



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

О Т Ч Е Т

по лабораторной работе № 1 0

Название:

Формирование и отображение XML в HTML средствами сервера
и клиента.

Дисциплина: Языки интернет программирования

Студент

ИУ6-31Б

(Группа)

(Подпись, дата)

Т.Е. Старжевский

(И.О. Фамилия)

Преподаватель

(Подпись, дата)

Д. В. Малахов

(И.О. Фамилия)

Москва, 2022

Задание состоит из 2 приложений, выполнять будем поэтапно.

Текст задания 1 части:

Модифицировать код ЛР 8 таким образом, чтобы по запросу с указанными параметрами выдавался результат в формате XML (средствами стандартной сериализации ActiveSupport).

- 1) Проверить формирование XML и сохранить в файл для отладки XSLT и второго приложения.
- 2) Написать функциональный тест, проверяющий формат выдаваемых данных при запросе RSS.
- 3) Разработать XSLT-программу преобразования полученной XML в HTML.
- 4) Добавить в проверяемый XML-файл строку привязки к преобразованию `<?xml-stylesheet type="text/xsl" href="some_transformer.xslt"?>`. Проверить корректность отображения браузером результата преобразования.
- 5) 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)
```

Последовательность выполненных команд:

```
rails new xml_api -T
rails g controller render_xml
```

После чего удалим представление из `xml_api/app/views/render_xml` так как оно нам не понадобится.

Добавим логики в контроллер:

`xml_api/app/controllers/render_xml_controller.rb`

```
# frozen_string_literal: true

# Docs
class RenderXmlController < ApplicationController
  def show
    result = palindroms_before params[:n].to_i

    data = if result.empty?
```

```

{ message: "Error: incorrect params #{params[:n]}" }
else
result.map { |el| { palindrom: el, square: el**2 } }
end

respond_to do |format|
format.xml { render xml: data.to_xml }
format.rss { render xml: data.to_xml }
end
end

private

def palindroms_before(number)
[*1..number].select { |el| el.to_s.reverse == el.to_s && (el**2).to_s.reverse ==
(el**2).to_s }
end
end

```

xml_api/config/routes.rb

```

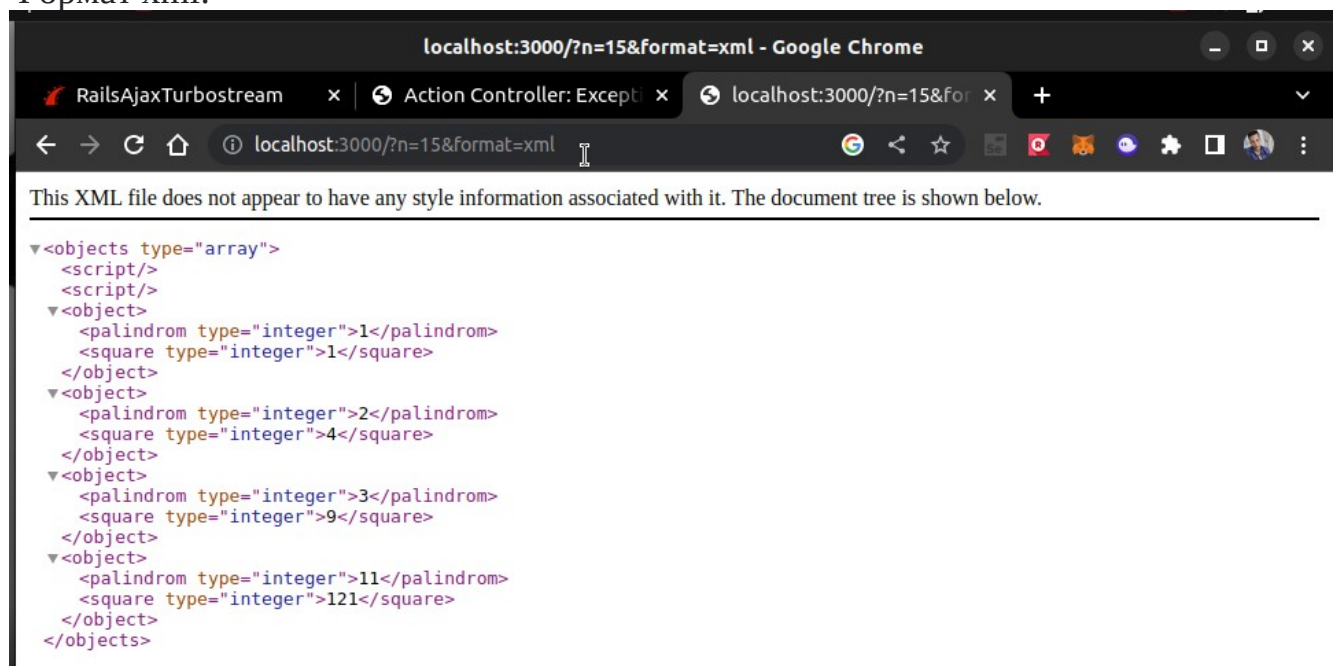
Rails.application.routes.draw do
# Define your application routes per the DSL in https://guides.rubyonrails.org/routing.html
root "render_xml#show"
# Defines the root path route ("/")
# root "articles#index"
end

```

Запустим приложение:

rails s

Формат xml:



Формат rss:

