



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: Тема лабораторной работы или домашнего задания

Дисциплина: Языки интернет-программирования

Студент

ИУ6-35 Б

(Группа)

  
24.11.2023

(Подпись, дата)

И.А. Дулина

(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 << pos1.join(' ')
if len > max
  max = len
  solution = pos1.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
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
end

```

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

```
root 'chisla_api#view'
```

### **Приложение CHISLA\_PROXY**

**Chisla-proxy/test\_xslt.rb** (автономная 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>

```



```

</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
```

```
    response = make_query BASE_API_URL, '.xml'
```

```
    respond_to do |format|
```

```
      format.html do
```

```
        if response == 'Unknown!' || response == "Something is wrong"
```

```
          @output = response
```

```
        else
```

```
          @output = xslt_transform(response).to_html
```

```
        end
```

```
      end
```

```
      format.xml do
```

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

```
          @output = '<catalog>' + response + '</catalog>'
```

```
        else
```

```
          @output = insert_browser_xslt(response).to_xml
```

```
        end
```

```
      end
```

```

    format.rss { render xml: insert_browser_xslt(response).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/xml\" 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  
<catalog><cd><former>1 2 3 4 5 6 7 8 9  
10</former><every>1 2 3 4 5 6 7 8 9  
10</every><plus>+</plus></cd></catalog>\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

```

← RubyM... 127.0.0.1:3001 ChislaProxy

## ChislaProxy#input

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

[Мы находимся по адресу: http://127.0.0.1:3001/chisla\\_proxy/input.html.erb](http://127.0.0.1:3001/chisla_proxy/input.html.erb)

☒ Серверный обработчик ☐ Клиентский обработчик

Введите не менее 10 чисел

Рисунок 1 – главная страница с вводом и выбором приложения

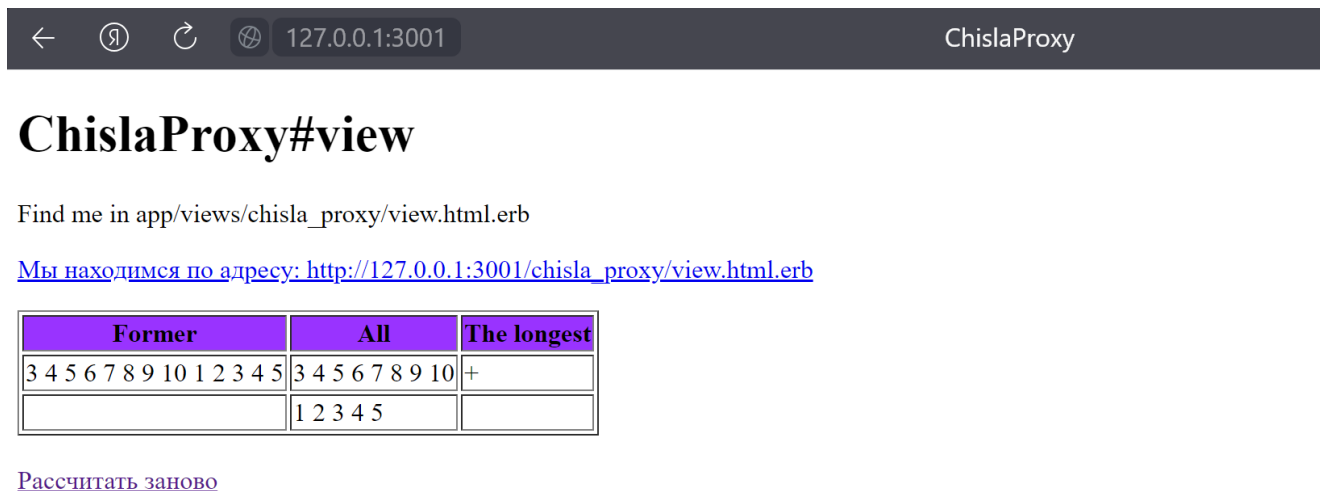


Рисунок 2 – результат работы сервера и XSLT программы (преобразователь)

