.**to\_i**.downto**(**2**)**.**each** **{** **|**i**|** **return** **false** **if** **(**num **%** i**)**.zero? **}**

**true**

**end**

**def** twins**(**n**)**

pairs = **{}**

**(**n..2 **\*** n**)**.**each** **do** **|first|**

**(first**..2 **\*** n**)**.**each** **do** **|**second**|**

**if** **(**second **-** **first** == 2**)** **&&** is\_prime**(first)** **&&** is\_prime**(**second**)**

pairs**[first]** = second

**end**

**end**

**end**

**[**pairs, to\_table**(**pairs**)]**

**end**

**def** to\_table**(**pairs = **@result)**

*# Создание xml таблицы*

**@table** = 'Unknown!'

**unless** pairs.empty?

rows = ''

pairs.**each** **do** **|**key, value**|**

rows **+**= "<cd><id>#{key}</id><item>#{value}</item></cd>"

**end**

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

**end**

**puts** **@table**

**@table**

**end**

**end**

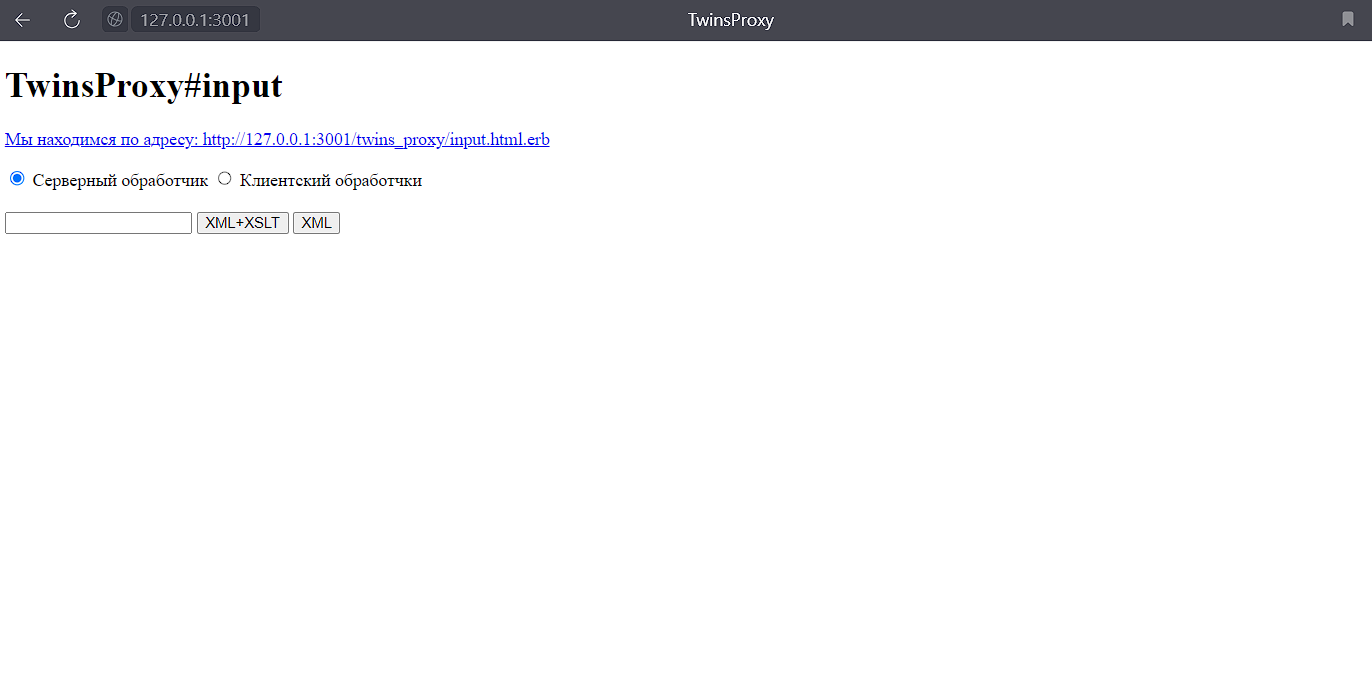
****

Рисунок 1 – страница ввода

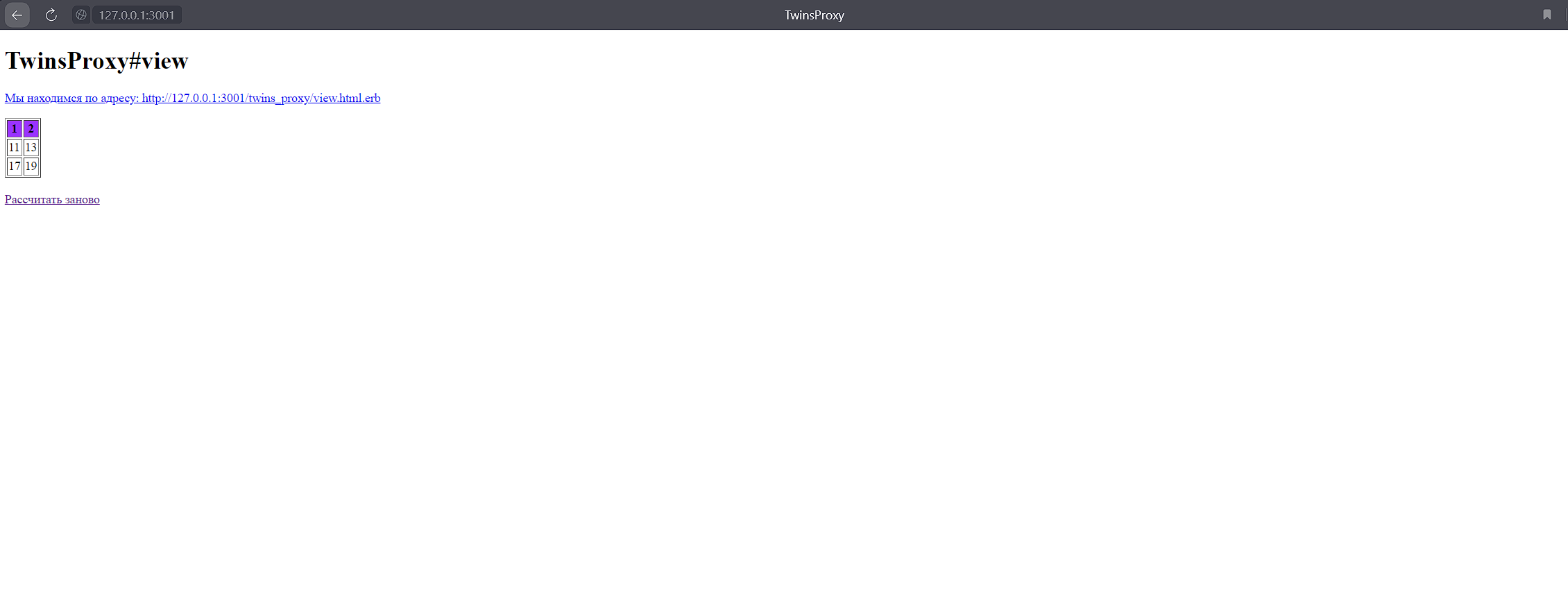
****

Рисунок 2 – результат работы сервера + xslt программы

