**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 № 1**

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

Студент: Су Минцзе

Группа: P41141

Преподаватель: Дергачев Андрей Михайлович

Санкт-Петербург

2021

**Задание:**

В данной работе в веб-сервис, разработанный в первой работе,

необходимо добавить методы для создания, изменения и удаления записей из таблицы. Метод создания должен принимать значения полей новой записи, метод изменения – идентификатор изменяемой записи, а также новые значения полей, а метод удаления – только идентификатор удаляемой записи.

Метод создания должен возвращать идентификатор новой записи, а

методы обновления или удаления – статус операции. В данной работе следует вносить изменения только в standalone-реализацию сервиса. В соответствии с изменениями сервиса необходимо обновить и клиентское приложение.

**Выполнение работы**:

В результате выполнения первой лабораторной работы был написан

SOAP-сервис в виде standalone-приложения. Дополним классы

PictureWebServise.java и PostgreSQLDAO.java методами createPicture,

updatePicture, deletePicture, которые позволяют создавать, обновлять и удалять записи из таблицы pictures соответственно.

Метод createPicture получает на вход имя, автора, год, материал, высоту и ширину картины. В случае успешного добавления возвращает id записи, иначе -1. Метод updatePicture получает на вход id, по которому будет обновлена запись, и параметр типа MyRequest, который содержит данные для обновления указанной записи. В случае успешного обновления записи возвращает 1, иначе -1. Метод deletePicture получает на вход id, по которому будет удалена запись. В случае успешного удаления записи возвращает 1, иначе -1. Исходный код классов PictureWebServise.java и PostgreSQLDAO.java представлен в листингах 2.1-2.2.

Листинг 2.1 – Файл PictureWebService.java

package com.sumingjie.service;

import java.sql.SQLException;

import java.util.List;

import javax.jws.WebMethod;

import javax.jws.WebParam;

import javax.jws.WebService;

@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);

}

@WebMethod(operationName = "createPicture")

public int createPicture(@WebParam(name = "name") String name,

@WebParam(name = "author") String author,

@WebParam(name = "year") int year,

@WebParam(name = "material") String material,

@WebParam(name = "height") float height,

@WebParam(name = "width") float width) {

PostgreSQLDAO dao = new PostgreSQLDAO();

try {

return dao.createPicture(name, author, year, material, height, width);

} catch (SQLException e) {

e.printStackTrace();

}

return -1;

}

@WebMethod(operationName = "updatePicture")

public int updatePicture(@WebParam(name = "id") int id,

@WebParam(name = "q") MyRequest request) {

PostgreSQLDAO dao = new PostgreSQLDAO();

try {

return dao.updatePicture(id, request);

} catch (SQLException e) {

e.printStackTrace();

}

return -1;

}

@WebMethod(operationName = "deletePicture")

public int deletePicture(@WebParam(name = "id") int id) {

PostgreSQLDAO dao = new PostgreSQLDAO();

try {

return dao.deletePicture(id);

} catch (SQLException e) {

e.printStackTrace();

}

return -1;

}

}

Листинг 2.2 – Файл PostgreSQLDAO.java

package com.sumingjie.service;

import java.sql.Connection;

import java.sql.PreparedStatement;

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> getAllPictures() {

return executeQuery("SELECT \* FROM pictures");

}

public List<Picture> findPictures(MyRequest request) {

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);

}

public int createPicture(String name, String author, int year, String material, float height, float width) throws SQLException {

String sql = "INSERT INTO pictures (name, author, year, material, height, width) VALUES(?, ?, ?, ?, ?, ?)";

PreparedStatement preparedStatement;

preparedStatement = this.connection.prepareStatement(sql);

preparedStatement.setString(1, name);

preparedStatement.setString(2, author);

preparedStatement.setInt(3, year);

preparedStatement.setString(4, material);

preparedStatement.setFloat(5, height);

preparedStatement.setFloat(6, width);

int affectedRows = preparedStatement.executeUpdate();

if (affectedRows == 0) {

return -1;

}

try (ResultSet generatedKeys = preparedStatement.getGeneratedKeys()) {

if (generatedKeys.next()) {

return (int) generatedKeys.getLong(1);

}

else {

return -1;

}

}

}

public int updatePicture(int id, MyRequest request) throws SQLException {

String sql = "UPDATE pictures SET" + createUpdateQuery(request) + " WHERE id=?";

