ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И

ОПТИКИ»

Факультет программной инженерии и компьютерной техники

Кафедра вычислительной техники

**Отчёт по лабораторной работе №1**

**ПО КУРСУ**

**«Технологии веб-сервисов» Поиск с помощью SOAP-сервиса**

Выполнил: студент группы P41142

Белоголов М.А.

Проверил: канд. техн. наук, доцент Дергачев А. М.

Санкт-Петербург

2020

# 

**Задание:**

В данной работе требуется создать таблицу в БД, содержащую не менее 5 полей, а также реализовать возможность поиска по любым комбинациям полей с помощью SOAP-сервиса. Данные для поиска должны передаваться в метод сервиса в качестве аргументов.

Веб-сервис необходимо реализовать в виде standalone-приложения и J2EE-приложения. При реализации в виде J2EE-приложения следует на стороне сервера приложений настроить источник данных, и осуществлять его инъекцию в коде сервиса.

Для демонстрации работы разработанных сервисов следует также разработать и клиентское консольное приложение.

**Выполнение работы:**

В файле *Picture.sql*, код которого представлен в листинге 1.1, происходит создание таблицы pictures (картины) с полями id, название, автор, год, материал, высота и ширина картины. А также происходит вставка 9 записей в таблицу pictures.

# Листинг 1.1 – Файл Picture.sql

**DROP TABLE** "pictures" **CASCADE**;

**CREATE TABLE** "pictures" ( **id bigserial NOT NULL**, **name character varying**(200), **author character varying**(200), **year integer**,

**material character varying**(200), **height numeric**, **width numeric**,

**CONSTRAINT** "pictures\_pk" **PRIMARY KEY** (**id**)

);

**INSERT INTO** pictures(**name**, **author**, **year**, **material**, **height**, **width**) **values**

(**'Мона Лиза'**, **'Леонардо да Винчи'**, 1503, **'Маслянные краски'**, 77, 53);

**INSERT INTO** pictures(**name**, **author**, **year**, **material**, **height**, **width**) **values**

(**'Звездная ночь'**, **'Винсент Ван Гог'**, 1889, **'Маслянные краски'**, 73.7, 92.1);

**INSERT INTO** pictures(**name**, **author**, **year**, **material**, **height**, **width**) **values**

(**'Тайная вечеря'**, **'Леонардо да Винчи'**, 1495, **'Темпера'**, 460, 880);

**INSERT INTO** pictures(**name**, **author**, **year**, **material**, **height**, **width**) **values**

(**'Черный квадрат'**, **'Казимир Малевич'**, 1915, **'Маслянные краски'**, 80, 80); **INSERT INTO** pictures(**name**, **author**, **year**, **material**, **height**, **width**) **values** (**'Рождение Венеры'**, **'Сандро Боттичелли'**, 1484, **'Темпера'**, 172.5, 278.5); **INSERT INTO** pictures(**name**, **author**, **year**, **material**, **height**, **width**) **values**

(**'Утро в сосновом лесу'**, **'Иван Иванович Шишкин'**, 1889, **'Маслянные краски'**, 139, 213);

**INSERT INTO** pictures(**name**, **author**, **year**, **material**, **height**, **width**) **values** (**'Постоянство памяти'**, **'Сальвадор Дали'**, 1931, **'Гобелен ручной работы'**, 24, 33);

**INSERT INTO** pictures(**name**, **author**, **year**, **material**, **height**, **width**) **values**

(**'Девочка на шаре'**, **'Пабло Пикассо'**, 1905, **'Маслянные краски'**, 147, 95);

**INSERT INTO** pictures(**name**, **author**, **year**, **material**, **height**, **width**) **values** (**'Девятый вал'**, **'Иван Константинович Айвазовский'**, 1850, **'Маслянные краски'**, 221, 332);

Код сервиса в виде standalone-приложения представлен в листингах 1.21.7. Класс *App.java* содержит main метод, и его основная цель – это запустить веб-сервис. *ConnectionUtil.java* используется для получения JDBC-соединений с базой данных. *MyRequest.java* представляет из себя структуру запроса, по которому будет выполняться поиск, который может включать поля *id*, *name*, *author*, *year*, *material*, *height*, *width*. *Picture.java* – POJO, который соответствует сущности, описанной в таблице picture базы данных. *PictureWebServise.java* содержит две операции: *getAllPictures*, который получает все картины, и *findPictures,* который ищет картины по параметру с типом *MyRequest.* *PostgreSQLDAO.java* содержит методы для выборки данных из базы данных, а также установки этих данных в объекты класса *Person*.

