|  |  |
| --- | --- |
| 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 **09.03.03 Прикладная информатика**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 11 |

**Название:**

Добавление модели. ORM. Разработка БД, подключение, хранение и поиск данных.

**Дисциплина:** Языки интернет-программирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-35 Б |  |  | И.А. Дулина |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Е.Ю. Гаврилова |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

**Вариант 8**

**Задание:**

Модифицировать код ЛР 8 таким образом, чтобы запросы, которые были ранее выполнены, сохранялись в БД и при следующем запросе не требовали повтора вычислений.

* Сформировать модель в соответствии с потребностями хранения данных. Входные параметры являются ключами, по которым извлекается результат.
* Выполнить создание БД и миграцию соответствующими запросами rake.
* Написать тест на добавление и поиск данных с помощью модели. Проверить выполнение теста.
* Модифицировать код приложения таким образом, чтобы результат вычислений преобразовывался в строковый или бинарный формат (на выбор: json, xml, и пр.). Проверить через отладочную печать в консоль, что преобразование выполняется корректно.
* Вставить код для сохранения данных в БД и запрос на поиск предыдущего результата вычислений.
* Добавить действие в контроллер, позволяющее определить, что хранится в БД через сериализацию в XML.
* Проверить, что при выполнении запроса, данные добавляются в БД.
* При помощи консоли сообщений Puma/Webrick определить, производится ли поиск результата предыдущего запроса в БД и не повторяются ли одни и те же вычисления.
* Модифицировать модель таким образом, чтобы добавление записей с одинаковыми параметрами было невозможно (string:text:uniq)
* Реализовать тест модели, проверяющий невозможность повторного добавления одних и тех же результатов вычислений.
* Реализовать функциональный тест, проверяющий, что результаты вычислений различны при различных входных параметрах.
* Проверить маршруты приложения с помощью rake routes и убрать лишние. Обеспечить доступ при обращении по адресу /.

*В консоль:*

$rails new chisla11

$cd chisla11

$rails generate controller Chisla input view

**Сформируем модель (с невозможностью добавления одинаковых параметров, т.е. параметр string:text:uniq):**

$rails g scaffold ChislaResult string:text:uniq result:text my\_table:json

**Создание бд:**

$rake db:create

**Создание миграции:**

$rake db:migrate

*В gem file добавляем строку*

gem 'rails-controller-testing'

*config/routes.rb добавить*

root 'chisla#input'

*db/migrate/20231120135713\_create\_chisla\_results.rb*

class CreateChislaResults < ActiveRecord::Migration[7.0]

def change

create\_table :chisla\_results do |t|

t.text :string

t.text :result

t.json :my\_table

t.timestamps

end

add\_index :chisla\_results, :string, unique: true

end

end

*app/views/chisla/input.html.erb*

<h1>Chisla#input</h1>

<p>Find me in app/views/chisla/input.html.erb</p>

<div>

<form action="/chisla/view" method="get" data-remote="true" id="chisla\_form" accept-charset="UTF-8">

<div>

<label for="str">Введите не менее 10 чисел

<input type="text" id="str" name="str" required/>

</label>

</div>

<br>

<div>

<input type="submit" value="Найти наиболее длинную монотонно возврастающую последовательность"/>

</div>

</form>

</div>

*app/views/chisla/view.html.erb*

<h1>Chisla#view</h1>

<p>Find me in app/views/chisla/view.html.erb</p>

<p>Таблица результатов</p>

<%= @result[1].html\_safe %>

<br>

<%= link\_to "Найти для других чисел", :chisla\_input %>

app/controllers/chisla\_controller.rb

require 'json'

class ChislaController < ApplicationController

def input

end

def view

if params[:str]

begin

res = params[:str].scan(/-?\d+(?:\.\d+)?/).map(&:to\_i)

raise StandardError if res.length < 10

@result = check(res)

rescue StandardError

@result = [{}, 'Что-то пошло не так']

end

else

@result = [{}, 'Unknown!']

end

end

def check(res)

