**Министерство образования и науки Российской Федерации**

федеральное государственное автономное образовательное учреждение высшего образования

**“НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**УНИВЕРСИТЕТ ИТМО**”

**Факультет программной инженерии и компьютерной техники**

**ЛАБОРАТОРНАЯ РАБОТА № 4**

**ПО ДИСЦИПЛИНЕ «ТЕХНОЛОГИИ ВЕБ-СЕРВИСОВ»**

Студент: Су Минцзе

Группа: P41141

Преподаватель: Дергачев Андрей Михайлович

Санкт-Петербург

2021

**Задание:**

Необходимо выполнить задание из первой лабораторной работы, но с использованием REST-сервиса. Таблицу базы данных, а также код для ее работы с ней можно оставить без изменений. Веб-сервис необходимо реализовать в виде standalone-приложения и J2EE-приложения. Для демонстрации работы разработанных сервисов следует также разработать и клиентское консольное приложение.

**Выполнение работы**:

Код сервиса в виде standalone-приложения представлен в листингах 4.1-4.5. Класс App.java содержит main метод, и его основная цель – это запустить веб-сервис. ConnectionUtil.java используется для получения JDBC-соединений с базой данных. Picture.java – POJO, который соответствует сущности, описанной в таблице picture базы данных. PictureResource.java – класс, реализующий REST-ресурс, содержит две операции: getOne, который получает картину по указанному id, и find, который ищет картины по полученным параметрам. PostgreSQLDAO.java содержит методы для выборки данных из базы данных, а также установки этих данных в объекты класса Person

Листинг 4.1 – Файл App.java

public class App

{

private static final URI BASE\_URI = URI.create("http://localhost:8080/rest/");

public static void main(String[] args) {

HttpServer server = null;

try {

ResourceConfig resourceConfig = new ClassNamesResourceConfig(PictureResource.class);

server = GrizzlyServerFactory.createHttpServer(BASE\_URI,resourceConfig);

server.start();

System.in.read();

stopServer(server);

} catch (IOException e) {

e.printStackTrace();

stopServer(server);

}

}

private static void stopServer(HttpServer server) {

if (server != null)

server.stop();

}

}

Листинг 4.2 – Файл ConnectionUtil.java

package com.sumingjie.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/ifmo-ws";

private static final String JDBC\_USER = "ifmo-ws";

private static final String JDBC\_PASSWORD = "ifmo-ws";

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 4.3 – Файл Picture.java

@XmlRootElement

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 4.4 – Файл PictureResource.java

package com.sumingjie.service；

import java.util.List;

import javax.ws.rs.GET;

import javax.ws.rs.Path;

import javax.ws.rs.PathParam;

import javax.ws.rs.Produces;

import javax.ws.rs.core.Context;

import javax.ws.rs.core.MediaType;

import javax.ws.rs.core.UriInfo;

@Path("/pictures")

@Produces({MediaType.APPLICATION\_JSON})

public class PictureResource {

@GET

public List<Picture> find(@Context UriInfo info) {

String id = info.getQueryParameters().getFirst("id");

String name = info.getQueryParameters().getFirst("name");

String author = info.getQueryParameters().getFirst("author");

String year = info.getQueryParameters().getFirst("year");

String material = info.getQueryParameters().getFirst("material");

String height = info.getQueryParameters().getFirst("height");

String width = info.getQueryParameters().getFirst("width");

return new PostgreSQLDAO().findPictures(id, name, author, year, material, height, width);

}

@GET

@Path("/{id}")

public List<Picture> getOne(@PathParam("id") int id) {

return new PostgreSQLDAO().findOne(id); }}

Листинг 4.5 – Файл PostgreSQLDAO.java

package com.sumingjie.service;

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

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() {

this.connection = ConnectionUtil.getConnection();

}

public List<Picture> findPictures(String id, String name, String author, String year, String material, String height, String width) {

StringBuilder sb = new StringBuilder("");

StringBuilder query = new StringBuilder("");

boolean where = false;

if (id != null) {

sb.append("id = ").append(Integer.parseInt(id)).append(" AND ");

where = true;

}

if (name != null) {

sb.append("name = '").append(name).append("' AND ");

where = true;

}

if (author != null) {

sb.append("author = '").append(author).append("' AND ");

where = true;

}

if (year != null) {

sb.append("year = ").append(Integer.parseInt(year)).append(" AND ");

where = true;

}

if (material != null) {

sb.append("material = '").append(material).append("' AND ");

where = true;

}

if (height != null) {

sb.append("height = ").append(Float.parseFloat(height)).append(" AND ");

where = true;

}

if (width != null) {

sb.append("width = ").append(Float.parseFloat(width)).append(" AND ");

where = true;