Листинг 1.2 – Файл App.java **package** com.maxart.service;

**import** javax.xml.ws.Endpoint;

**public class** App

{

**public static void** main( String[] args )

{

String url = **"http://0.0.0.0:8080/PictureService"**;

Endpoint.*publish*(url, **new** PictureWebService());

}

}

# Листинг 1.3 – Файл ConnectionUtil.java **package** com.maxart.service;

**import** java.sql.Connection; **import** java.sql.DriverManager; **import** java.sql.SQLException; **import** java.util.logging.Level; **import** java.util.logging.Logger;

**public class** ConnectionUtil {

**private static final** String ***JDBC\_URL*** = **"jdbc:postgresql://localhost:5432/studs"**; **private static final** String ***JDBC\_USER*** = **"\*\*\*\*\*\*"**; **private static final** String ***JDBC\_PASSWORD*** = **"\*\*\*\*\*\*"**;

**static** { **try** {

Class.*forName*(**"org.postgresql.Driver"**);

} **catch** (ClassNotFoundException ex) {

Logger.*getLogger*(PostgreSQLDAO.**class**.getName()).log(Level.***SEVERE***, **null**, ex);

}

}

**public static** Connection getConnection() { Connection connection = **null**; **try** {

connection = DriverManager.*getConnection*(***JDBC\_URL***, ***JDBC\_USER***,

***JDBC\_PASSWORD***); } **catch** (SQLException ex) {

Logger.*getLogger*(ConnectionUtil.**class**.getName()).log(Level.***SEVERE***, **null**, ex);

}

**return** connection;

}

}

# Листинг 1.4 – Файл MyRequest.java **package** com.maxart.service;

**import** javax.xml.bind.annotation.XmlAccessType; **import** javax.xml.bind.annotation.XmlAccessorType; **import** javax.xml.bind.annotation.XmlElement; **import** javax.xml.bind.annotation.XmlRootElement; **import** java.io.Serializable;

@XmlAccessorType(XmlAccessType.***FIELD***)

@XmlRootElement(name = **"MyRequest"**, namespace=**"http://standalone.maxart.com"**) **public class** MyRequest **implements** Serializable

{

@XmlElement(name = **"id"**, required = **false**) **private int id**;

@XmlElement(name = **"name"**, required = **false**) **protected** String **name**;

@XmlElement(name = **"author"**, required = **false**) **private** String **author**;

@XmlElement(name = **"year"**, required = **false**) **private int year**;

@XmlElement(name = **"material"**, required = **false**) **private** String **material**;

@XmlElement(name = **"height"**, required = **false**) **private float height**;

@XmlElement(name = **"width"**, required = **false**) **private float width**;

**public int** getId() { **return id**;

}

**public void** setId(**int** id) { **this**.**id** = id;

}

**public** String getName() { **return name**;

}

**public void** setName(String name) { **this**.**name** = name;

}

**public** String getAuthor() { **return author**;

}

**public void** setAuthor(String author) { **this**.**author** = author;

}

**public int** getYear() { **return year**;

}

**public void** setYear(**int** year) { **this**.**year** = year;

}

**public** String getMaterial() { **return material**;

}

**public void** setMaterial(String material) { **this**.**material** = material;

}

**public float** getHeight() { **return height**;

}

**public void** setHeight(**float** height) { **this**.**height** = height;

}

**public float** getWidth() { **return width**;

}

**public void** setWidth(**float** width) { **this**.**width** = width;

}

}

# Листинг 1.5 – Файл Picture.java

**package** com.maxart.service;

**public class** Picture { **private int id**; **private** String **name**; **private** String **author**; **private int year**; **private** String **material**; **private float height**; **private float width**;

**public** Picture() {

}

Picture(**int** id, String name, String author, **int** year, String material, **float** height, **float** width) { **this**.**id** = id; **this**.**name** = name; **this**.**author** = author; **this**.**year** = year; **this**.**material** = material; **this**.**height** = height; **this**.**width** = width;