**#поиск предыдущего результата вычислений**

if (res\_bs = ChislaResult.find\_by\_string(res.join(' ')))

returning = [ActiveSupport::JSON::decode(res\_bs.result), JSON.parse(res\_bs.my\_table)]

pp 'Результат уже посчитан'

else

**# сохранение данных в бд:**

returning = create(res)

pp 'Результат просчитан впервые'

res\_bs = ChislaResult.create :string => res.join(' '), :result => ActiveSupport::JSON::encode(returning[0]), :my\_table => returning[1].to\_json

res\_bs.save

print 'Запись добалена в БД ', res\_bs, "\n"

end

sol = returning[0]

everything = returning[1]

[sol, create\_table(everything)]

end

def create(res)

max = 0

all = []

solution = ''

i = 0

loop do

len = 1

posl = [res[i]]

i += 1

while res[i] > res[i - 1]

len += 1

posl << res[i]

break unless i + 1 < res.length

i += 1

end

all << posl.join(' ')

if len > max

max = len

solution = posl.join(' ')

end

break if i + 1 == res.length

end

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

[solution, result]

end

def create\_table(result)

rows = "<tr><th>#{'Изначальный'}</th><th>#{'Все возможные'}</th><th>#{'Самая длинная'}</th></tr>"

result.each do |init, all, sol|

rows += "<tr><td>#{init}</td><td>#{all}</td><td>#{sol}</td></tr>"

end

@table = "<table border='1' class=\"table\"><tbody>#{rows}</tbody></table>"

end

end

*app/controllers/chisla\_results\_controller.rb добавить*

XSLT\_TRANSFORM = "#{Rails.root}/public/some\_transformer.xslt".freeze # Путь до xslt файла

# Добавить действие в контроллер, позволяющее определить, **что хранится в БД** через сериализацию в XML.

#http://127.0.0.1:3000/show\_all.xml

def show\_all

respond\_to do |format|

results = ChislaResult.all

rows = ''

results.each do |record|

rows += "<cd><former>#{record.string}</former><res\_string>#{record.result}</res\_string><my\_t>#{record.my\_table}</my\_t></cd>"

end

responce = "<catalog>#{rows}</catalog>"

format.xml { render xml: xslt\_transform(responce).to\_xml }

end

end

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

# Функция преобразования

pp 'checkpoint2'

print data, transform, "\n"

doc = Nokogiri::XML(data)

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

xslt.transform(doc)

end

*public/some\_transformer.xslt*

<?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 string</th>

<th>Result</th>

<th>Table</th>

</tr>

<xsl:for-each select="catalog/cd">

<tr>

<td><xsl:value-of select="former"/></td>

<td><xsl:value-of select="res\_string"/></td>

<td><xsl:value-of select="my\_t"/></td>

</tr>

</xsl:for-each>

</table>

</xsl:template>

</xsl:stylesheet>

*test/controllers/chisla\_controller\_test.rb*

require "test\_helper"

# rake test TEST=test/controllers/chisla\_controller\_test.rb - запуск теста

class ChislaControllerTest < ActionDispatch::IntegrationTest

# rake db:migrate RAILS\_ENV=test - запуск миграции для теста

# Написать **тест на добавление и поиск данных** с помощью модели. Проверить выполнение теста.

def add\_record(str = '1 2 3 4 5 6 7 8 9 10', data = '1 2 3 4 5 6 7 8 9 10', table = ' ')

record = ChislaResult.create :string => str, :result => ActiveSupport::JSON::encode(data), :my\_table => table

record.save

pp 'Добавлено'

record

end

test "Add data" do

record = add\_record('1 2 3 4 -1 -2 1 2 3 1', '1 2 3 4')

assert record

end

test "Find data" do

add\_record

record = ChislaResult.find\_by\_string('1 2 3 4 5 6 7 8 9 10')

assert record

end

# тест на **невозможность повторения**

test "Add same result" do

add\_record

assert\_raises(ActiveRecord::RecordNotUnique) do

add\_record

end

end

**# проверка маршрутов**: rails routes --expanded

test "should get input" do

get chisla\_input\_url

assert\_response :success

end

test "should get view" do

get chisla\_view\_url

assert\_response :success

end

# **тест** на то, что **результаты вычислений различны** при различных входных параметрах

test "different" do

get chisla\_view\_url, params: { str: '1 2 3 4 5 6 7 8 9 10' }

result1 = assigns[:result]

