*Федеральное государственное бюджетное образовательное учреждение* *высшего профессионального образования*

|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | ***«Московский государственный технический университет  имени Н.Э. Баумана»***  ***(МГТУ им. Н.Э. Баумана)*** |

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Компьютерные системы и сети

**Отчет**

**по лабораторной работе № 10**

**Дисциплина: Языки интернет-программирования**

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

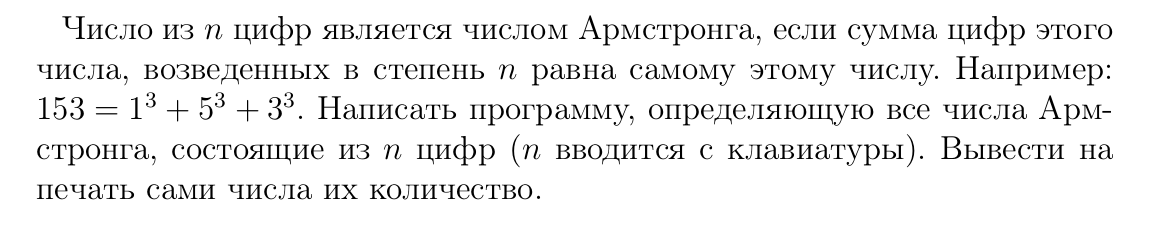
Студент гр. \_\_ИУ6-34 **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лавренов И.С.**

(Подпись, дата) (И.О. Фамилия)

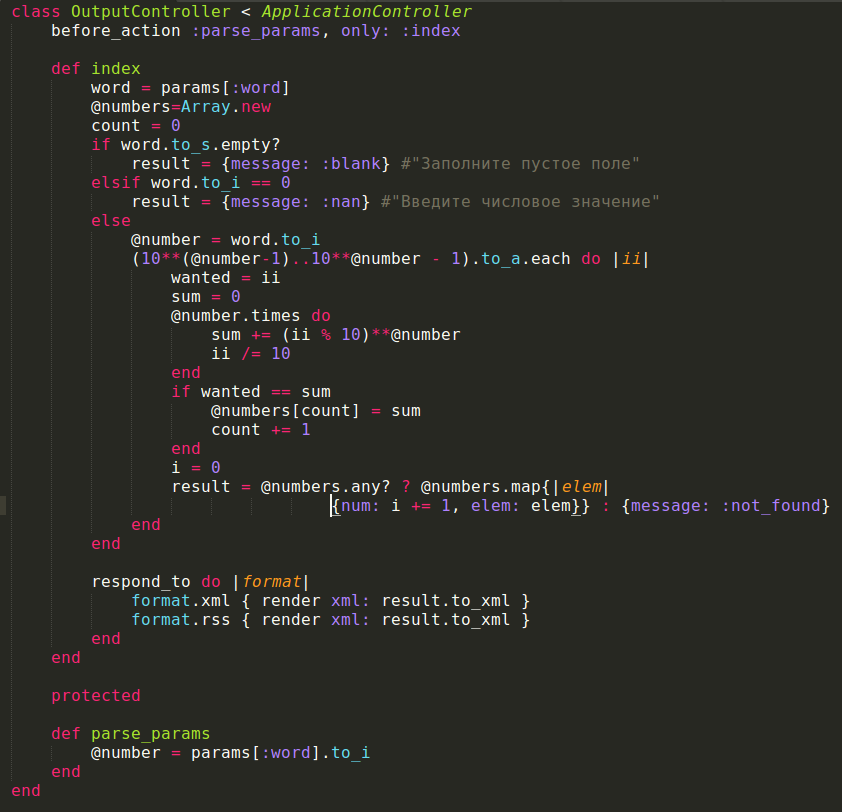
Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О. Фамилия)

Москва, 2017

Задание

Код контроллера 1 приложения:



Тестирование 1 приложения:

require 'test\_helper'