}

**public int** getId() { **return id**;

}

**public void** setId(**int** id) { **this**.**id** = id;

}

**public** String getName() { **return name**;

}

**public void** setName(String name) { **this**.**name** = name;

}

**public** String getAuthor() { **return author**;

}

**public void** setAuthor(String author) { **this**.**author** = author;

}

**public int** getYear() { **return year**;

}

**public void** setYear(**int** year) { **this**.**year** = year;

}

**public** String getMaterial() { **return material**;

}

**public void** setMaterial(String material) { **this**.**material** = material;

}

**public float** getHeight() { **return height**;

}

**public void** setHeight(**float** height) { **this**.**height** = height;

}

**public float** getWidth() { **return width**;

}

**public void** setWidth(**float** width) { **this**.**width** = width;

}

@Override

**public** String toString() { **return "Picture{"** +

**"name='"** + **name** + **'\''** +

**", author='"** + **author** + **'\''** +

**", year="** + **year** +

**", material='"** + **material** + **'\''** +

**", height="** + **height** +

**", width="** + **width** +

**'}'**;

}

}

# Листинг 1.6 – Файл PictureWebService.java **package** com.maxart.service;

**import** javax.jws.WebMethod; **import** javax.jws.WebParam; **import** javax.jws.WebService; **import** java.util.List;

@WebService(serviceName = **"PictureService"**) **public class** PictureWebService {

@WebMethod(operationName = **"getAllPictures"**) **public** List<Picture> getAllPictures() { PostgreSQLDAO dao = **new** PostgreSQLDAO(); **return** dao.getAllPictures();

}

@WebMethod(operationName = **"findPictures"**)

**public** List<Picture> findPictures(@WebParam(name = **"q"**) MyRequest request) {

PostgreSQLDAO dao = **new** PostgreSQLDAO(); **return** dao.findPictures(request);

}

}

# Листинг 1.7 – Файл PostgreSQLDAO.java **package** com.maxart.service;

**import** java.sql.\*; **import** java.util.ArrayList; **import** java.util.List; **import** java.util.logging.Level; **import** java.util.logging.Logger;

**public class** PostgreSQLDAO {

**private** Connection **connection**;

PostgreSQLDAO(Connection connection) { **this**.**connection** = connection;

}

PostgreSQLDAO() {

**this**.**connection** = ConnectionUtil.*getConnection*();

}

**public** List<Picture> getAllPictures() { **return** executeQuery(**"SELECT \* FROM pictures"**);

}

**public** List<Picture> findPictures(MyRequest request) {

List<Picture> pictures = **new** ArrayList<>();

String sql = **"SELECT \* FROM pictures WHERE "** +

**"(id = "** + request.getId() + **" OR "** + request.getId() + **" = 0) AND "** +

**"(name = '"** + request.getName() + **"' OR '"** +

request.getName() + **"' = '' OR '"** + request.getName() + **"' = '?') AND "** + **"(author = '"** + request.getAuthor() + **"' OR '"** + request.getAuthor() + **"' = '' OR '"** + request.getAuthor() + **"' = '?') AND "** +

**"(year = "** + request.getYear() + **" OR "** + request.getYear() + **" = 0) AND "** +

**"(material = '"** + request.getMaterial() + **"' OR '"** + request.getMaterial() + **"' = '' OR '"** + request.getMaterial() + **"' = '?') AND "** +

**"(height = "** + request.getHeight() + **" OR "** + request.getHeight() + **" = 0) AND "** +

**"(width = "** + request.getWidth() + **" OR "** + request.getWidth() + **" = 0)"**;

**return** executeQuery(sql);

}

**private** List<Picture> executeQuery(String sql) { List<Picture> pictures = **new** ArrayList<>(); **try** {

Statement stmt = **connection**.createStatement(); ResultSet rs = stmt.executeQuery(sql); **while** (rs.next()) {

**int** id = rs.getInt(**"id"**);

String name = rs.getString(**"name"**);

String author = rs.getString(**"author"**); **int** year = rs.getInt(**"year"**);

String material = rs.getString(**"material"**); **float** height = rs.getFloat(**"height"**); **float** width = rs.getFloat(**"width"**);

Picture picture = **new** Picture(id, name, author, year, material, height, width);

pictures.add(picture);

}

} **catch** (SQLException ex) {

Logger.*getLogger*(PostgreSQLDAO.**class**.getName()).log(Level.***SEVERE***, **null**, ex);

}

**return** pictures; }

}

WSDL сервиса, представленный в листинге 1.8, доступен по адресу <http://localhost:8080/PictureService?wsdl>

Листинг 1.8 – WSDL сервиса в виде standalone-приложения *<?***xml version='1.0' encoding='UTF-8'***?><!-- Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.1.5-hudson-*