```
localhost:3000/?n=15&format=rss - Google Chrome
RailsAjaxTurbostream x | Action Controller: Excepti x | localhost:3000/?n=15&for x
localhost:3000/?n=15&format=rss
<?xml version="1.0" encoding="UTF-8"?>
<objects type="array">
  <object>
    <palindrom type="integer">1</palindrom>
    <square type="integer">1</square>
  </object>
  <object>
    <palindrom type="integer">2</palindrom>
    <square type="integer">4</square>
  </object>
  <object>
    <palindrom type="integer">3</palindrom>
    <square type="integer">9</square>
  </object>
  <object>
    <palindrom type="integer">11</palindrom>
    <square type="integer">121</square>
  </object>
</objects>
```

XML & XSLT Преобразования

Скопируем из браузера код xml:

xml_api/response.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<objects type="array">
  <object>
    <palindrom type="integer">1</palindrom>
    <square type="integer">1</square>
  </object>
  <object>
    <palindrom type="integer">2</palindrom>
    <square type="integer">4</square>
  </object>
  <object>
    <palindrom type="integer">3</palindrom>
    <square type="integer">9</square>
  </object>
  <object>
    <palindrom type="integer">11</palindrom>
    <square type="integer">121</square>
  </object>
  <object>
    <palindrom type="integer">22</palindrom>
    <square type="integer">484</square>
  </object>
  <object>
    <palindrom type="integer">101</palindrom>
    <square type="integer">10201</square>
  </object>
  <object>
    <palindrom type="integer">111</palindrom>
    <square type="integer">12321</square>
  </object>
</objects>
```

```

</object>
<object>
<palindrom type="integer">121</palindrom>
<square type="integer">14641</square>
</object>
</objects>

```

Создадим файл трансформации

xml_api/transform.xslt

```

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

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <!--xsl:template говорит о том, что тут будет замена. match показывает, к какой части
  документа это применимо-->
  <xsl:template match="/">
    <!--Внутри шаблона пишем наше преобразование-->

    <html>
    <head>
    <title>Response</title>
    </head>
    <body>
    <table>
    <thead>
    <tr>
    <th>#</th>
    <th>Палиндром</th>
    <th>Квадрат</th>
    </tr>
    </thead>
    <tbody>
    <!--Цикл-->
    <xsl:for-each select="objects/object">
    <!--Создание переменной-->
    <xsl:variable name="counter" select="position()"/>
    <tr>
    <th>
    <!--Извлекаем значение из переменной (обратите внимание на $)-->
    <xsl:value-of select="$counter"></xsl:value-of>
    </th>
    <th>
    <!--Извлекаем значение из XML-тега-->
    <xsl:value-of select="palindrom"></xsl:value-of>
    </th>
    <th>
    <xsl:value-of select="square"></xsl:value-of>
    </th>
    </tr>
    </xsl:for-each>
    </tbody>

```

```
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Добавим строку привязки файла к его трансформации, для дальнейшей логики задания создадим другой файл:

xml_api/response_with_xslt.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="transform.xslt"?>
<objects type="array">
  <object>
    <palindrom type="integer">1</palindrom>
    <square type="integer">1</square>
  </object>
  <object>
    <palindrom type="integer">2</palindrom>
    <square type="integer">4</square>
  </object>
  <object>
    <palindrom type="integer">3</palindrom>
    <square type="integer">9</square>
  </object>
  <object>
    <palindrom type="integer">11</palindrom>
    <square type="integer">121</square>
  </object>
  <object>
    <palindrom type="integer">22</palindrom>
    <square type="integer">484</square>
  </object>
  <object>
    <palindrom type="integer">101</palindrom>
    <square type="integer">10201</square>
  </object>
  <object>
    <palindrom type="integer">111</palindrom>
    <square type="integer">12321</square>
  </object>
  <object>
    <palindrom type="integer">121</palindrom>
    <square type="integer">14641</square>
  </object>
</objects>
```