class OutputControllerTest < ActionDispatch::IntegrationTest

# test "the truth" do

# assert true

# end

test 'check\_rss\_format' do

get '/', params: { word: '3', format: :rss }

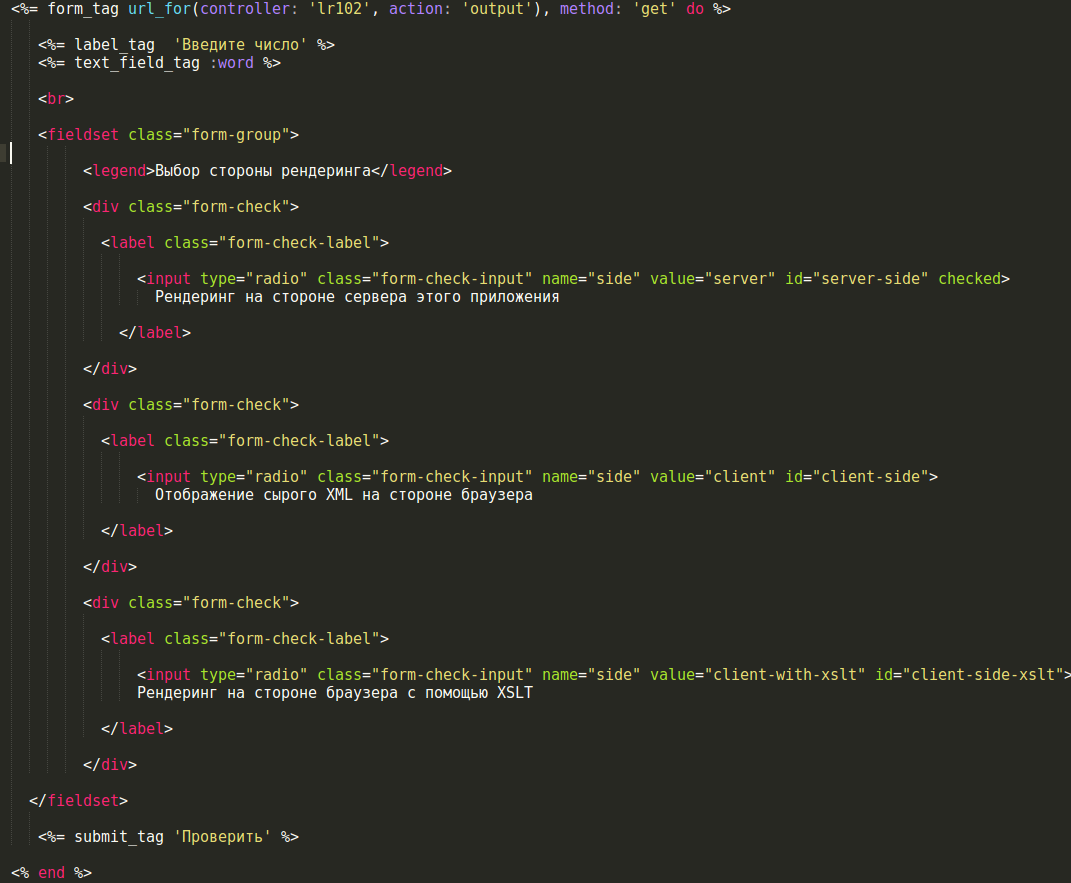
assert\_response :success

assert\_includes @response.headers['Content-Type'], 'application/rss'#тип ответа

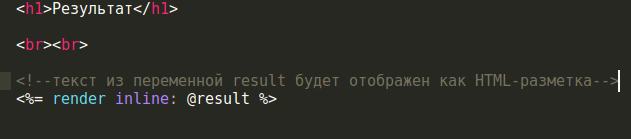
end

end

Предствавление для ввода 2 приложения:



Представление вывода:



Контроллер 2 приложения:

require 'nokogiri'

require 'open-uri'

class Lr102Controller < ApplicationController

before\_action :parse\_params, only: :output

before\_action :prepare\_url, only: :output

def input

end

def output

output\_response = open(@url)

if @side == 'server'

@result = xslt\_transform(output\_response).to\_html

# Если делать XML -> HTML в браузере.

elsif @side == 'client-with-xslt'

render xml: insert\_browser\_xslt(output\_response).to\_xml

else

render xml: output\_response # в неизменном виде

end

end

private

BASE\_OUTPUT\_URL = 'http://localhost:3000/?format=xml'.freeze

# Откуда берется XSLT для преобразования на стороне сервера

XSLT\_SERVER\_TRANSFORM = "#{Rails.root}/public/server-transform.xslt".freeze

# Откуда браузер должен брать XSLT

XSLT\_BROWSER\_TRANSFORM = '/browser-transform.xslt'.freeze

def parse\_params

@number = params[:word].to\_i

@side = params[:side]

end

# Создается url с использованием разобранных параметров запроса

def prepare\_url

@url = BASE\_OUTPUT\_URL + "&word=#{@number}"

end

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

doc = Nokogiri::XML(data)

xslt = Nokogiri::XSLT(File.read(transform))

xslt.transform(doc)

end

def insert\_browser\_xslt(data, transform: XSLT\_BROWSER\_TRANSFORM)

doc = Nokogiri::XML(data)

xslt = Nokogiri::XML::ProcessingInstruction.new(doc, 'xml-stylesheet', 'type="text/xsl" href="' + transform + '"')