}

if (where) {

if (sb.toString().endsWith(" AND ")) {

sb.setLength(sb.length() - 5);

}

query.append("SELECT \* FROM pictures WHERE ").append(sb.toString());

} else {

query.append("SELECT \* FROM pictures");

}

return executeQuery(query.toString());

}

public List<Picture> findOne(int id) {

String query = "SELECT \* FROM pictures WHERE id = " + id;

List<Picture> pictures = executeQuery(query);

return pictures;

}

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; }}

Код сервиса в виде J2EE-приложения представлен в листинге 4.6. Классы Picture.java, PostgreSQLDAO.java аналогичны классам standalone- приложения. PictureResource.java – класс, реализующий REST-ресурс, содержит две операции: getOne, который получает картину по указанному id, и find, который ищет картины по полученным параметрам. Также содержит инъекцию источника данных, настроенного на стороне сервера приложений glassfish

Листинг 4.6 – Файл PictureResource.java

package com.sumingjie.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.enterprise.context.RequestScoped;

import javax.sql.DataSource;

import javax.ws.rs.GET;

import javax.ws.rs.Path;

import javax.ws.rs.PathParam;

import javax.ws.rs.Produces;

import javax.ws.rs.core.Context;

import javax.ws.rs.core.MediaType;

import javax.ws.rs.core.UriInfo;

@RequestScoped

@Path("/pictures")

@Produces({MediaType.APPLICATION\_JSON})

public class PictureResource {

@Resource(lookup = "jdbc/ifmo-ws")

private DataSource dataSource;

/\* @GET

public List<Picture> find(@Context UriInfo info) {

String id = info.getQueryParameters().getFirst("id");

String name = info.getQueryParameters().getFirst("name");

String author = info.getQueryParameters().getFirst("author");

String year = info.getQueryParameters().getFirst("year");

String material = info.getQueryParameters().getFirst("material");

String height = info.getQueryParameters().getFirst("height");

String width = info.getQueryParameters().getFirst("width");

return new PostgreSQLDAO(getConnection()).findPictures(id, name, author, year, material, height, width);

}\*/

@GET

public List<Picture> getOne(@PathParam("id") int id) {

return new PostgreSQLDAO(getConnection()).findOne(id);

}

private Connection getConnection() {

Connection result = null;

try {

result = dataSource.getConnection();

} catch (SQLException ex) {

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

}

return result; }}

Код клиента содержит файлы Picture.java, который был представлен в листинге 4.3. App.java, исходный код которого представлен в листинге 4.7, содержит следующие методы:

• метод main – последовательно выполняет запросы к веб-сервису;

• метод display выводит результат ответа;

• метод getQueryMap парсит строку вида *name=Леонардо да Винчи&year=1495* на пары ключ-значение;

• метод findPictures непосредственно выполняет запрос, принимает на вход client, url и query – строка вида *name=Леонардо да Винчи&year=1495*, по которой выполняется поиск.

В классе App.java последовательно выполняются запросы /pictures, /pictures?author*=Леонардо да Винчи*, /pictures?author*=Леонардо да Винчи&year=1495*, /pictures?id=7 и /pictures/7. Результат выполнения приведен на рисунке 4.1.

Листинг 4.6 – App.java

package com.sumingjie.client;

import com.sun.jersey.api.client.Client;

import com.sun.jersey.api.client.ClientResponse;

import com.sun.jersey.api.client.GenericType;

import com.sun.jersey.api.client.WebResource;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.Set;

import javax.ws.rs.core.MediaType;

public class App {

private static final String URL = "http://localhost:8080/rest/pictures";

public static void main(String[] args){

Client client = Client.create();

System.out.println("Query:/pictures");

printList(findPictures(client,URL,""));

System.out.println();

System.out.println("query:/pictures?author=Леонардо да Винчи");

printList(findPictures(client,URL,"author=Леонардо да Винчи"));

System.out.println();

System.out.println("Query: /pictures?author=Леонардо да Винчи&year=1495");

printList(findPictures(client, URL, "author=Леонардо да Винчи&year=1495")); System.out.println();

System.out.println("Query: /pictures?id=7");

printList(findPictures(client, URL, "id=7"));

System.out.println();

System.out.println("Query: /pictures/7");

printList(findPictures(client, URL + "/7", ""));

System.out.println();

}

private static List<Picture> findPictures(Client client,String url,String query){

WebResource webResource = client.resource(url);

if(!query.isEmpty()){

Map<String,String> map = getQueryMap(query);

Set<String> keys = map.keySet();

for(String key : keys){

webResource = webResource.queryParam(key,map.get(key));

}

}

ClientResponse response = webResource.accept(MediaType.APPLICATION\_JSON).get(ClientResponse.class);

if(response.getStatus() != ClientResponse.Status.OK.getStatusCode()){

throw new IllegalStateException("Requset failed");