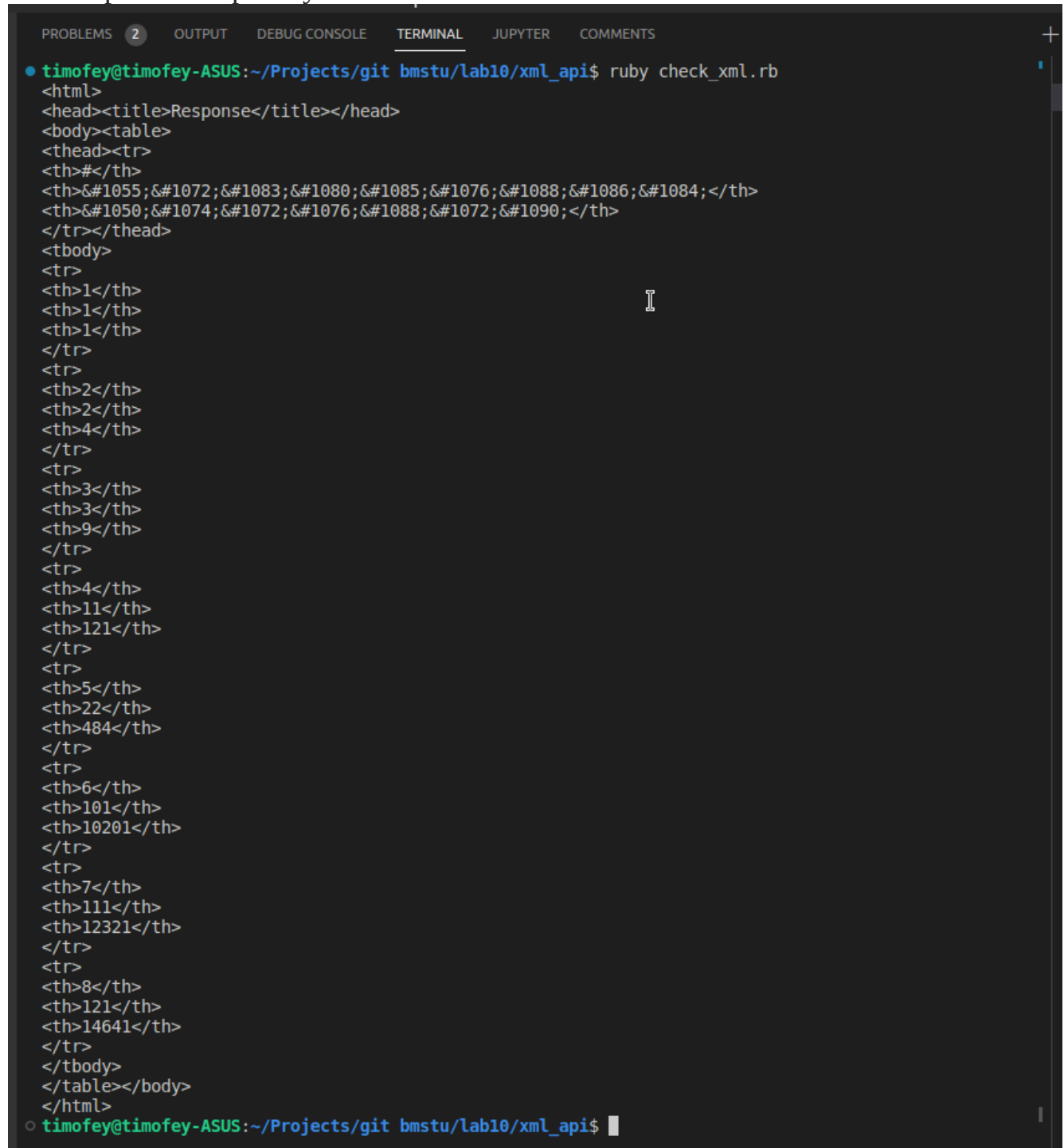
К сожалению ни один из моих браузеров (Chrome, firefox, yandex) в целях безопасности не подгружают файл трансформации для xml, но мы можем протестировать корректность создав программу:

xml_api/check_xml.rb

```
require 'nokogiri'

if $PROGRAM_NAME == __FILE__
  doc = Nokogiri::XML(File.read('response.xml'))
  xslt = Nokogiri::XSLT(File.read('transform.xslt'))
  puts xslt.transform(doc)
end
```

Посмотрим на ее работу:



```
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER COMMENTS
• timofey@timofey-ASUS:~/Projects/git bmstu/lab10/xml_api$ ruby check_xml.rb
<html>
<head><title>Response</title></head>
<body><table>
<thead><tr>
<th>#</th>
<th>№1055;№1072;№1083;№1080;№1085;№1076;№1088;№1086;№1084;</th>
<th>№1050;№1074;№1072;№1076;№1088;№1072;№1090;</th>
</tr></thead>
<tbody>
<tr>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
<tr>
<th>2</th>
<th>2</th>
<th>4</th>
</tr>
<tr>
<th>3</th>
<th>3</th>
<th>9</th>
</tr>
<tr>
<th>4</th>
<th>11</th>
<th>121</th>
</tr>
<tr>
<th>5</th>
<th>22</th>
<th>484</th>
</tr>
<tr>
<th>6</th>
<th>101</th>
<th>10201</th>
</tr>
<tr>
<th>7</th>
<th>111</th>
<th>12321</th>
</tr>
<tr>
<th>8</th>
<th>121</th>
<th>14641</th>
</tr>
</tbody>
</table></body>
</html>
○ timofey@timofey-ASUS:~/Projects/git bmstu/lab10/xml_api$
```

Как видим, преобразование происходит корректно.