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

doc

end

end

Отображение xml на сервере:

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

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

<xsl:template match='/'>

<xsl:choose>

<xsl:when test="/\*/message">

<h2>

<xsl:value-of select="."/>

</h2>

</xsl:when>

<xsl:otherwise>

<table>

<tr>

<th>

№

</th>

<th>

Число Армстронга

</th>

</tr>

<xsl:for-each select='objects/object'>

<tr>

<td>

<xsl:value-of select='num'/>

</td>

<td>

<xsl:value-of select='elem'/>

</td>

</tr>

</xsl:for-each>

</table>

</xsl:otherwise>

</xsl:choose>

</xsl:template>

</xsl:stylesheet>

Отобоажение xml браузером:

<?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>

Ответ сервера

</title>

<style>

table{

border: 1px solid black;

}

td{

border: 1px solid black;

text-align: center;

}

</style>

</head>

<body>

<xsl:choose>

<xsl:when test="/\*/message">

<h2>

<xsl:value-of select="."/>

</h2>

</xsl:when>

<xsl:otherwise>

<table>

<tr>

<th>

№

</th>

<th>

Число Армстронга

</th>

</tr>

<xsl:for-each select='objects/object'>

<tr>

<td>

<xsl:value-of select='num'/>

</td>

<td>

<xsl:value-of select='elem'/>

</td>

</tr>

</xsl:for-each>

</table>

</xsl:otherwise>

</xsl:choose>

</body>

</html>

</xsl:template>

</xsl:stylesheet>

Тестирование 2 приложения:

require 'nokogiri'

require 'open-uri'

class Lr102Controller < ApplicationController

before\_action :parse\_params, only: :output

before\_action :prepare\_url, only: :output

def input

end

def output

output\_response = open(@url)

if @side == 'server'

@result = xslt\_transform(output\_response).to\_html

# Если делать XML -> HTML в браузере.

elsif @side == 'client-with-xslt'

render xml: insert\_browser\_xslt(output\_response).to\_xml

else

render xml: output\_response # в неизменном виде

end

end

private

BASE\_OUTPUT\_URL = 'http://localhost:3000/?format=xml'.freeze

# Откуда берется XSLT для преобразования на стороне сервера

XSLT\_SERVER\_TRANSFORM = "#{Rails.root}/public/server-transform.xslt".freeze

# Откуда браузер должен брать XSLT

XSLT\_BROWSER\_TRANSFORM = '/browser-transform.xslt'.freeze

def parse\_params

@number = params[:word].to\_i

@side = params[:side]

end

# Создается url с использованием разобранных параметров запроса

def prepare\_url

@url = BASE\_OUTPUT\_URL + "&word=#{@number}"

end

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

doc = Nokogiri::XML(data)

xslt = Nokogiri::XSLT(File.read(transform))

xslt.transform(doc)

end

def insert\_browser\_xslt(data, transform: XSLT\_BROWSER\_TRANSFORM)

doc = Nokogiri::XML(data)

xslt = Nokogiri::XML::ProcessingInstruction.new(doc, 'xml-stylesheet', 'type="text/xsl" href="' + transform + '"')

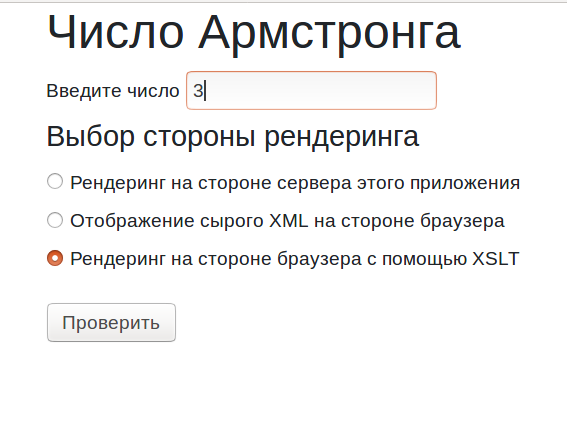
doc.root.add\_previous\_sibling(xslt)

doc

end

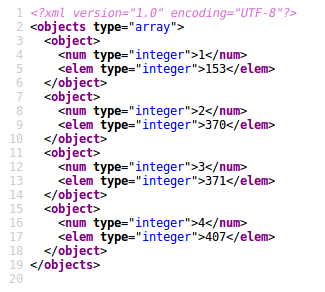
end

Результаты:

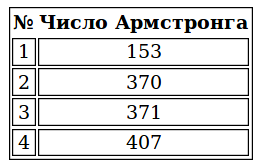




Преобразование на стороне сервера



Отображение чистого xml



Преобразование на стороне браузера



Код xml на стороне браузера