get chisla\_view\_url, params: { str: '-1 -2 -3 2 3 4 -2 -3 2 3' }

result2 = assigns[:result]

pp result2

assert\_not\_same result1, result2

end

#тест на то, что **запись уже добавлена**

test 'already added' do

add\_record('-1 -2 -3 2 3 4 -2 -3 2 3', '-3 2 3 4', '[["-1 -2 -3 2 3 4 -2 -3 2 3", "-1", " "], [" ", "-2", " "], [" ", "-3 2 3 4", "+"], [" ", "-2", " "], [" ", "-3 2 3", " "]]')

get chisla\_view\_url, params: { str: '-1 -2 -3 2 3 4 -2 -3 2 3' }

result2 = assigns[:result]

pp result2

assert\_equal(true, result2[2])

end

end

*config/routes.rb*

Rails.application.routes.draw do

resources :chisla\_results

get 'show\_all', to: 'chisla\_results#show\_all'

get 'chisla/input'

get 'chisla/view'

# Define your application routes per the DSL in https://guides.rubyonrails.org/routing.html

root 'chisla#input'

# Defines the root path route ("/")

# root "articles#index"

end

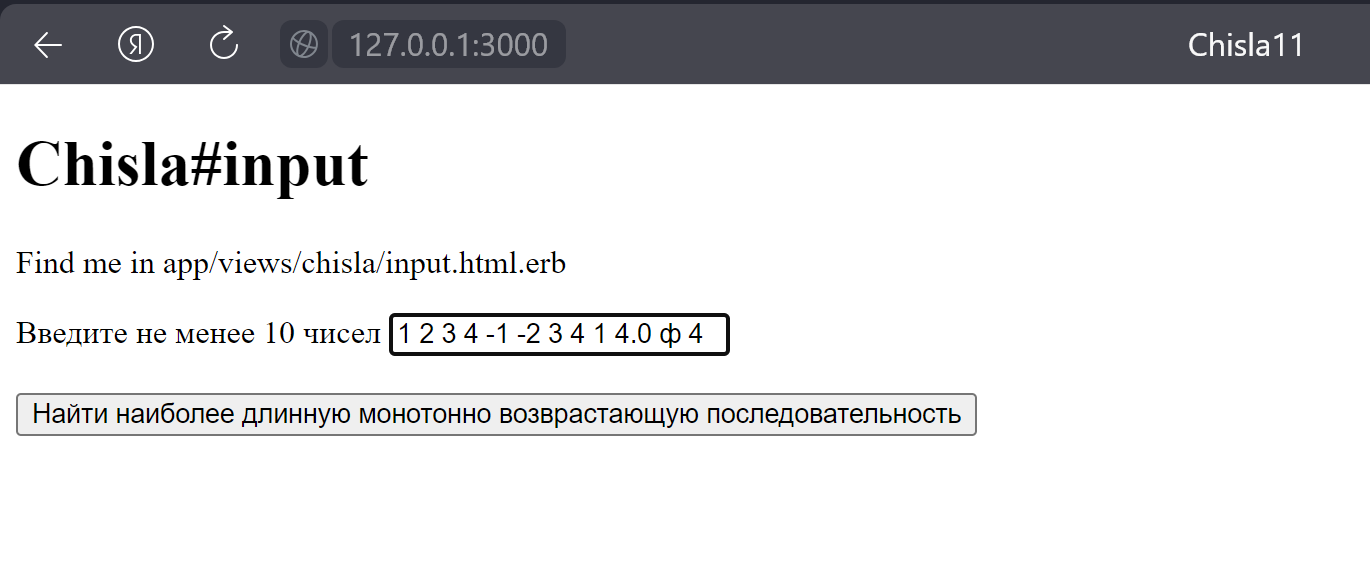


Рисунок 1 – вид страницы chisla\_input

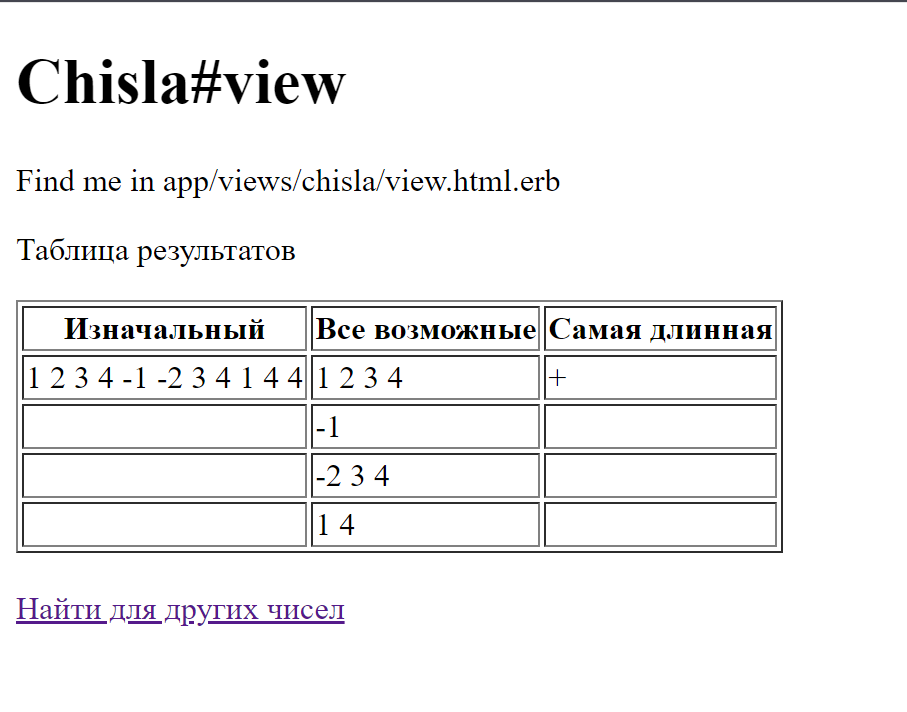


Рисунок 2 – вид страницы chisla\_view

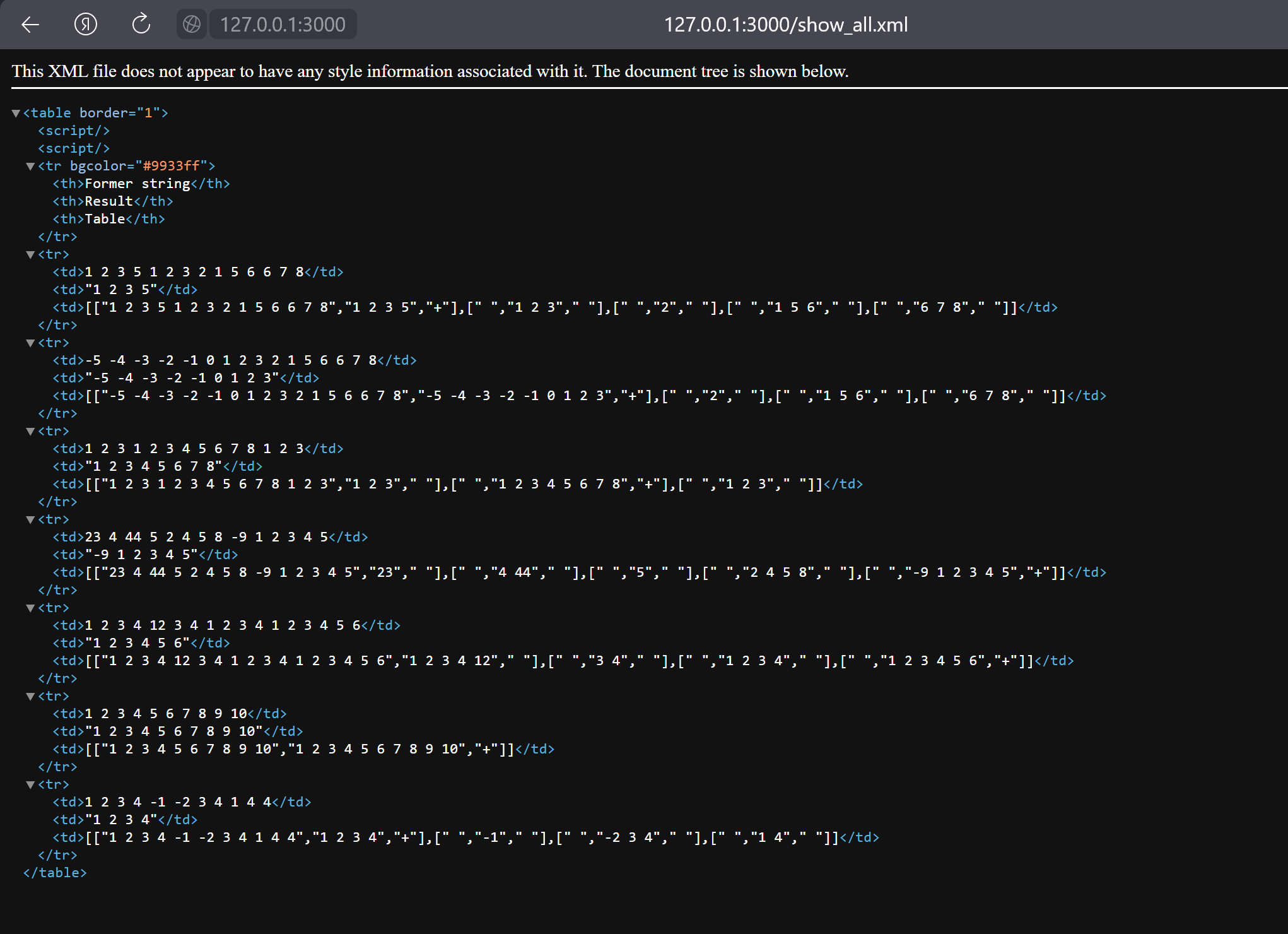


Рисунок 3 – вывод содержимого БД через сериализацию в XML

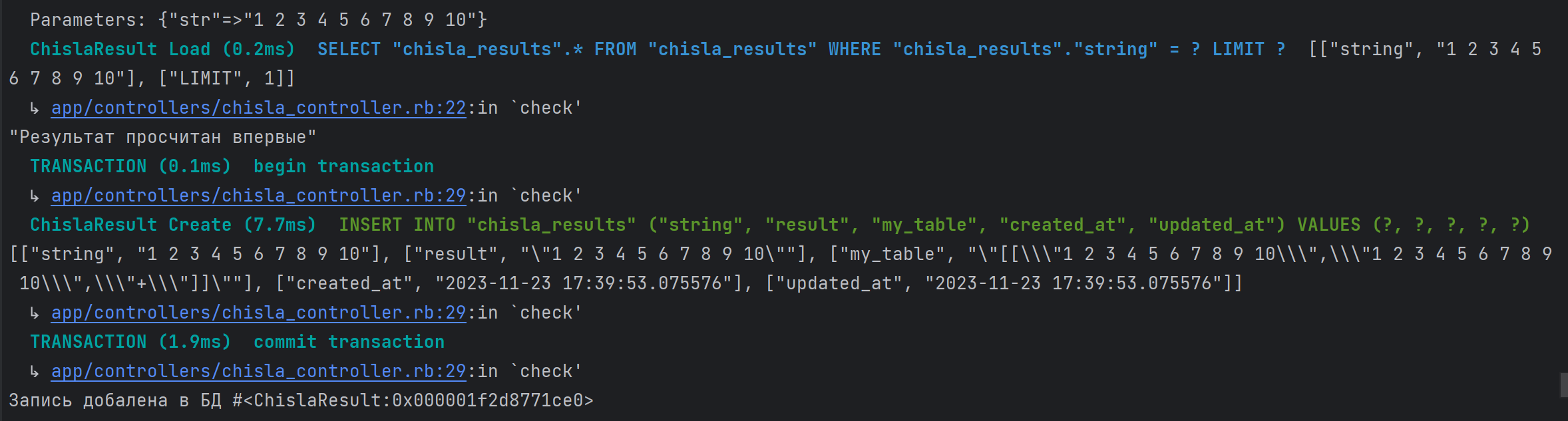


Рисунок 4 – вывод сообщения в консоль о добавлении ранее не существовавшей в БД записи