*$BUILD\_NUMBER-. --><!-- Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.1.5-hudson-$BUILD\_NUMBER-. -->*

<**definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401wss-wssecurity-utility-1.0.xsd"**

**xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"**

**xmlns:tns="http://service.maxart.com/"**

**xmlns:xsd="http://www.w3.org/2001/XMLSchema"**

**xmlns="http://schemas.xmlsoap.org/wsdl/"**

**targetNamespace="http://service.maxart.com/" name="PictureService"**>

<**types**>

<**xsd:schema**>

<**xsd:import namespace="http://standalone.maxart.com"**

**schemaLocation="http://localhost:8080/PictureService?xsd=1"**/>

</**xsd:schema**>

<**xsd:schema**>

<**xsd:import namespace="http://service.maxart.com/"**

**schemaLocation="http://localhost:8080/PictureService?xsd=2"**/>

</**xsd:schema**>

</**types**>

<**message name="findPictures"**>

<**part name="parameters" element="tns:findPictures"**/>

</**message**>

<**message name="findPicturesResponse"**>

<**part name="parameters" element="tns:findPicturesResponse"**/>

</**message**>

<**message name="getAllPictures"**>

<**part name="parameters" element="tns:getAllPictures"**/>

</**message**>

<**message name="getAllPicturesResponse"**>

<**part name="parameters" element="tns:getAllPicturesResponse"**/>

</**message**>

<**portType name="PictureWebService"**>

<**operation name="findPictures"**>

<**input message="tns:findPictures"**/>

<**output message="tns:findPicturesResponse"**/>

</**operation**>

<**operation name="getAllPictures"**>

<**input message="tns:getAllPictures"**/>

<**output message="tns:getAllPicturesResponse"**/>

</**operation**>

</**portType**>

<**binding name="PictureWebServicePortBinding" type="tns:PictureWebService"**>

<**soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"**/>

<**operation name="findPictures"**>

<**soap:operation soapAction=""**/>

<**input**>

<**soap:body use="literal"**/>

</**input**>

<**output**>

<**soap:body use="literal"**/>

</**output**>

</**operation**>

<**operation name="getAllPictures"**>

<**soap:operation soapAction=""**/>

<**input**>

<**soap:body use="literal"**/>

</**input**>

<**output**>

<**soap:body use="literal"**/>

</**output**>

</**operation**>

</**binding**>

<**service name="PictureService"**> <**port name="PictureWebServicePort" binding="tns:PictureWebServicePortBinding"**>

<**soap:address location="http://localhost:8080/PictureService"**/>

</**port**> </**service**>

</**definitions**>

Код сервиса в виде J2EE-приложения представлен в листинге 1.9. Классы *MyRequest.java*, *Picture.java*, *PostgreSQLDAO.java* аналогичны классам в standalone-приложении. *PictureWebServise.java* содержит две операции: *getAllPictures*, который получает все картины, и *findPictures,* который ищет картины по параметру с типом *MyRequest.* Также содержит инъекцию источника данных, настроенного на стороне сервера приложений glassfish.

# Листинг 1.9 – Файл PictureWebService.java **package** com.maxart.service;

**import** java.sql.Connection; **import** java.sql.SQLException; **import** java.util.List; **import** java.util.logging.Level; **import** java.util.logging.Logger; **import** javax.annotation.Resource; **import** javax.jws.WebMethod; **import** javax.jws.WebParam; **import** javax.jws.WebService; **import** javax.sql.DataSource;

@WebService(serviceName = **"PictureService"**) **public class** PictureWebService { @Resource(lookup = **"jdbc/studs"**) **private** DataSource **dataSource**;

@WebMethod(operationName = **"getAllPictures"**) **public** List<Picture> getAllPictures() {

PostgreSQLDAO dao = **new** PostgreSQLDAO(getConnection()); **return** dao.getAllPictures();

}

@WebMethod(operationName = **"findPictures"**)

**public** List<Picture> findPictures(@WebParam(name = **"q"**) MyRequest request) {

PostgreSQLDAO dao = **new** PostgreSQLDAO(getConnection());

**return** dao.findPictures(request);

}

**private** Connection getConnection() { Connection result = **null**; **try** {

result = **dataSource**.getConnection();

} **catch** (SQLException ex) {

Logger.*getLogger*(PictureWebService.**class**.getName()).log(Level.***SEVERE***, **null**, ex);

} **return** result;