Тестирование

Создаем тесты, предварительно добавив соответствующие гемы в Gemfile:

```
group :test do
  gem 'rspec-rails'
end
```

Выполним команды:

```
rails g rspec install
rails g rspec:controller render_xml
```

Добавим логику в тест:

xml_api/spec/requests/render_xml_spec.rb

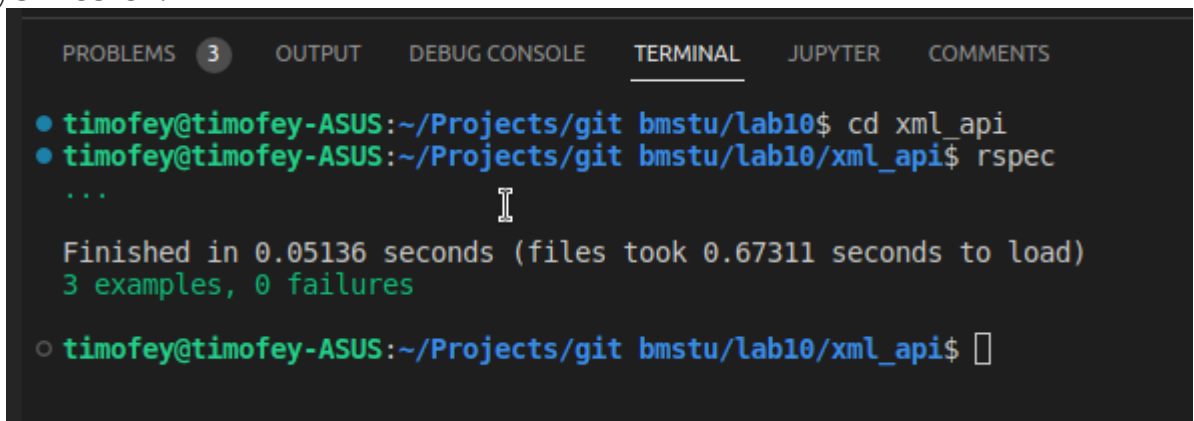
```
require 'rails_helper'

RSpec.describe 'RenderXmls', type: :request do
  describe 'GET /' do
    context 'should return' do
      it 'http success' do
        get root_path, params: { n: 15, format: :xml }
        expect(response).to have_http_status(:success)
      end

      it 'xml format' do
        get root_path, params: { n: 45, format: :xml }
        expect(response.headers['Content-Type']).to eq("application/xml; charset=utf-8")
      end

      it 'rss format' do
        get root_path, params: { n: 145, format: :rss }
        expect(response.headers['Content-Type']).to eq("application/rss+xml; charset=utf-8")
      end
    end
  end
end
```

Запуск тестов:



```
PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER COMMENTS
• timofey@timofey-ASUS:~/Projects/git bmstu/lab10$ cd xml_api
• timofey@timofey-ASUS:~/Projects/git bmstu/lab10/xml_api$ rspec
...
Finished in 0.05136 seconds (files took 0.67311 seconds to load)
3 examples, 0 failures
• timofey@timofey-ASUS:~/Projects/git bmstu/lab10/xml_api$
```

Далее перейдем ко второй части задания

Текст задания 2 части:

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

Создание и наполнение второго приложения

```
rails new Rails_with_xml_proxy -T
rails g controller palindrom input show
```

Rails_with_xml_proxy/app/controllers/palindrom_controller.rb

```
# frozen_string_literal: true

require 'nokogiri'
require 'open-uri'

# Documentation
class PalindromController < ApplicationController
  before_action :parse_params, only: :show
  before_action :require_params, only: :show
  before_action :prepare_url, only: :show

  def input; end

  def show
    api_response = URI.open(@url)

    case @side
    when 'On server'
      @result = xslt_server_transform(api_response).to_html
    when 'On client'
      render xml: xslt_browser_transform(api_response).to_xml
    else
      render xml: api_response
    end
  end

  private

  # Куда шлем запрос.
  BASE_API_URL = 'http://localhost:3000/?format=xml'
  # Откуда берем XSLT для преобразования на стороне сервера
  # (тут нужен обычный путь, Rails.root - путь к каталгу приложения).
  XSLT_SERVER_TRANSFORM = "#{Rails.root}/public/server_transform.xslt".freeze
  # Откуда браузер должен брать XSLT. Это подставится к localhost:3001. Именно так
  # грузятся файлы из public.
  XSLT_BROWSER_TRANSFORM = '/browser_transform.xslt'

  def parse_params
    @upper = params[:n]
    @side = params[:side]
  end

  def require_params
    if @upper.nil? || @upper.empty?
      flash[:error] = 'Error: Empty params'
      redirect_to root_path
    elsif !@upper.match(/^\\d+$/)
      flash[:error] = "Error: Incorrect params'#{@upper}"
      redirect_to root_path
    end
  end
end
```

```

def prepare_url
  @url = "#{BASE_API_URL}&n=#{@upper}"
end

def xslt_server_transform(data)
  doc = Nokogiri::XML(data)
  xslt = Nokogiri::XSLT(File.read(XSLT_SERVER_TRANSFORM))
  xslt.transform(doc)
end

def xslt_browser_transform(data)
  doc = Nokogiri::XML(data)
  xslt = Nokogiri::XML::ProcessingInstruction.new(doc,
    'xml-stylesheet',
    "type=\"text/xsl\" href=\"#{XSLT_BROWSER_TRANSFORM}\"")
  doc.root.add_previous_sibling(xslt)
  doc
end
end