PreparedStatement preparedStatement = this.connection.prepareStatement(sql);

preparedStatement.setInt(1, id);

int affectedRows = preparedStatement.executeUpdate();

return affectedRows == 0 ? -1 : 1;

}

public int deletePicture(int id) throws SQLException {

String sql = "DELETE FROM pictures WHERE id = ?";

PreparedStatement preparedStatement = this.connection.prepareStatement(sql);

preparedStatement.setInt(1, id);

int affectedRows = preparedStatement.executeUpdate();

return affectedRows == 0 ? -1 : 1;

}

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;

}

private String createUpdateQuery(MyRequest request) {

StringBuilder stringBuilderField = new StringBuilder("(");

StringBuilder stringBuilderValues = new StringBuilder("(");

if (request.getId() > 0) {

stringBuilderField.append("id,");

stringBuilderValues.append(request.getId()).append(",");

}

if (!request.getName().equals("?") && !request.getName().equals("")) {

stringBuilderField.append("name,");

stringBuilderValues.append("'").append(request.getName()).append("',");

}

if (!request.getAuthor().equals("?") && !request.getAuthor().equals("")) {

stringBuilderField.append("author,");

stringBuilderValues.append("'").append(request.getAuthor()).append("',");

}

if (request.getYear() > 0) {

stringBuilderField.append("year,");

stringBuilderValues.append(request.getYear()).append(",");

}

if (!request.getMaterial().equals("?") && !request.getMaterial().equals("")) {

stringBuilderField.append("material,");

stringBuilderValues.append("'").append(request.getMaterial()).append("',");

}

if (request.getHeight() > 0) {

stringBuilderField.append("height,");

stringBuilderValues.append(request.getHeight()).append(",");

}

if (request.getWidth() > 0) {

stringBuilderField.append("width,");

stringBuilderValues.append(request.getWidth()).append(",");

}

if (stringBuilderField.toString().endsWith(",")) {

stringBuilderField.setLength(stringBuilderField.length() - 1);

stringBuilderValues.setLength(stringBuilderValues.length() - 1);

}

stringBuilderField.append(")");

stringBuilderValues.append(")");

return stringBuilderField.toString() + " = " + stringBuilderValues.toString();

}

}

WSDL сервиса, представленный в листинге 2.3, доступен по адресу

http://localhost:8080/PictureService?wsdl



Код

клиента содержит файлы CreatePicture.java, CreatePictureResponse.java, DeletePicture.java, DeletePictureResponse.java, FindPictures.java, FindPicturesResponse.java, GetAllPictures.java, GetAllPicturesResponse.java, MyRequest.java, ObjectFactory.java, packageinfo.java, Picture.java, PictureService.java, PictureWebService.java, UpdatePicture.java, UpdatePictureResponse.java и был сгенерирован следующей командой:

$ wsimport -keep http://localhost:8080/PictureService?wsdl –p com.maxart.client

Исходный код WebServiceClient.java представлен в листинге 2.4. В этом

классе последовательно выполняются запросы createPicture, updatePicture и

deletePicture. Результат выполнения приведен на рисунке 2.1.

Листинг 2.4 – WebServiceClient.java

package com.sumingjie.client;

import java.net.MalformedURLException;

import java.net.URL;

import java.util.List;

public class WebServiceClient {

public static void getStatus(PictureWebService pictureWebService) {

System.out.println("Pictures Status");

List<Picture> pictures = pictureWebService.getAllPictures();

for (Picture picture : pictures) {

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

}

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

System.out.println();

}

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");

getStatus(pictureWebService);

System.out.println("Query: createPicture?name=Богатыри&author=Виктор Михайлович Васнецов&" +

"year=1881&material=Маслянные краски&height=295.3&width=446");

int creatingPictureId = pictureWebService.createPicture(

"Богатыри",

"Виктор Михайлович Васнецов",

1881,

"Маслянные краски",

295.3f,

446);

System.out.println("Inserting id: " + creatingPictureId);

System.out.println();

getStatus(pictureWebService);

System.out.println("Query: createPicture?name=picture&author=author&" +

"year=2018&material=Акварель&height=30&width=40");

creatingPictureId = pictureWebService.createPicture(

"picture",

"author",

2018,

"Акварель",

30,

40);

System.out.println("Inserting id: " + creatingPictureId);

