**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 № 3**

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

Студент: Су Минцзе

Группа: P41141

Преподаватель: Дергачев Андрей Михайлович

Санкт-Петербург

2021

**Задание:**

Основываясь на информации из раздела 2.8, добавить поддержку обработки ошибок в сервис. Возможные ошибки, которые могут происходить при добавлении новых записей – например, неверное значение одного из полей, при изменении, удалении – попытка изменить или удалить несуществующую запись. В соответствии с изменениями сервиса необходимо обновить и клиентское приложение.

**Выполнение работы**:

В результате выполнения второй лабораторной работы был написан SOAP-сервис в виде standalone-приложения с операциями CRUD. В классы PictureWebServise.java и PostgreSQLDAO.java были добавлены исключительные ситуации:

• Метод findPictures в случае, если полностью не установлены критерии поиска, выдает исключение IllegalQException.

• Метод createPicture в случае, если не установлен один из параметров для создания записи, выдает исключение InvalidCreatingParametersException и указывает на параметр. Если при вставке такой id уже есть в таблице, то выдает исключение InsertingException. В случае успеха возвращает id новой записи.

• Метод updatePicture в случае, если не установлен id или полностью данные для обновления записи,выдает исключения IllegalIdException и IllegalQException соответственно. Если при обновлении записи не найден указанный id, то выдает исключение InvalidEntityException. В случае успеха возвращает количество обновленных полей.

• Метод deletePicture в случае, если не установлен id, выдает исключение IllegalIdException. Если при удалении записи не найден указанный id, то выдает исключение InvalidEntityException. В случае успеха возвращает количество удаленных полей.В класс App.java добавлена строка, позволяющая не выводить stacktrace выбрасываемого исключения. PictureServiceFault – класс, используемый в качестве faultBean. Классы IllegalIdException.java, IllegalQException.java, InvalidCreatingParametersException.java, InsertingException.java и InvalidEntityException.java – классы-наследники java.lang.Exception. Исходный

код классов App.java, PictureWebServise.java, PostgreSQLDAO.java, PictureServiceFault.java, IllegalIdException.java, IllegalQException.java, InvalidCreatingParametersException.java, InsertingException.java и InvalidEntityException.java представлен в листингах

3.1-3.9.

Листинг 3.1 – Файл App.java

package com.sumingjie.service;

import javax.xml.ws.Endpoint;

public class App {

public static void main(String[] args) {

//disable stacktraces in soap-message

System.setProperty("com.sun.xml.ws.fault.SOAPFaultBuilder.disableCaptureStackTrace", "false");

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

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

}

}

Листинг 3.2 – Файл PictureWebService.java

package com.sumingjie.service;

import com.sumingjie.serviceexception.AuthException;

import com.sumingjie.serviceexception.IllegalIdException;

import com.sumingjie.serviceexception.IllegalQException;

import com.sumingjie.serviceexception.InsertingException;

import com.sumingjie.serviceexception.InvalidCreatingParametersException;

import com.sumingjie.serviceexception.InvalidEntityException;

import java.util.List;

import java.util.Map;

import java.util.StringTokenizer;

import javax.annotation.Resource;

import javax.jws.WebMethod;

import javax.jws.WebParam;

import javax.jws.WebService;

import javax.xml.ws.WebServiceContext;

import javax.xml.ws.handler.MessageContext;

import org.postgresql.util.Base64;

/\*\*

\*

\* @author Administrator

\*/

@WebService(serviceName = "PictureService")

public class PictureWebService {

private static final String AUTH\_HEADER\_KEY = "Authorization";

private static final String AUTH\_HEADER\_PREFIX = "Basic ";

private static final String USERNAME = "admin";

private static final String PASSWORD = "admin";

@Resource

private

WebServiceContext webServiceContext;

@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) throws IllegalQException {

if (request == null || request.isNull()) {

PictureServiceFault fault = PictureServiceFault.defaultInstance();

throw new IllegalQException("Parameter q cannot be null or empty", fault);

}

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) throws InsertingException,

InvalidCreatingParametersException, AuthException {

PictureServiceFault fault = PictureServiceFault.defaultInstance();

MessageContext messageContext = webServiceContext.getMessageContext();

if (!isAuth(messageContext))

{

fault.setMessage("Authentication error");

throw new AuthException("Invalid login-password", fault);

}

fault.setMessage("Invalid creating parameter");

if (name == null || name.trim().isEmpty()) {

fault.setMessage("Parameter name cannot be null or empty");

throw new InvalidCreatingParametersException("Invalid creating parameter", fault);