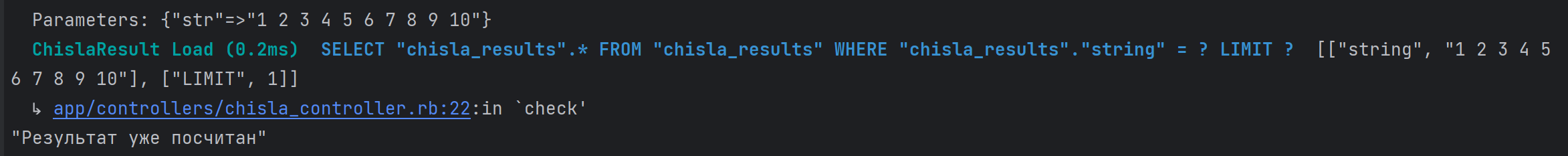


Рисунок 5 – вывод сообщения в консоли о поиске и нахождении предыдущего результата

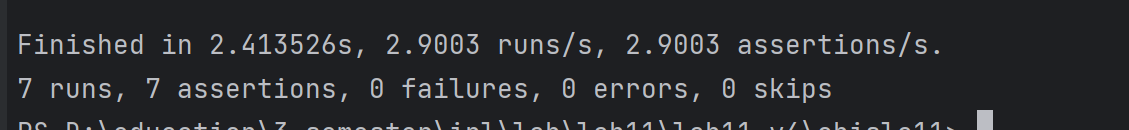


Рисунок 7 – результат выполнения тестов

**Распечатка БД:**

<?xml version="1.0" encoding="System"?>

<table>

<database></database>

<name>chisla\_results</name>

<ddl><![CDATA[CREATE TABLE "chisla\_results" ("id" integer PRIMARY KEY AUTOINCREMENT NOT NULL, "string" text, "result" text, "my\_table" json, "created\_at" datetime(6) NOT NULL, "updated\_at" datetime(6) NOT NULL);]]></ddl>

<columns>

<column>

<name>id</name>

<type>integer</type>

<constraints>

<constraint>

<type>PRIMARY KEY</type>

<definition>PRIMARY KEY AUTOINCREMENT </definition>

</constraint>

<constraint>

<type>NOT NULL</type>

<definition>NOT NULL</definition>

</constraint>

</constraints>

</column>

<column>

<name>string</name>

<type>text</type>

</column>

<column>

<name>result</name>

<type>text</type>

</column>

<column>

<name>my\_table</name>

<type>json</type>

</column>

<column>

<name>created\_at</name>

<type>datetime</type>

<constraints>

<constraint>

<type>NOT NULL</type>

<definition>NOT NULL</definition>

</constraint>

</constraints>

</column>

<column>

<name>updated\_at</name>

<type>datetime</type>

<constraints>

<constraint>

<type>NOT NULL</type>

<definition>NOT NULL</definition>

</constraint>

</constraints>

</column>

</columns>

<rows>

<row>

<value column="0">3</value>

<value column="1">1 2 3 5 1 2 3 2 1 5 6 6 7 8</value>

<value column="2">&quot;1 2 3 5&quot;</value>

<value column="3"><![CDATA["[[\"1 2 3 5 1 2 3 2 1 5 6 6 7 8\",\"1 2 3 5\",\"+\"],[\" \",\"1 2 3\",\" \"],[\" \",\"2\",\" \"],[\" \",\"1 5 6\",\" \"],[\" \",\"6 7 8\",\" \"]]"]]></value>

<value column="4">2023-11-20 18:26:06.265756</value>

<value column="5">2023-11-20 18:26:06.265756</value>

</row>

<row>

<value column="0">4</value>

<value column="1">-5 -4 -3 -2 -1 0 1 2 3 2 1 5 6 6 7 8</value>

<value column="2">&quot;-5 -4 -3 -2 -1 0 1 2 3&quot;</value>

<value column="3"><![CDATA["[[\"-5 -4 -3 -2 -1 0 1 2 3 2 1 5 6 6 7 8\",\"-5 -4 -3 -2 -1 0 1 2 3\",\"+\"],[\" \",\"2\",\" \"],[\" \",\"1 5 6\",\" \"],[\" \",\"6 7 8\",\" \"]]"]]></value>