System.out.println();

getStatus(pictureWebService);

System.out.println("Query: updatePicture?id=11&name=My own picture&author=ITMO&year=2018");

MyRequest myRequest = new MyRequest();

myRequest.setName("My own picture");

myRequest.setAuthor("ITMO");

myRequest.setYear(2018);

creatingPictureId = pictureWebService.updatePicture(11, myRequest);

System.out.println("Updating status: " + creatingPictureId);

System.out.println();

getStatus(pictureWebService);

System.out.println("Query: updatePicture?id=22&name=My own picture&author=ITMO&year=2018");

creatingPictureId = pictureWebService.updatePicture(22, myRequest);

System.out.println("Updating status: " + creatingPictureId);

System.out.println();

getStatus(pictureWebService);

System.out.println("Query: deletePicture?id=11");

creatingPictureId = pictureWebService.deletePicture(11);

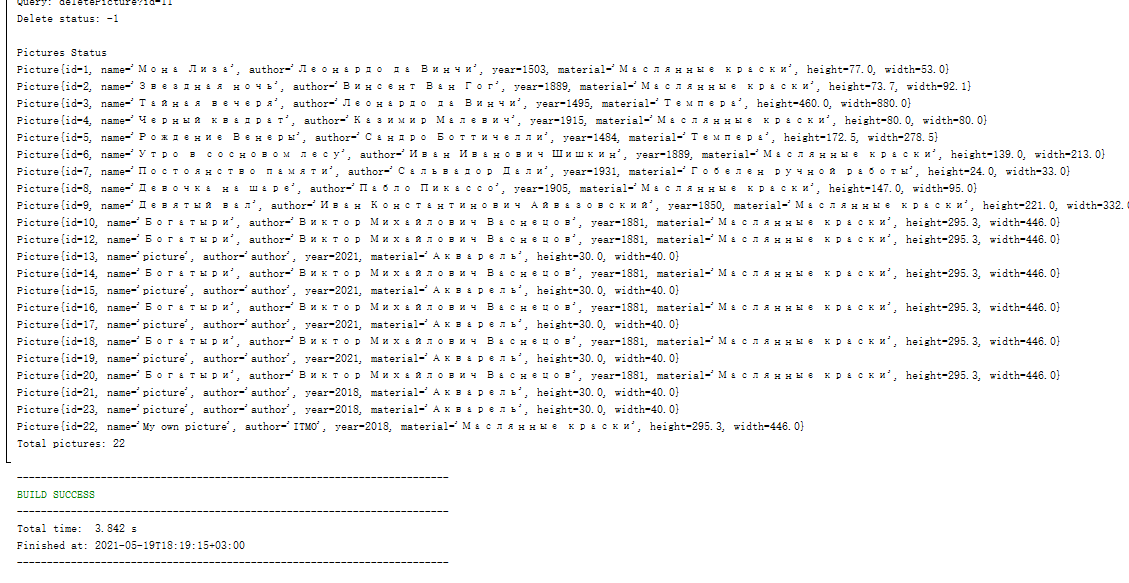