}

if (author == null || author.trim().isEmpty()) {

fault.setMessage("Parameter author cannot be null or empty");

throw new InvalidCreatingParametersException("Invalid creating parameter", fault);

}

if (year <= 0) {

fault.setMessage("Parameter year cannot be null or empty and less or equal to 0");

throw new InvalidCreatingParametersException("Invalid creating parameter", fault);

}

if (material == null || material.trim().isEmpty()) {

fault.setMessage("Parameter material cannot be null or empty");

throw new InvalidCreatingParametersException("Invalid creating parameter", fault);

}

if (height <= 0) {

fault.setMessage("Parameter height cannot be null or empty and less or equal to 0");

throw new InvalidCreatingParametersException("Invalid creating parameter", fault);

}

if (width <= 0) {

fault.setMessage("Parameter width cannot be null or empty and less or equal to 0");

throw new InvalidCreatingParametersException("Invalid creating parameter", fault);

}

PostgreSQLDAO dao = new PostgreSQLDAO();

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

}

@WebMethod(operationName = "updatePicture")

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

@WebParam(name = "q") MyRequest request) throws IllegalQException, IllegalIdException, InvalidEntityException, AuthException {

MessageContext messageContext = webServiceContext.getMessageContext();

if (!isAuth(messageContext))

{

PictureServiceFault fault = PictureServiceFault.defaultInstance();

fault.setMessage("Authentication error");

throw new AuthException("Invalid login-password", fault);

}

if (id == 0) {

PictureServiceFault fault = PictureServiceFault.defaultInstance();

fault.setMessage("Parameter id cannot be null or empty");

throw new IllegalIdException("Parameter id cannot be null or empty", fault);

}

if (request == null || request.isNull()) {

PictureServiceFault fault = PictureServiceFault.defaultInstance();

throw new IllegalQException("Parameter q cannot be null or empty", fault);

}

PostgreSQLDAO dao = new PostgreSQLDAO();

return dao.updatePicture(id, request);

}

/\*\*

\*

\* @param id

\* @return

\* @throws InvalidEntityException

\* @throws IllegalIdException

\* @throws AuthException

\*/

@WebMethod(operationName = "deletePicture")

public int deletePicture(@WebParam(name = "id") int id) throws InvalidEntityException, IllegalIdException, AuthException {

MessageContext messageContext = webServiceContext.getMessageContext();

if (!isAuth(messageContext))

{

PictureServiceFault fault = PictureServiceFault.defaultInstance();

fault.setMessage("Authentication error");

throw new AuthException("Invalid login-password", fault);

}

if (id == 0) {

PictureServiceFault fault = PictureServiceFault.defaultInstance();

fault.setMessage("Parameter id cannot be null or empty");

throw new IllegalIdException("Parameter id cannot be null or empty", fault);

}

PostgreSQLDAO dao = new PostgreSQLDAO();

return dao.deletePicture(id);

}

private boolean isAuth(MessageContext ctx) {

Map headers = (Map) ctx.get(MessageContext.HTTP\_REQUEST\_HEADERS);

if (!headers.containsKey(AUTH\_HEADER\_KEY)) {

return false;

}

List<String> authHeader = (List<String>) headers.get(AUTH\_HEADER\_KEY);

String authToken = authHeader.get(0);

if (authToken.isEmpty()) {

return false;

}

authToken = authToken.replaceFirst(AUTH\_HEADER\_PREFIX, "");

String decodedString = new String(Base64.decode(authToken));

StringTokenizer stringTokenizer = new StringTokenizer(decodedString, ":");

String username = stringTokenizer.nextToken();

String password = stringTokenizer.nextToken();

return username.equals(USERNAME) && password.equals(PASSWORD);

}

}

Листинг 3.3 – Файл PostgreSQLDAO.java

package com.sumingjie.service;

import com.sumingjie.serviceexception.InsertingException;

import com.sumingjie.serviceexception.InvalidEntityException;

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 InsertingException {

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

PreparedStatement preparedStatement = null;

int id = 0;

try {

preparedStatement = this.connection.prepareStatement(sql, Statement.RETURN\_GENERATED\_KEYS);

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();

ResultSet generatedKeys = preparedStatement.getGeneratedKeys();

if (generatedKeys.next()) {

id = (int) generatedKeys.getLong(1);

}

} catch (SQLException e) {

e.printStackTrace();