```

Rails_with_xml_proxy/app/views/palindrom/input.html.erb

```

<h1 class="color-text center">Palindroms</h1>
<p class="center">For example 202 and 202 ** 2 = 40804</p>
<div class="container">
  <%= form_with url: "/palindrom/show", method: :get do |form| %>
    <div class="input">
      <%= form.label :n, "Input end of palindrom`s range:"%>
      <%= form.text_field :n, class: "text-field", value: 5262 %>
    </div>
    <div class="checkbox">
      <%= form.radio_button :side, "Blank xml"%>
      <%= form.label :xml, "Blank xml" %>
    </div>
    <div class="checkbox">
      <%= form.radio_button :side, "On server"%>
      <%= form.label :xslt_server, "On server" %>
    </div>
    <div class="checkbox">
      <%= form.radio_button :side, "On client" %>
      <%= form.label :xslt_client, "On client" %>
    </div>
    <div class="center">
      <%= form.submit "Show", class:"button" %>
    </div>
  <% end %>
</div>
<% unless flash[:error].nil?%>
<div class="error-text"> <%= flash[:error] %> </div>
<% end %>

```

Rails_with_xml_proxy/app/views/palindrom/show.html.erb

```
<h1 class="color-text center"> Palindroms from 1 to <%= params[:n] %> </h1>
<br><br>
<%= render inline: @result %>
```

Rails_with_xml_proxy/config/routes.rb

```
Rails.application.routes.draw do
  root "palindrom#input"
  get 'palindrom/show'
  # Define your application routes per the DSL in https://guides.rubyonrails.org/routing.html

  # Defines the root path route ("/")
  # root "articles#index"
end
```

Rails_with_xml_proxy/app/assets/stylesheets/application.css

```
/*
 * This is a manifest file that'll be compiled into application.css, which will include all the files
 * listed below.
 *
 * Any CSS (and SCSS, if configured) file within this directory, lib/assets/stylesheets, or any
 * plugin's
 * vendor/assets/stylesheets directory can be referenced here using a relative path.
 *
 * You're free to add application-wide styles to this file and they'll appear at the bottom of the
 * compiled file so the styles you add here take precedence over styles defined in any other
 * CSS
 * files in this directory. Styles in this file should be added after the last require_* statement.
 * It is generally better to create a new file per style scope.
 */
*= require_tree .
*= require_self
*/
body {
  background: #000;
  color: #fff;
  font-family: system-ui, -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen,
  Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue', sans-serif;
}
.color-text {
  color: #fc4f83;
}
.center {
  display: flex;
  align-items: center;
  justify-content: center;
}
table {
  font-size: 20px;
```

```
color: #fff;
text-align: center;
}
table tr {
border-bottom: 1px solid #444;
}
.text-field {
margin: 0px 10px;
padding: 2px 6px;
width: 150px;
border-radius: 10px;
font-size: 20px;
font-weight: 800;
outline: none;
}
.button {
position: relative;
text-align: center;
border-radius: 10px;
padding: 5px 6px;
font-size: 18px;
font-weight: 800;
transition: 0.5s;

margin: 0px 10px;
color: #cecece;
border: 2px solid #cecece;
background: #000;
}
.button:hover {
cursor: pointer;
color: #fff;
border: 2px solid #fff;
}

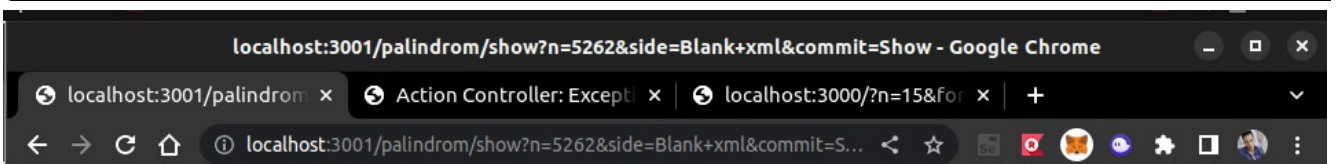
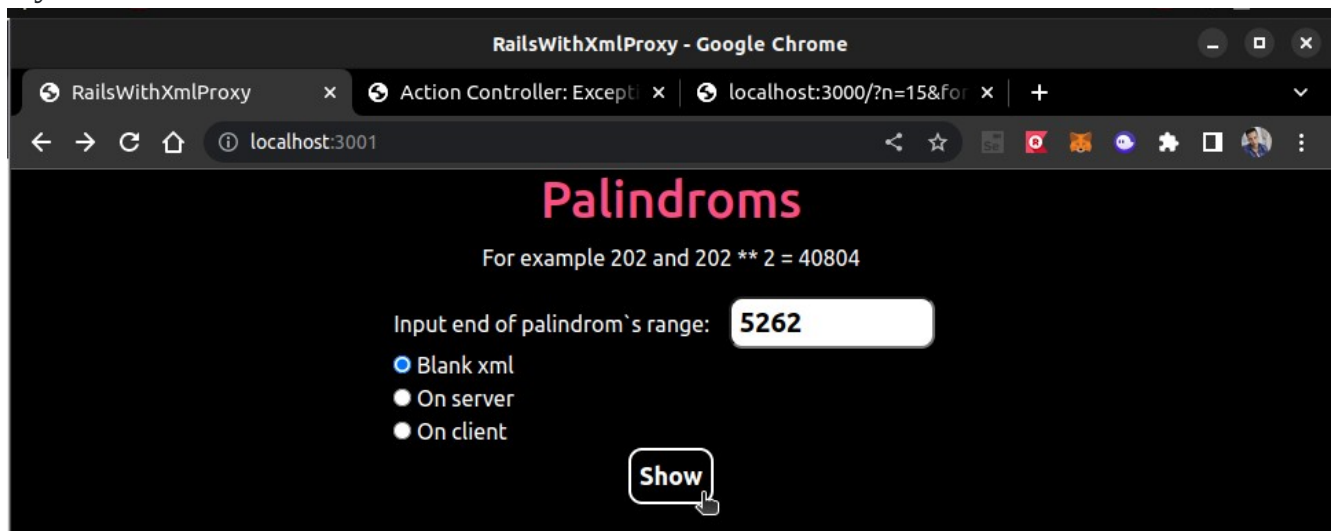
.container {
display: flex;
justify-content: space-around;
align-items: center;
}

.error-text {
display: flex;
justify-content: center;
align-items: center;
font-size: 20px;
color: red;
}
```