<value column="4">2023-11-20 18:26:14.793879</value>

<value column="5">2023-11-20 18:26:14.793879</value>

</row>

<row>

<value column="0">5</value>

<value column="1">1 2 3 1 2 3 4 5 6 7 8 1 2 3</value>

<value column="2">&quot;1 2 3 4 5 6 7 8&quot;</value>

<value column="3"><![CDATA["[[\"1 2 3 1 2 3 4 5 6 7 8 1 2 3\",\"1 2 3\",\" \"],[\" \",\"1 2 3 4 5 6 7 8\",\"+\"],[\" \",\"1 2 3\",\" \"]]"]]></value>

<value column="4">2023-11-23 17:24:15.411602</value>

<value column="5">2023-11-23 17:24:15.411602</value>

</row>

<row>

<value column="0">6</value>

<value column="1">23 4 44 5 2 4 5 8 -9 1 2 3 4 5</value>

<value column="2">&quot;-9 1 2 3 4 5&quot;</value>

<value column="3"><![CDATA["[[\"23 4 44 5 2 4 5 8 -9 1 2 3 4 5\",\"23\",\" \"],[\" \",\"4 44\",\" \"],[\" \",\"5\",\" \"],[\" \",\"2 4 5 8\",\" \"],[\" \",\"-9 1 2 3 4 5\",\"+\"]]"]]></value>

<value column="4">2023-11-23 17:25:05.623126</value>

<value column="5">2023-11-23 17:25:05.623126</value>

</row>

<row>

<value column="0">7</value>

<value column="1">1 2 3 4 12 3 4 1 2 3 4 1 2 3 4 5 6</value>

<value column="2">&quot;1 2 3 4 5 6&quot;</value>

<value column="3"><![CDATA["[[\"1 2 3 4 12 3 4 1 2 3 4 1 2 3 4 5 6\",\"1 2 3 4 12\",\" \"],[\" \",\"3 4\",\" \"],[\" \",\"1 2 3 4\",\" \"],[\" \",\"1 2 3 4 5 6\",\"+\"]]"]]></value>

<value column="4">2023-11-23 17:39:06.222681</value>

<value column="5">2023-11-23 17:39:06.222681</value>

</row>

<row>

<value column="0">8</value>

<value column="1">1 2 3 4 5 6 7 8 9 10</value>

<value column="2">&quot;1 2 3 4 5 6 7 8 9 10&quot;</value>

<value column="3">&quot;[[\&quot;1 2 3 4 5 6 7 8 9 10\&quot;,\&quot;1 2 3 4 5 6 7 8 9 10\&quot;,\&quot;+\&quot;]]&quot;</value>

<value column="4">2023-11-23 17:39:53.075576</value>

<value column="5">2023-11-23 17:39:53.075576</value>

</row>

<row>

<value column="0">9</value>

<value column="1">1 2 3 4 -1 -2 3 4 1 4 4</value>

<value column="2">&quot;1 2 3 4&quot;</value>

<value column="3"><![CDATA["[[\"1 2 3 4 -1 -2 3 4 1 4 4\",\"1 2 3 4\",\"+\"],[\" \",\"-1\",\" \"],[\" \",\"-2 3 4\",\" \"],[\" \",\"1 4\",\" \"]]"]]></value>

<value column="4">2023-11-23 18:15:07.363712</value>

<value column="5">2023-11-23 18:15:07.363712</value>

</row>

<row>

<value column="0">10</value>

<value column="1">-1 -2 -3 2 3 4 -2 -3 2 3</value>

<value column="2">&quot;-3 2 3 4&quot;</value>

<value column="3"><![CDATA["[[\"-1 -2 -3 2 3 4 -2 -3 2 3\",\"-1\",\" \"],[\" \",\"-2\",\" \"],[\" \",\"-3 2 3 4\",\"+\"],[\" \",\"-2\",\" \"],[\" \",\"-3 2 3\",\" \"]]"]]></value>

<value column="4">2023-11-24 07:55:53.499932</value>

<value column="5">2023-11-24 07:55:53.499932</value>

</row>

</rows>

</table>

**Вывод:** мы научились работать с базой данных на языке Ruby, проверять наличие уже имеющихся записей в ней, добавлять новые и использовать данные из неё.