}

if (id == 0) {

PictureServiceFault fault = PictureServiceFault.defaultInstance();

fault.setMessage("Error During creation entity");

throw new InsertingException("Error During creation entity", fault);

}

return id;

}

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

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

PreparedStatement preparedStatement = null;

int affectedRows = 0;

try {

preparedStatement = this.connection.prepareStatement(sql);

preparedStatement.setInt(1, id);

affectedRows = preparedStatement.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

if (affectedRows == 0) {

PictureServiceFault fault = PictureServiceFault.defaultInstance();

fault.setMessage("Invalid entity");

throw new InvalidEntityException("Invalid entity", fault);

}

return affectedRows;

}

public int deletePicture(int id) throws InvalidEntityException {

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

PreparedStatement preparedStatement = null;

int affectedRows = 0;

try {

preparedStatement = this.connection.prepareStatement(sql);

preparedStatement.setInt(1, id);

affectedRows = preparedStatement.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

if (affectedRows == 0) {

PictureServiceFault fault = PictureServiceFault.defaultInstance();

fault.setMessage("Invalid entity");

throw new InvalidEntityException("Invalid entity", fault);

}

return affectedRows;

}

private List<Picture> executeQuery(String sql) {

List<Picture> pictures;

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();

}

}

Листинг 3.4 – Файл PictureServiceFault.java

package com.sumingjie.service;

public class PictureServiceFault {

private static final String DEFAULT\_MESSAGE = "Parameter q cannot be null or empty";

private String message;

public String getMessage() {

return message;

}

public void setMessage(String message) {

this.message = message;

}

public static PictureServiceFault defaultInstance() {

PictureServiceFault fault = new PictureServiceFault();

fault.message = DEFAULT\_MESSAGE;

return fault;

}

}

Листинг 3.5 – Файл IllegalIdException.java

package com.sumingjie.serviceexception;

import com.sumingjie.service.PictureServiceFault;

import javax.xml.ws.WebFault;

@WebFault(faultBean = "com.sumingjie.service.PictureServiceFault")

public class IllegalIdException extends Exception {

private static final long serialVersionUID = -6647544772732631047L;

private final PictureServiceFault fault;

public IllegalIdException(String message, PictureServiceFault fault) {

super(message);

this.fault = fault;

}

public IllegalIdException(String message, PictureServiceFault fault, Throwable cause) {

super(message, cause);

this.fault = fault;

}

public PictureServiceFault getFaultInfo() {

return fault;}}

Листинг 3.6 – Файл IllegalQException.java

package com.sumingjie.serviceexception;

import com.sumingjie.service.PictureServiceFault;

import javax.xml.ws.WebFault;

@WebFault(faultBean = "com.sumingjie.service.PictureServiceFault")

public class IllegalQException extends Exception {

private static final long serialVersionUID = -6647544772732631047L;

private final PictureServiceFault fault;

public IllegalQException(String message, PictureServiceFault fault) {

super(message);

this.fault = fault;

}

public IllegalQException(String message, PictureServiceFault fault, Throwable cause) {

super(message, cause);

this.fault = fault;

}

public PictureServiceFault getFaultInfo() {

return fault; }}

Листинг 3.7 – Файл InvalidCreatingParametersException.java

package com.sumingjie.serviceexception;

import com.sumingjie.service.PictureServiceFault;

import javax.xml.ws.WebFault;

@WebFault(faultBean = "com.sumingjie.service.PictureServiceFault")

public class InvalidCreatingParametersException extends Exception {

private static final long serialVersionUID = -6647544772732631047L;

private final PictureServiceFault fault;

public InvalidCreatingParametersException(String message, PictureServiceFault fault) {

super(message);

this.fault = fault;

}

public InvalidCreatingParametersException(String message, PictureServiceFault fault, Throwable cause) {

super(message, cause);

this.fault = fault;

}

public PictureServiceFault getFaultInfo() {

return fault; } }

Листинг 3.8 – Файл InsertingException.java

package com.sumingjie.serviceexception;

import com.sumingjie.service.PictureServiceFault;

import javax.xml.ws.WebFault;

@WebFault(faultBean = "com.sumingjie.service.PictureServiceFault")

public class InsertingException extends Exception {

private static final long serialVersionUID = -6647544772732631047L;

private final PictureServiceFault fault;

public InsertingException(String message, PictureServiceFault fault) {

super(message);

this.fault = fault;

}

public InsertingException(String message, PictureServiceFault fault, Throwable cause) {

super(message, cause);

this.fault = fault;