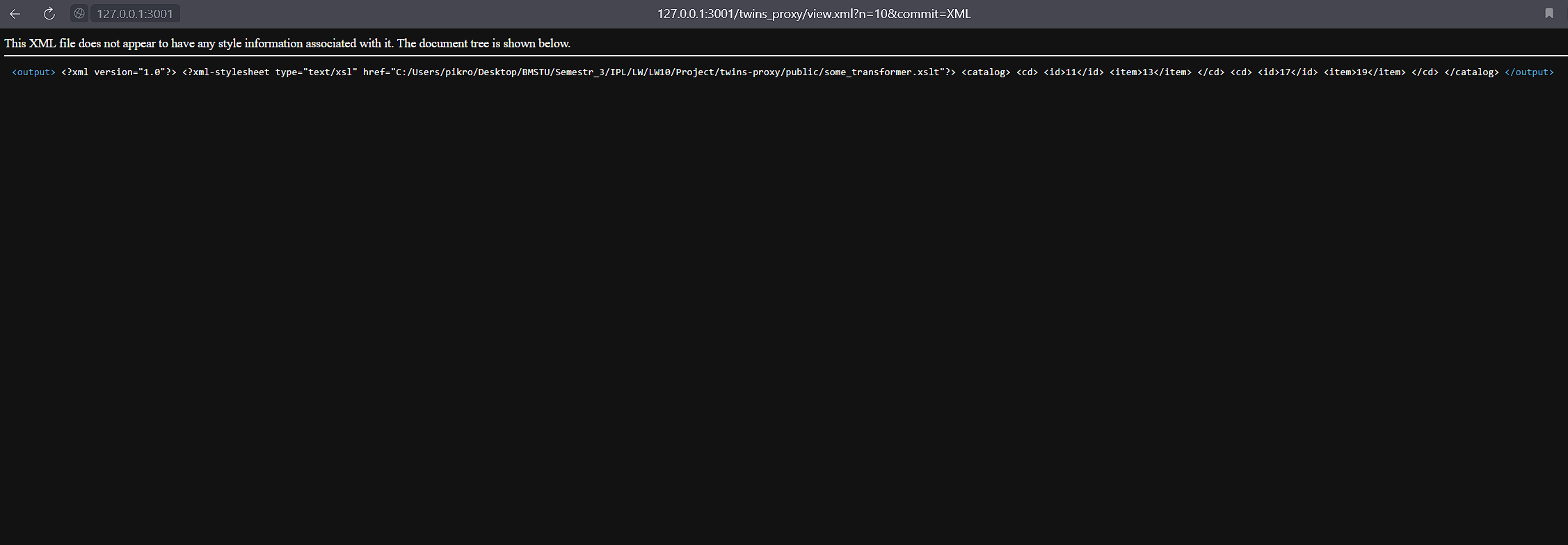
****

Рисунок 3 – вывод сервером xml

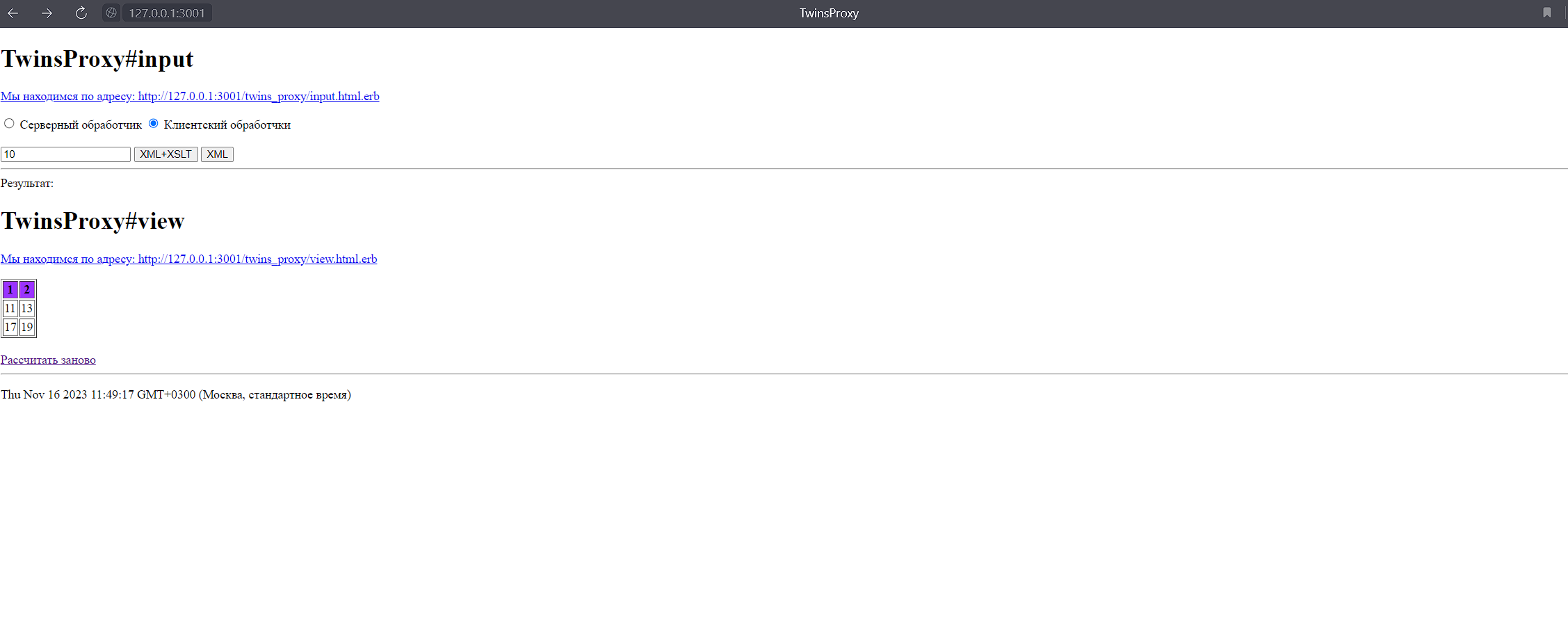
****

Рисунок 5 – клиентский обработчик с xslt

****

Рисунок 6 – вывод xml

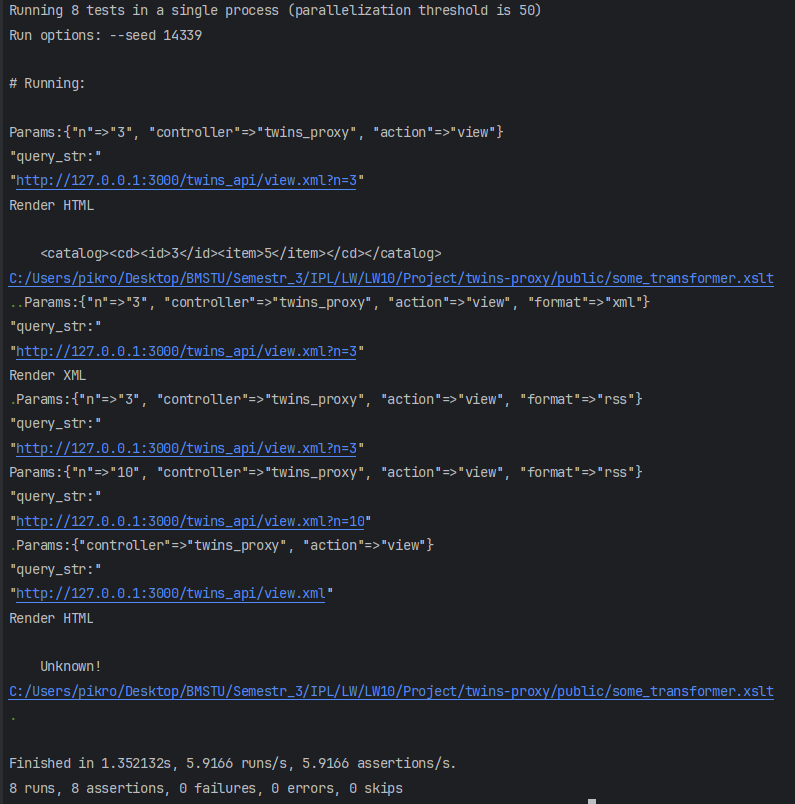
****

Рисунок 7 – тесты

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