



**«Московский государственный технический университет  
имени Н.Э. Баумана»**

**(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 № 10**

**Дисциплина: Языки Интернет-программирования**

**Название лабораторной работы: Формирование и отображение XML в HTML  
средствами сервера и клиента. Вариант 10.**

Студент гр. ИУ6-34Б

\_\_\_\_\_  
(Подпись, дата)

К.Ю. Каташинский

\_\_\_\_\_  
(И.О. Фамилия)

Преподаватель

\_\_\_\_\_  
(Подпись, дата)

\_\_\_\_\_  
(И.О. Фамилия)

Москва, 2020

Условие задания:

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.

## Приложение laboratory

Код файла base\_controller.rb:

```
require 'test_helper'

class BaseController < ApplicationController
  def index; end

  def create
    number = params[:number].to_i, array = params[:str].split.map{|elem| elem.to_i}
    @error = number != array.length ? 'Введённые числа' : ''
    @all_segments = array.chunk_while do |first, second|
      multiply_five?(first) == multiply_five?(second)
    end.find_all {|segment| multiply_five?(segment[0]) }
    @has_elements = (@all_segments.length > 0), @count = @all_segments.length
    @max_segment = @all_segments.max_by { |segment| segment.length }
    @all_segments = @all_segments.join(" ")
    @max_segment = @max_segment&.join(" "), @array = array.join(" ")
    respond_to do |format|
      format.xml
    end
  end

  private def multiply_five?(elem)
    x = Math.log(elem, 5)
    elem != 1 ? (x.round == x) : false
  end
end
```

Код файла create.xml.erb:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="transformer.xslt"?>
<params>
  <% if @has_elements %>
    <array><%= @array %></array>
    <all_segments><%= @all_segments %></all_segments>
    <max_segment><%= @max_segment %></max_segment>
    <count><%= @count %></count>
  <% else %>
    <array><%= @array %></array>
    <all_segments></all_segments>
    <max_segment>Нен отрезков</max_segment>
    <count></count>
  <% end %>
</catalog>
```

## Приложение middle

Код файла generator\_controller.rb:

```
# frozen_string_literal: true
require 'net/http'
require 'uri'
require 'nokogiri'

# GeneratorController class
class GeneratorController < ApplicationController
  def index; end

  def create
    number = params[:number]
    array = params[:str]
    is_transformed_on_server = (params[:transformer] == 'server')
    uri = URI.parse('http://localhost:4000/create.xml')
    params = { number: number, str: array }
    uri.query = URI.encode_www_form(params)
    res = Net::HTTP.get_response(uri)
    respond_to do |format|
      if is_transformed_on_server
        doc = Nokogiri::XML(res.body)
        xslt = Nokogiri::XSLT(File.read('public/transformer.xslt'))
        f = File.new('app/views/generator/create.html.erb', 'w')
        f.write(xslt.transform(doc))
        f.close
        format.html
      else
        f = File.new('app/views/generator/create.xml.erb', 'w')
        f.write(Nokogiri::XML(res.body))
        f.close
        format.xml
      end
    end
  end
end
```

Код файла application.js:

```
handleAjaxSuccess = function(event) {
  [data, status, xhr] = event.detail;
  show_result(data);
}

document.addEventListener('DOMContentLoaded', function() {
  document.querySelector("#add").addEventListener(
```

Продолжение кода файла application.js:

```
'ajax:success', handleAjaxSuccess)
})
function show_result(data)
{
    var table = document.getElementById("add_table");
    if (document.getElementsByName("transformer")[0].checked)
        table.appendChild(data.getElementsByTagName("tr")[0]);
    else
    {
        var xhr = new XMLHttpRequest();
        xhr.open('GET', 'transformer.xslt', false);
        xhr.send();
        if (xhr.status == 200)
        {
            transformer = xhr.responseXML;
            processor = new XSLTProcessor();
            processor.importStylesheet(transformer);
            resultDocument = processor.transformToFragment(data, document);
            table.appendChild(resultDocument.children[2].getElementsByTagName("tr")[0]);
        }
        else
            alert("Произошла ошибка!")
    }
}
```

Код файла transformer.xslt:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
    <html>
    <head>
    <style>
        td {
            font-size: 15pt;
        }
        #add_table {
            width: 70%;
            border: 1px solid black;
            margin: auto;
            border-collapse: collapse;
        }
    </style>
    <table border="1">
        <tr>
            <td>
                <xsl:copy-of select="/*/*/*" />
            </td>
        </tr>
    </table>
    </html>
</template>
</xsl:stylesheet>
```

Продолжение кода файла transformer.xslt:

```
.col {
    border: 1px solid black;
    width: 50%;
    text-align:center;
}
</style>
</head>
<body>
    <table id="add_table">
        <caption><h1>Таблица вывода</h1></caption>
        <tbody>
            <tr>
                <xsl:choose>
                    <xsl:when test="params/max_segment != 'Нем отрезков'">
                        <td class="col"><xsl:value-of
select="params/array"/></td>
                        <td class="col"><xsl:value-of
select="params/all_segments"/></td>
                        <td class="col"><xsl:value-of
select="params/max_segment"/></td>
                        <td class="col"><xsl:value-of
select="params/count"/></td>
                    </xsl:when>
                    <xsl:otherwise>
                        <td class="col"><xsl:value-of
select="params/array"/></td>
                        <td colspan="3" id="error_cell">Нем отрезков</td>
                    </xsl:otherwise>
                </xsl:choose>
            </tr>
        </tbody>
    </table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Код файла index.html.erb:

```
<%= form_with(url: "/create", method: "post", id: "add") do %>
  <table id="add_form">
    <caption><h1>Форма ввода</h1></caption>
    <tr>
      <td width="40%">Введите число n</td>
      <td><%= text_field_tag :number, nil, class: "enter" %></td>
    </tr>
    <tr>
      <td>Введите n чисел через пробел</td>
      <td><%= text_field_tag :str, nil, class: "enter" %></td>
    </tr>
    <tr>
      <td>Выберите место преобразования</td>
      <td>
        Сервер<%= radio_button_tag(:transformer, "server", :checked) %>
        Клиент<%= radio_button_tag(:transformer, "client") %>
      </td>
    </tr>
    <tr>
      <td><%= submit_tag "Найти", class: "btn" %></td>
      <td><span style="color: red;"></span></td>
    </tr>
  </table>
<% end %>
<table id="add_table">
  <caption><h1>Таблица вывода</h1></caption>
  <tr>
    <td class="col">Входные данные</td>
    <td class="col">Полученные отрезки</td>
    <td class="col">Максимальный отрезок</td>
    <td class="col">Количество</td>
  </tr>
</table>
```

The screenshot shows a web browser window with the address bar displaying 'localhost:3000'. The page contains a form titled 'Форма ввода' (Input Form) and a table titled 'Таблица вывода' (Output Table).

**Форма ввода**

Введите число n:

Введите n чисел через пробел:

Выберите место преобразования: Сервер ☒ Клиент ☐

**Таблица вывода**

Входные данные	Полученные отрезки	Максимальный отрезок	Количество
1 2 3 4 5	5	5	1

Below the table, there is a light blue button labeled 'Вывод'.

Рисунок 1 «Результаты выполнения программы»

### Вывод

В ходе данной лабораторной работы был изучен язык XML и его основные способы преобразования в HTML. На основе полученных навыков был создан механизм преобразования XML в HTML с помощью XSLT на стороне сервера и клиента.