}

public PictureServiceFault getFaultInfo() {

return fault; } }

Листинг 3.9 – Файл InvalidEntityException.java

package com.sumingjie.serviceexception;

import com.sumingjie.service.PictureServiceFault;

import javax.xml.ws.WebFault;

@WebFault(faultBean="com.sumingjie.service.PictureServiceFault")

public class InvalidEntityException extends Exception{

private static final long serialVersionUID = -6647544772732631047L;

private final PictureServiceFault fault;

public InvalidEntityException(String message,PictureServiceFault fault){

super(message);

this.fault = fault;

}

public InvalidEntityException(String message,PictureServiceFault fault,Throwable cause){

super(message,cause);

this.fault = fault;

}

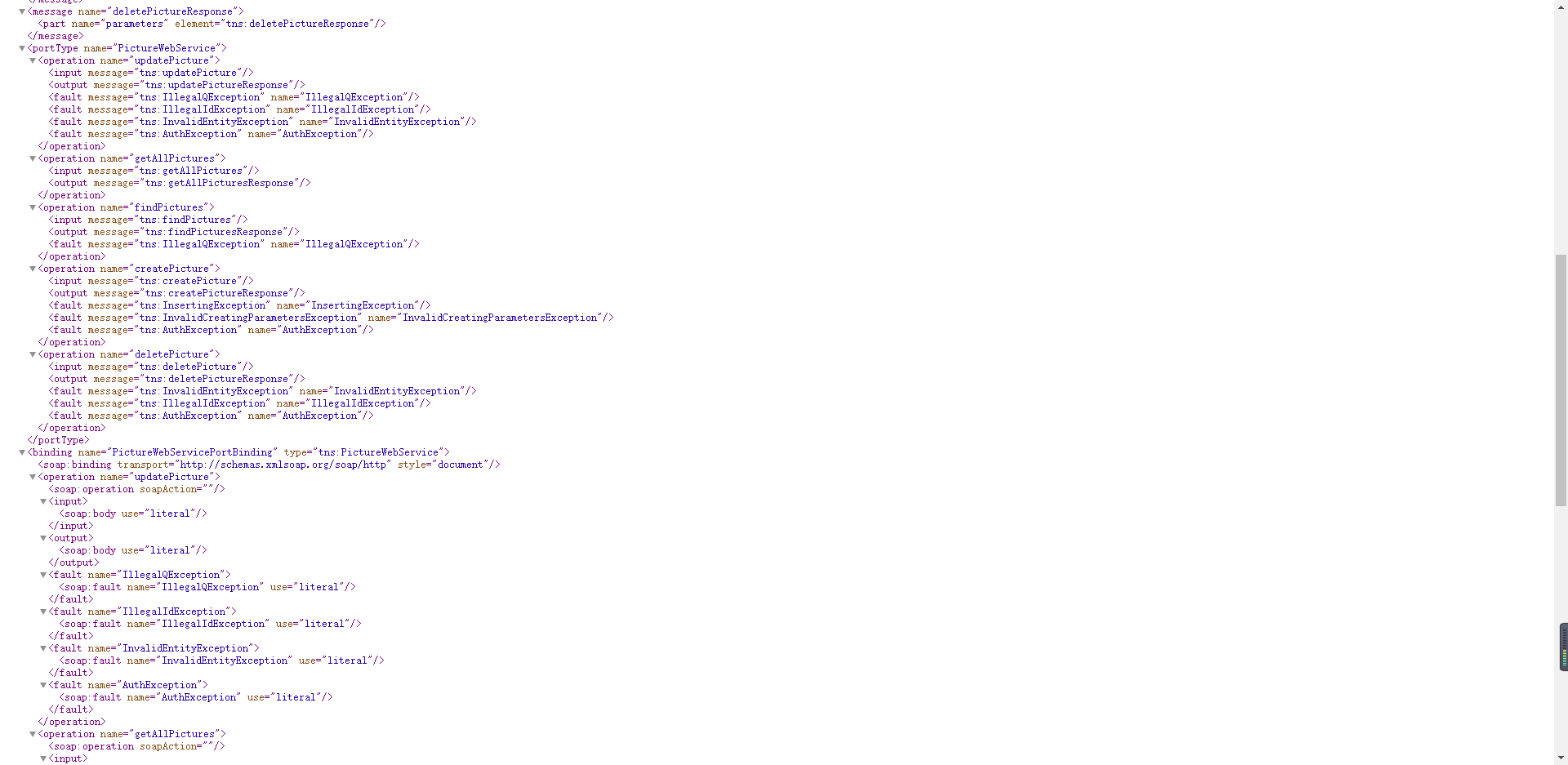
public PictureServiceFault getFaultInfo(){