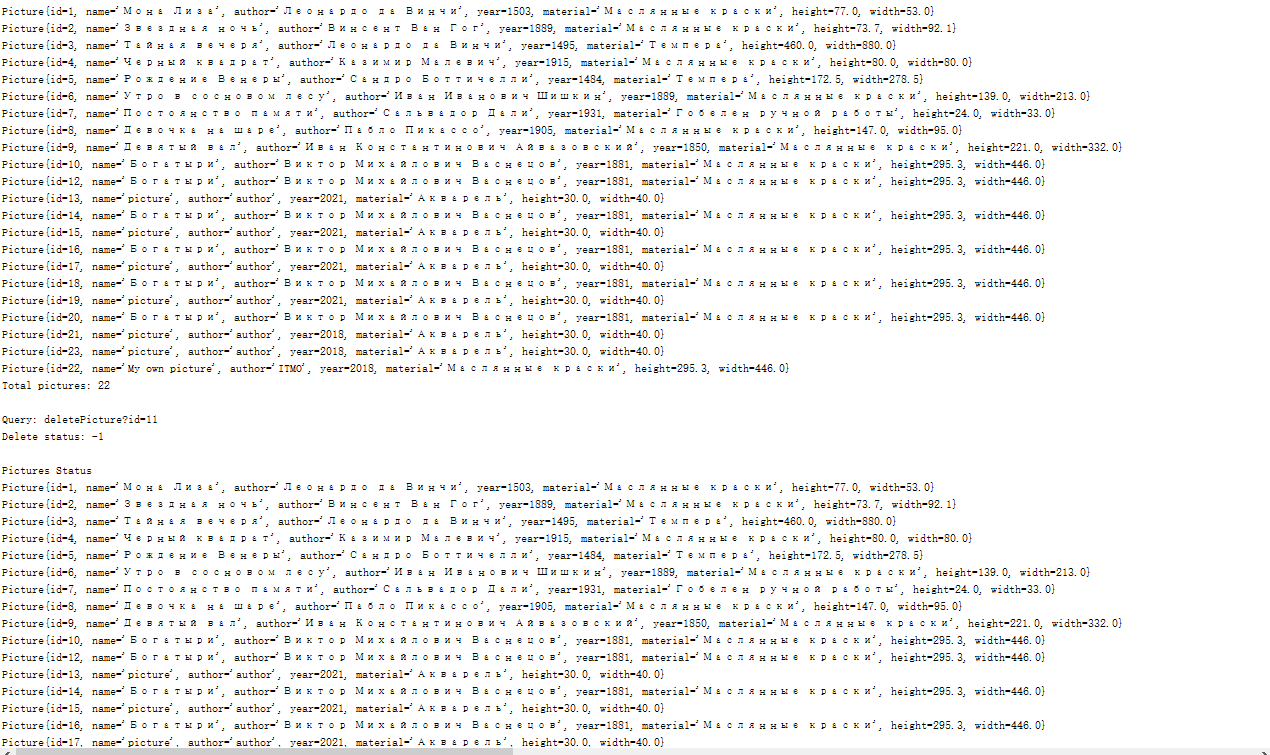
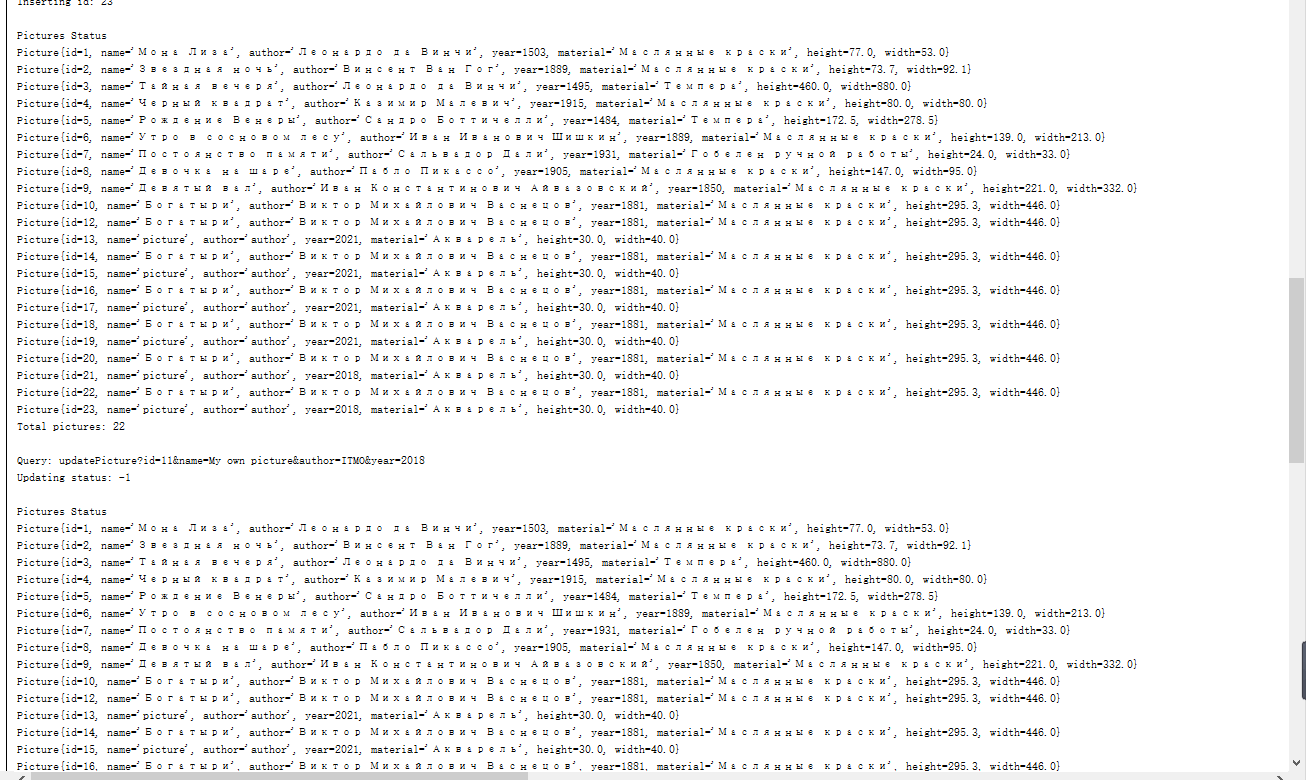
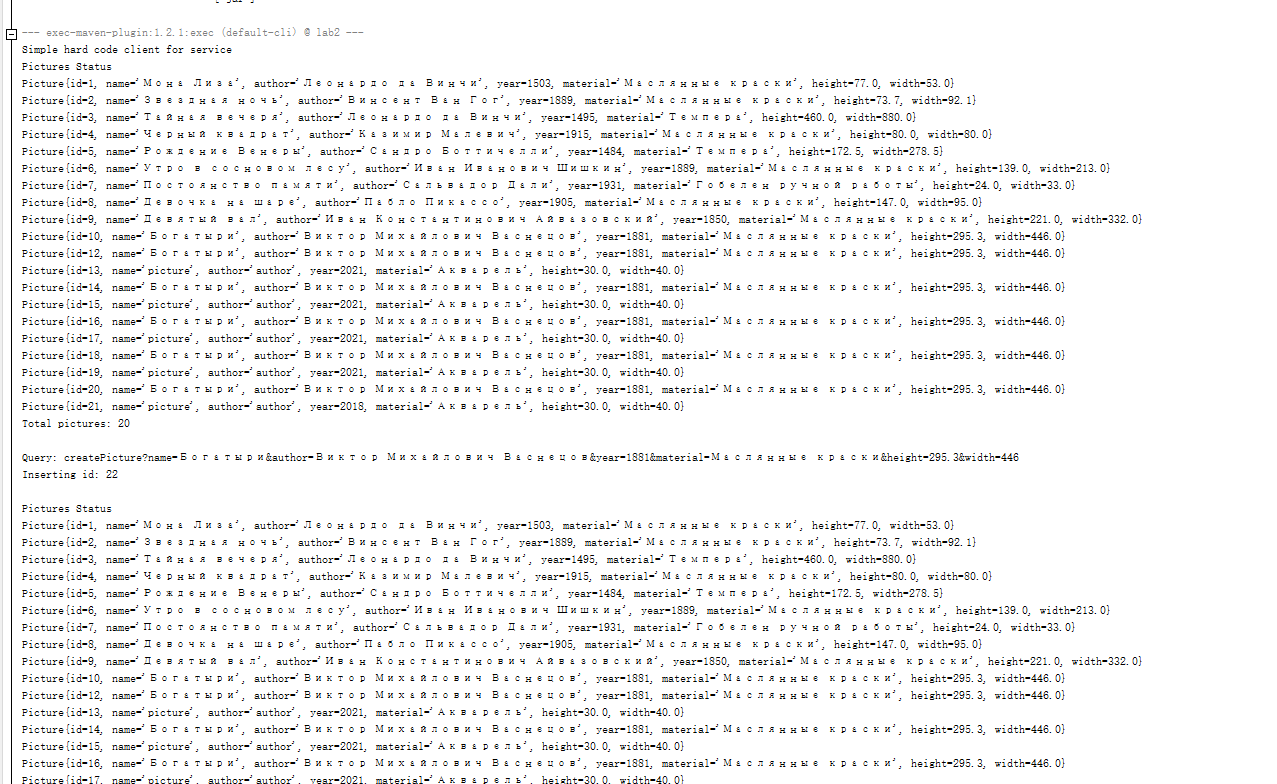
System.out.println("Delete status: " + creatingPictureId);

System.out.println();

getStatus(pictureWebService);

}

}



**Вывод**: в ходе выполнения работы был реализован CRUD с помощью SOAP-сервиса в виде standalone-приложения. Для демонстрации работы разработанного сервиса было разработано клиентское консольное приложение.