Запустим приложение одновременно с первым, только на другом порту:

```
rails s -p 3001
```

Пустой xml:



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<?xml version="1.0"?>
<objects type="array">
  <script/>
  <script/>
  <object>
    <palindrom type="integer">1</palindrom>
    <square type="integer">1</square>
  </object>
  <object>
    <palindrom type="integer">2</palindrom>
    <square type="integer">4</square>
  </object>
  <object>
    <palindrom type="integer">3</palindrom>
    <square type="integer">9</square>
  </object>
  <object>
    <palindrom type="integer">11</palindrom>
    <square type="integer">121</square>
  </object>
  <object>
    <palindrom type="integer">22</palindrom>
    <square type="integer">484</square>
  </object>
  <object>
    <palindrom type="integer">101</palindrom>
    <square type="integer">10201</square>
  </object>
  <object>
    <palindrom type="integer">111</palindrom>
    <square type="integer">12321</square>
  </object>
  <object>
    <palindrom type="integer">121</palindrom>
    <square type="integer">14641</square>
  </object>
  <object>
    <palindrom type="integer">202</palindrom>
    <square type="integer">40804</square>
  </object>
  <object>
    <palindrom type="integer">212</palindrom>
    <square type="integer">44944</square>
  </object>
  <object>
    <palindrom type="integer">1001</palindrom>
    <square type="integer">1002001</square>
  </object>
  <object>
    <palindrom type="integer">1111</palindrom>
    <square type="integer">1234321</square>
  </object>
  <object>
    <palindrom type="integer">2002</palindrom>
    <square type="integer">4008004</square>
  </object>
</objects>
```

Обработать на сервере:

RailsWithXmlProxy - Google Chrome

localhost:3001

Palindroms

For example 202 and 202 ** 2 = 40804

Input end of palindrom's range:

☐ Blank xml
☒ On server
☐ On client

Show

RailsWithXmlProxy - Google Chrome

localhost:3001/palindrom/show?n=526&side=On+server&commit=S...

Palindroms from 1 to 526

#	Палиндром	Квадрат
1	1	1
2	2	4
3	3	9
4	11	121
5	22	484
6	101	10201
7	111	12321
8	121	14641
9	202	40804
10	212	44944

Обработать на клиенте:

RailsWithXmlProxy - Google Chrome

localhost:3001

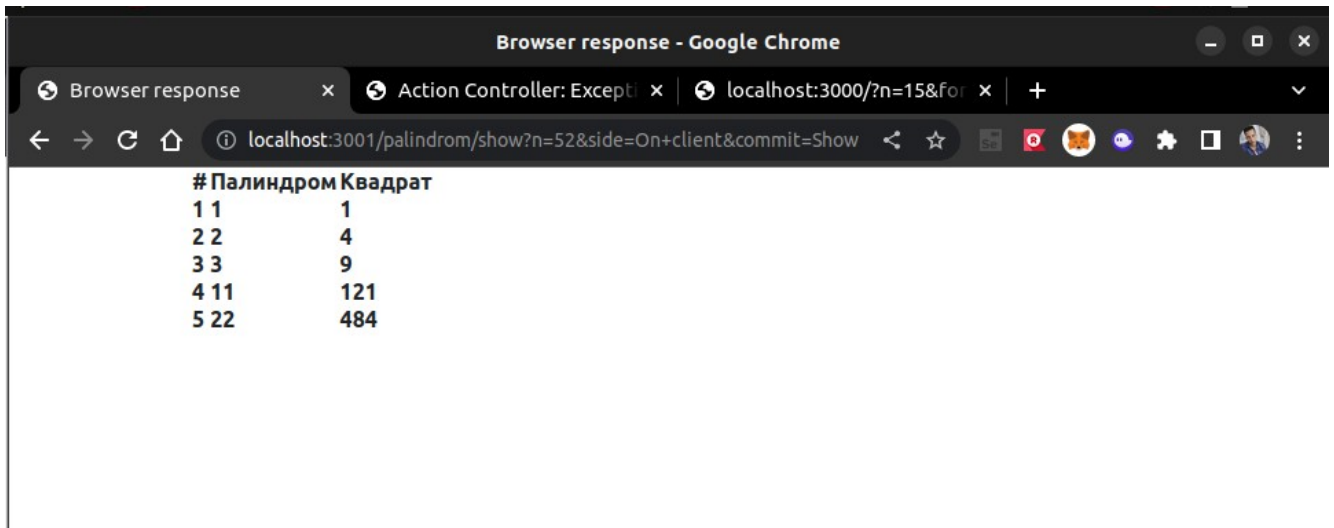
Palindroms

For example 202 and 202 ** 2 = 40804

Input end of palindrom's range:

☐ Blank xml
☐ On server
☒ On client

Show



Тестирование:

Добавим в Gemfile зависимости:

```
group :test do
  gem 'rspec-rails'
  gem 'selenium-webdriver'
end

gem 'nokogiri'
```

```
rails g rspec install
```

```
rails g rspec:controller palindrom
```

Rails_with_xml_proxy/spec/requests/palindrom_spec.rb

```
require 'rails_helper'
require 'selenium-webdriver'
require 'nokogiri'

RSpec.describe "Palindroms", type: :request do
  describe "Show" do
    before(:each) do
      @driver = Selenium::WebDriver.for :firefox
    end

    after(:each) do
      @driver.quit
    end

    context 'when params incorrected' do
      it 'should consist empty-message' do
        @driver.get('http://localhost:3001/')
        @driver.find_element(:id, 'n').click
        @driver.find_element(:id, 'n').clear
      end
    end
  end
end
```



```

@driver.find_element(:id, 'side_blank_xml').click
@driver.find_element(:name, 'commit').click
expect(@driver.find_element(:css, '.error-text').text).to eq('Error: Empty params')
end

it 'should consist incorrect-message' do
@driver.get('http://localhost:3001/')
@driver.find_element(:id, 'n').click
@driver.find_element(:id, 'n').send_keys('asd;')
@driver.find_element(:id, 'side_blank_xml').click
@driver.find_element(:name, 'commit').click
expect(@driver.find_element(:css, '.error-text').text).to eq('Error: Incorrect
params\'5262asd;\'')
end
end

context 'when send default params' do
it 'for get blank XML' do
@driver.get('http://localhost:3001/')
@driver.find_element(:id, 'side_blank_xml').click
@driver.find_element(:name, 'commit').click
xml = Nokogiri::XML(@driver.page_source)
text = xml.xpath('//objects/object[13]/square/text()').text
expect(text).to eq('4008004')
end

it 'On server' do
@driver.get('http://localhost:3001/')
@driver.find_element(:id, 'side_on_server').click
@driver.find_element(:name, 'commit').click
expect(@driver.find_element(:css, 'tr:nth-child(13) > th:nth-child(3)').text).to eq('4008004')
end

it 'On client' do
@driver.get('http://localhost:3001/')
@driver.find_element(:id, 'side_on_client').click
@driver.find_element(:name, 'commit').click
expect(@driver.find_element(:css, 'tr:nth-child(13) > th:nth-child(3)').text).to eq('4008004')
end
end

context 'when send params = 500' do
it 'for get blank XML' do
@driver.get('http://localhost:3001/')
@driver.find_element(:id, 'n').click
@driver.find_element(:id, 'n').clear
@driver.find_element(:id, 'n').send_keys('500')
@driver.find_element(:id, 'side_blank_xml').click
@driver.find_element(:name, 'commit').click
xml = Nokogiri::XML(@driver.page_source)
text = xml.xpath('//objects/object[10]/square/text()').text
expect(text).to eq('44944')
end
end

```