}

}

WSDL сервиса, представленный в листинге 1.10, доступен по адресу <http://localhost:8080/PictureService?wsdl>

Листинг 1.10 – WSDL сервиса в виде J2EE-приложения

*<?***xml version='1.0' encoding='UTF-8'***?><!-- Published by JAX-WS RI*

*(http://jax-ws.java.net). RI's version is Metro/2.4.0 (wsit240-7e98ff4; 2017-*

*08-03T21:19:54+0200) JAXWS-RI/2.3.0 JAXWS-API/2.3.0 JAXB-RI/2.3.0 JAXB-*

*API/2.3.0 svn-revision#unknown. --><!-- Generated by JAX-WS RI*

*(http://javaee.github.io/metro-jax-ws). RI's version is Metro/2.4.0 (wsit240-*

*7e98ff4; 2017-08-03T21:19:54+0200) JAXWS-RI/2.3.0 JAXWS-API/2.3.0 JAXBRI/2.3.0 JAXB-API/2.3.0 svn-revision#unknown. -->*<**definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wsswssecurity-utility-1.0.xsd" xmlns:wsp="http://www.w3.org/ns/ws-policy" xmlns:wsp1\_2="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://service.maxart.com/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/"**

**targetNamespace="http://service.maxart.com/" name="PictureService"**>

<**types**>

<**xsd:schema**>

<**xsd:import namespace="http://service.maxart.com/" schemaLocation="http://localhost:8080/PictureService?xsd=1"**/>

</**xsd:schema**>

<**xsd:schema**>

<**xsd:import namespace="http://service.maxart.com" schemaLocation="http://localhost:8080/PictureService?xsd=2"**/>

</**xsd:schema**>

</**types**>

<**message name="getAllPictures"**>

<**part name="parameters" element="tns:getAllPictures"**/>

</**message**>

<**message name="getAllPicturesResponse"**>

<**part name="parameters" element="tns:getAllPicturesResponse"**/>

</**message**>

<**message name="findPictures"**>

<**part name="parameters" element="tns:findPictures"**/>

</**message**>

<**message name="findPicturesResponse"**>

<**part name="parameters" element="tns:findPicturesResponse"**/>

</**message**>

<**portType name="PictureWebService"**>

<**operation name="getAllPictures"**>

<**input**

**wsam:Action="http://service.maxart.com/PictureWebService/getAllPicturesReques t" message="tns:getAllPictures"**/>

<**output**

**wsam:Action="http://service.maxart.com/PictureWebService/getAllPicturesRespon se" message="tns:getAllPicturesResponse"**/>

</**operation**>

<**operation name="findPictures"**>

<**input**

**wsam:Action="http://service.maxart.com/PictureWebService/findPicturesRequest" message="tns:findPictures"**/>

<**output**

**wsam:Action="http://service.maxart.com/PictureWebService/findPicturesResponse**

**" message="tns:findPicturesResponse"**/>

</**operation**>

</**portType**>

<**binding name="PictureWebServicePortBinding" type="tns:PictureWebService"**>

<**soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"**/>

<**operation name="getAllPictures"**>

<**soap:operation soapAction=""**/>

<**input**>

<**soap:body use="literal"**/>

</**input**>

<**output**>

<**soap:body use="literal"**/>

</**output**>

</**operation**>

<**operation name="findPictures"**>

<**soap:operation soapAction=""**/>

<**input**>

<**soap:body use="literal"**/>

</**input**>

<**output**>

<**soap:body use="literal"**/>

</**output**>

</**operation**>

</**binding**>

<**service name="PictureService"**> <**port name="PictureWebServicePort" binding="tns:PictureWebServicePortBinding"**>

<**soap:address location="http://localhost:8080/PictureService"**/>

</**port**> </**service**>

</**definitions**>

Код клиента содержит файлы *FindPictures.java*,

*FindPicturesResponse.java*, *GetAllPictures.java*, *GetAllPicturesResponse.java*,

*MyRequest.java*, *ObjectFactory.java*, *package-info.java*, *Picture.java*, *PictureService.java*, *PictureWebService.java* и был сгенерирован следующей командой:

$ wsimport -keep http://localhost:8080/PictureService?wsdl

*WebServiceClient.java*, исходный код которого представлен в листинге 1.11, содержит main метод и использует сгенерированные классы для обращения к веб-сервису. В этом классе последовательно выполняются запросы *getAllPictures*, *findPictures?name=Леонардо да Винчи*, *findPictures?name=Леонардо да Винчи&year=1495*, *findPictures?id=7*.