}

GenericType<List<Picture>> type = new GenericType<List<Picture>>(){

};

return response.getEntity(type);

}

private static Map<String,String> getQueryMap(String query){

String [] params = query.split("&");

Map<String,String> map = new HashMap<String,String>();

for(String param:params){

String name = param.split("=")[0];

String value = param.split("=")[1];

map.put(name,value);

}

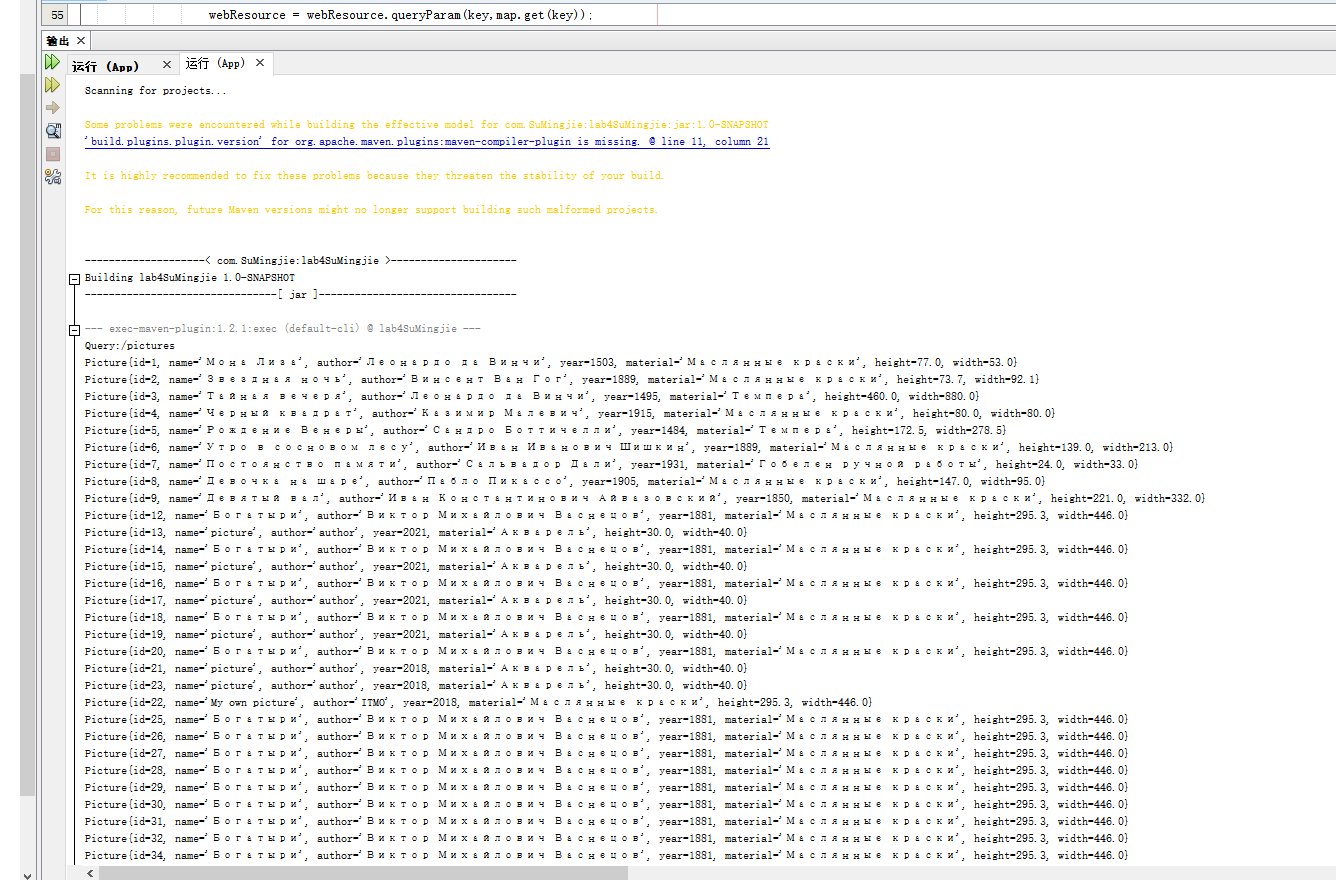
return map;

}

private static void printList(List<Picture> pictures){

for(Picture picture : pictures){

System.out.println(picture); } }}



**Вывод:** в ходе выполнения работы была реализована возможность поиска по любым комбинациям полей с помощью REST-сервиса в виде standalone-приложения и J2EE-приложения. Для демонстрации работы разработанных сервисов было разработано клиентское консольное приложение.