```

end

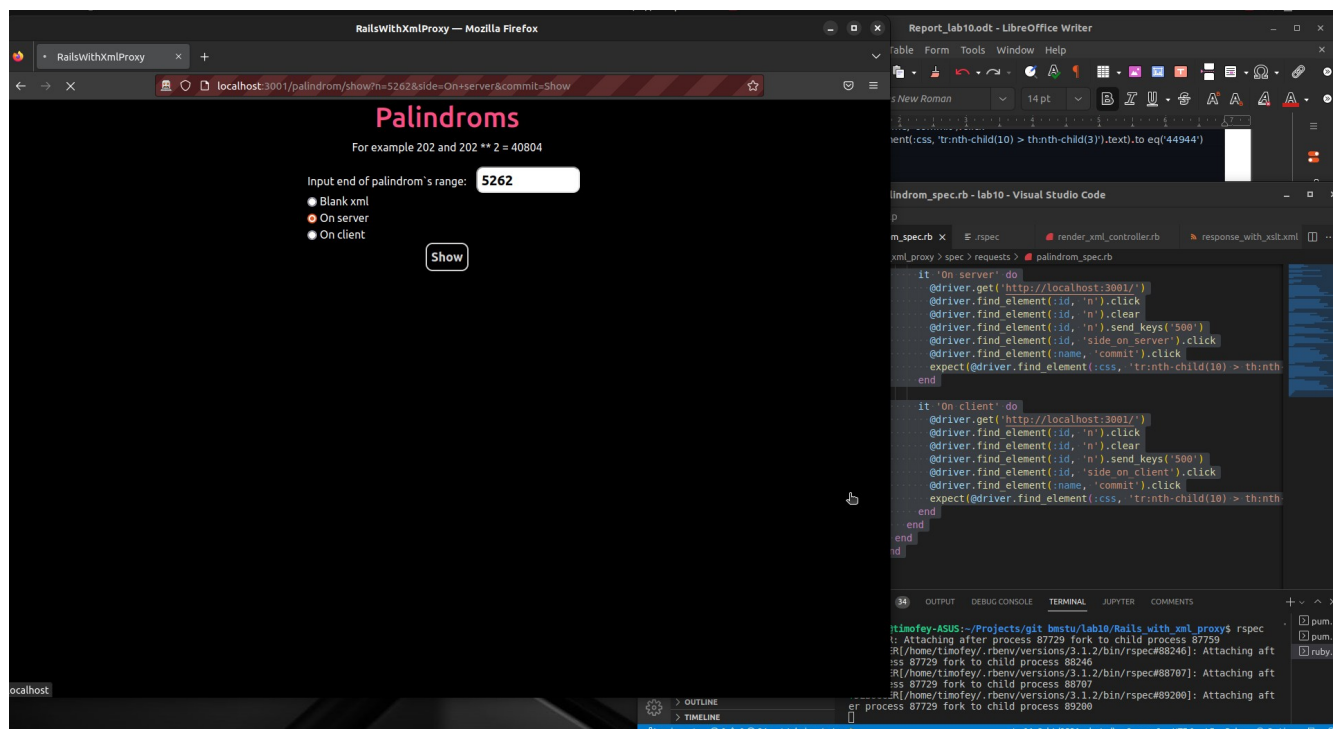
it 'On server' do
  @driver.get('http://localhost:3001/')
  @driver.find_element(:id, 'n').click
  @driver.find_element(:id, 'n').clear
  @driver.find_element(:id, 'n').send_keys('500')
  @driver.find_element(:id, 'side_on_server').click
  @driver.find_element(:name, 'commit').click
  expect(@driver.find_element(:css, 'tr:nth-child(10) > th:nth-child(3)').text).to eq('44944')
end

it 'On client' do
  @driver.get('http://localhost:3001/')
  @driver.find_element(:id, 'n').click
  @driver.find_element(:id, 'n').clear
  @driver.find_element(:id, 'n').send_keys('500')
  @driver.find_element(:id, 'side_on_client').click
  @driver.find_element(:name, 'commit').click
  expect(@driver.find_element(:css, 'tr:nth-child(10) > th:nth-child(3)').text).to eq('44944')
end
end
end
end

```

Запустим тесты одновременно с работающими приложениями на разных портах:

rspec



```

.DEBUGGER[/home/timofey/.rbenv/versions/3.1.2/bin/rspec#91156]: Attac
hing after process 87729 fork to child process 91156

```

```

Finished in 41.06 seconds (files took 0.75733 seconds to load)
8 examples, 0 failures

```

В тестах использовались Selenium, Nokogiri

Вывод: Научился формировать отображения xml используя xslt файлы, организовывать это в логике Rails приложения, запускать одновременно несколько веб-приложений и понял различие в обработке данных на клиенте и сервере или отдачи их в необработанном виде.