Результат выполнения приведен на рисунке 1.1.

# Листинг 1.11 – WebServiceClient.java **package** com.maxart.client;

**import** java.net.MalformedURLException; **import** java.net.URL; **import** java.util.List;

**public class** WebServiceClient {

**public static void** main(String[] args) **throws** MalformedURLException { URL url = **new** URL(**"http://localhost:8080/PictureService?wsdl"**);

PictureService pictureService = **new** PictureService(url);

PictureWebService pictureWebService = pictureService.getPictureWebServicePort();

System.***out***.println(**"Simple hard code client for service"**);

System.***out***.println(**"Query: getAllPictures"**);

List<Picture> pictures = pictureWebService.getAllPictures(); **for** (Picture picture : pictures) { System.***out***.println(picture.toString());

}

System.***out***.println(**"Total pictures: "** + pictures.size());

System.***out***.println();

System.***out***.println(**"Query: findPictures?name=Леонардо да Винчи"**); MyRequest myRequest = **new** MyRequest(); myRequest.setAuthor(**"Леонардо да Винчи"**); pictures = pictureWebService.findPictures(myRequest); **for** (Picture picture : pictures) { System.***out***.println(picture.toString());

}

System.***out***.println(**"Total pictures: "** + pictures.size());

System.***out***.println();

System.***out***.println(**"Query: findPictures?name=Леонардо да Винчи&year=1495"**);

myRequest.setYear(1495);

pictures = pictureWebService.findPictures(myRequest); **for** (Picture picture : pictures) { System.***out***.println(picture.toString());

}

System.***out***.println(**"Total pictures: "** + pictures.size());

System.***out***.println();

System.***out***.println(**"Query: findPictures?id=7"**); myRequest.setId(7);

myRequest.setYear(0); myRequest.setAuthor(**""**);

pictures = pictureWebService.findPictures(myRequest);

**for** (Picture picture : pictures) { System.***out***.println(picture.toString());

}

System.***out***.println(**"Total pictures: "** + pictures.size());

System.***out***.println();

}

}

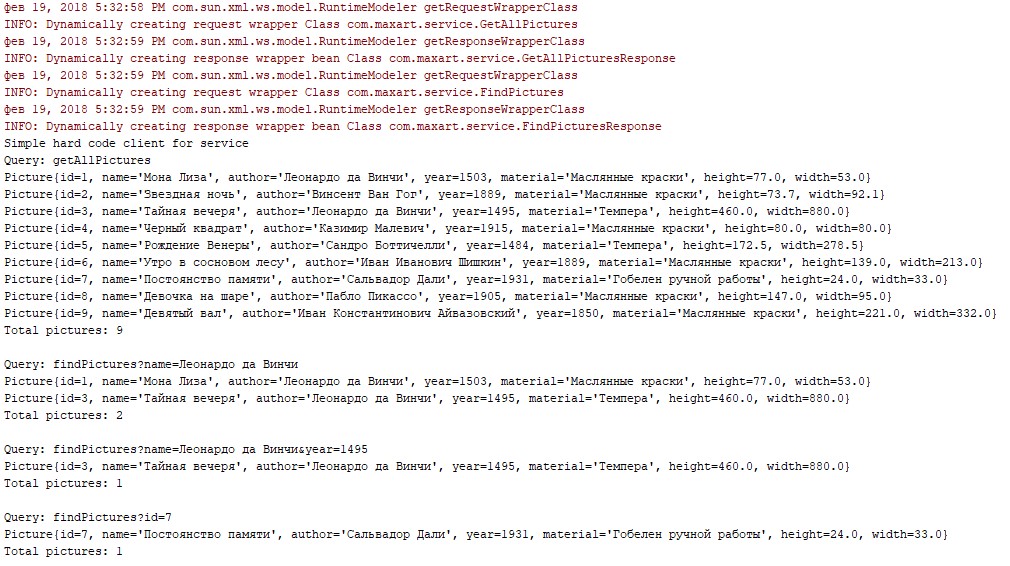


Рисунок 1.1 – Результат выполнения клиентского консольного приложения

**Вывод:** в ходе выполнения работы была создана и заполнена таблица pictures, а также реализована возможность поиска по любым комбинациям полей с помощью SOAP-сервиса в виде standalone-приложения и J2EEприложения. Для демонстрации работы разработанных сервисов было разработано клиентское консольное приложение.