|  |  |
| --- | --- |
| 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 **09.03.01 Информатика и вычислительная техника**

**Отчет**

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

**Дисциплина:** Языки интернет-программирования

**Вариант:** 23

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-33Б |  |  | Д. И. Мироненко |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | В. Д. Шульман |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

**Задание**

Модифицировать код Практикума 5 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
* 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

**Код программы**

**Файл “client/app/controllers/index\_controller.rb”**

|  |
| --- |
| # frozen\_string\_literal: true  require 'net/http'  XML\_SERVER = 'http://localhost:3000/?format=xml'  XSL\_LINK = "\n<?xml-stylesheet href=\"/visualizer.xsl\" type=\"text/xsl\"?>\n"  XSLT = Nokogiri::XSLT(File.read('public/visualizer.xsl').gsub(%r{<rm>(.\*)</rm>}, '<rm>server-side</rm>'))  class IndexController < ApplicationController  def input; end  def get\_xml(array)  uri = URI(XML\_SERVER)  uri.query = URI.encode\_www\_form({ array: array })  Net::HTTP.get\_response(uri)  end  def render\_client(xml)  xml.insert(xml.index("\n"), XSL\_LINK)  end  def render\_server(xml)  # rubocop:disable Rails/OutputSafety  XSLT.transform(Nokogiri::XML(xml))  .to\_s  .html\_safe  # rubocop:enable Rails/OutputSafety  end  def output  xml = get\_xml(params[:array]).body  respond\_to do |format|  # format.xml { render xml: render\_client(xml) }  format.html { render html: render\_server(xml) }  end  end  end |

**Файл “client/app/views/index/input.html.erb”**

|  |
| --- |
| <form action="<%= index\_output\_url %>">  <label>Array</label>  <br>  <input name="array" value="5 8 4 9 57 4 8 38">  <br><br>  <input type="submit" value="Get HTML">  </form> |

**Файл “client/public/visualizer.xsl”**

|  |
| --- |
| <?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>  <title>XML Visualiser</title>  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous"/>  </head>  <body>  <h1>Result</h1>  <table class="table">  <tbody id='result-table'>  <xsl:for-each select="objects/object">  <tr>  <td><xsl:value-of select="text"/></td>  <td><xsl:value-of select="sequence"/></td>  </tr>  </xsl:for-each>  </tbody>  </table>  </body>  </html>  </xsl:template>  </xsl:stylesheet> |

**Файл “server/app/controllers/index\_controller.rb”**

|  |
| --- |
| # frozen\_string\_literal: true  # This shiny device polishes bared foos  class IndexController < ApplicationController  def view  input = params[:array]  @result = []  @longest\_sequence = []  @sequences = []  raise ArgumentError unless input&.match?(/\A\s\*\d+(\s+\d+)\*\s\*\z/)  array = input.split  start = 0  max\_len = 0  curr\_len = 1  array.length.times do |i|  next if i.zero?  if array[i - 1].to\_i <= array[i].to\_i  curr\_len += 1  else  @sequences << array[start, curr\_len] if curr\_len > 1  if curr\_len > max\_len  @longest\_sequence = @sequences[-1]  max\_len = curr\_len  end  start = i  curr\_len = 1  end  end  @sequences << array[start, curr\_len] if curr\_len > 1  if curr\_len > max\_len  @longest\_sequence = @sequences[-1]  max\_len = curr\_len  end  @sequences.length.times do |i|  @result << { text: "Sequence #{i + 1}", sequence: @sequences[i].join(' ') }  end  @result << { text: "Longest Sequence", sequence: @longest\_sequence.join(' ') }  respond\_to do |format|  format.xml { render xml: @result }  format.rss { render xml: @result }  end  rescue ArgumentError  @error = 'Invalid input!'  end  end |

**Работа программы**

A screenshot of a computer

Description automatically generated

**Рисунок 1** – ввод значений

**A white background with black lines

Description automatically generated**

**Рисунок 2** – вывод результатов (XML, преобразованный в HTML с помощью шаблона XSLT)

**A black and white text

Description automatically generated with medium confidence**

**Рисунок 3** – результат проверки rubocop’ом

**A screenshot of a computer

Description automatically generated**

**Рисунок 4** – результат проверки rubocop’ом

**Вывод**

Изучена технология XSLT для преобразования XML в HTML при помощи шаблона XSL; полученные знания отработаны при создании простого web-сайта; исходный код проверен rubocop’ом.