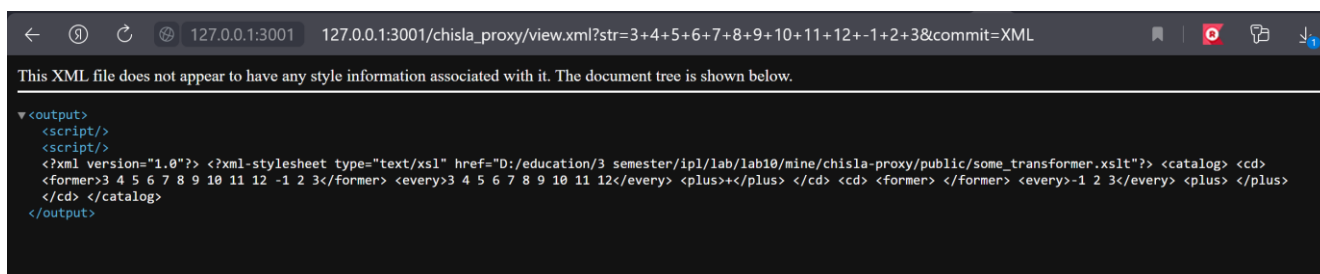


Рисунок 3 – результат работы сервера по выводу XML

## ChislaProxy#input

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

Мы находимся по адресу: [http://127.0.0.1:3001/chisla\\_proxy/input.html.erb](http://127.0.0.1:3001/chisla_proxy/input.html.erb)

☐ Серверный обработчик ☒ Клиентский обработчик

Введите не менее 10 чисел

Результат:

## ChislaProxy#view

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

Мы находимся по адресу: [http://127.0.0.1:3001/chisla\\_proxy/view.html.erb](http://127.0.0.1:3001/chisla_proxy/view.html.erb)

Former	All	The longest
3 4 5 6 7 8 9 10 11 12 -1 2 3	3 4 5 6 7 8 9 10 11 12	+
	-1 2 3	

[Рассчитать заново](#)

Sun Nov 19 2023 18:28:59 GMT+0300 (Москва, стандартное время)

### Рисунок 4 – результат работы клиентского обработчика и XSLT программы

←

↺

↻

🌐

127.0.0.1:3001

ChislaProxy

🔖

🔍

📄

⬇️

## ChislaProxy#input

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

Мы находимся по адресу: [http://127.0.0.1:3001/chisla\\_proxy/input.html.erb](http://127.0.0.1:3001/chisla_proxy/input.html.erb)

☐ Серверный обработчик ☒ Клиентский обработчик

Введите не менее 10 чисел

Результат: 

```
<?xml version="1.0"?> <?xml-stylesheet type="text/xsl" href="D:/education/3 semester/ipl/lab/lab10/mine/chisla-proxy/public/some_transformer.xslt"?> <catalog> <cd> <former>3
4 5 6 7 8 9 10 11 12 -1 2 3</former> <every>3 4 5 6 7 8 9 10 11 12</every> <plus>+</plus> </cd> <cd> <former> </former> <every>-1 2 3</every> <plus> </plus> </cd> </catalog>
```

Sun Nov 19 2023 18:28:40 GMT+0300 (Москва, стандартное время)

### Рисунок 5 – результат работы клиентского обработчика по выводу XML

```

Render HTML
<catalog><cd><former>1 2 3 4 5 6 7 8 9 10</former><every>1 2 3 4 5 6 7 8 9 10</every><plus>+</plus></cd></catalog>D:/education/3 semester/ipl/lab
/lab10/mine/chisla-proxy/public/some_transformer.xslt
Render XML
...Render HTML
...Render HTML
<catalog><cd><former>1 2 3 4 5 6 7 8 9 10</former><every>1 2 3 4 5 6 7 8 9 10</every><plus>+</plus></cd></catalog>D:/education/3 semester/ipl/lab
/lab10/mine/chisla-proxy/public/some_transformer.xslt
Render HTML
<catalog><cd><former>-1 -2 -3 2 3 4 -2 -3 2 3</former><every>-1</every><plus> </plus></cd><cd><former> </former><every>-2</every><plus> </plus></
cd><cd><former> </former><every>-3 2 3 4</every><plus>+</plus></cd><cd><former> </former><every>-2</every><plus> </plus></cd><cd><former> </forme
r><every>-3 2 3</every><plus> </plus></cd></catalog>D:/education/3 semester/ipl/lab/lab10/mine/chisla-proxy/public/some_transformer.xslt
.

Finished in 0.813116s, 9.8387 runs/s, 9.8387 assertions/s.
8 runs, 8 assertions, 0 failures, 0 errors, 0 skips

```

Рисунок 6 – результат работы тестов

**Вывод:** мы изучили формирования данных в формате XML и их визуализации с помощью клиентских и серверных средств с использованием XSLT-преобразований.