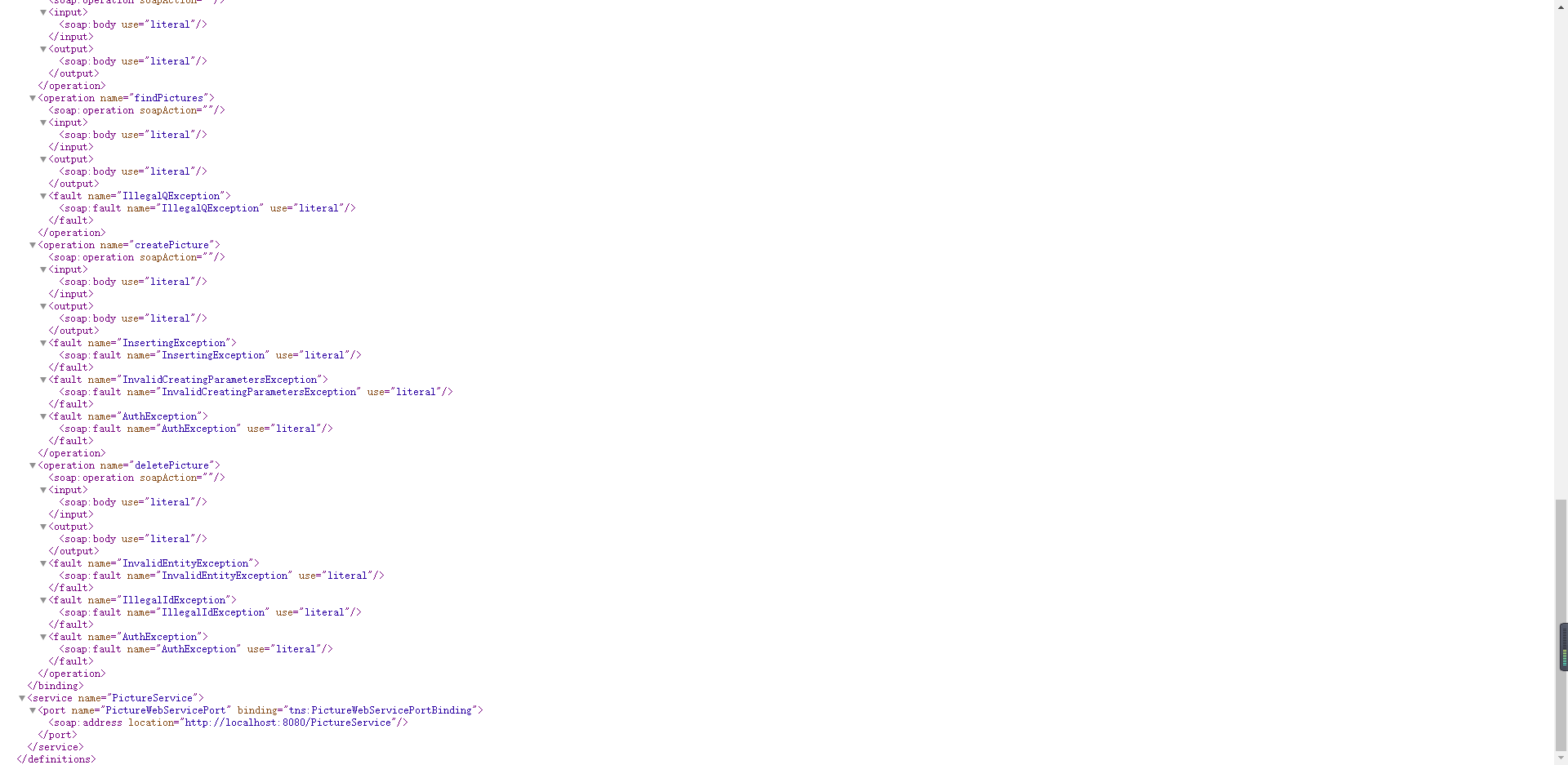
return fault; } }

WSDL сервиса, представленный в листинге 3.10, доступен по адресу

http://localhost:8080/PictureService?wsdl

Листинг 3.10 – WSDL сервиса





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, PictureServiceFault.java, InvalidEntityException.java, InvalidCreatingParametersException.java, IllegalQException.java, IllegalIdException.java, InsertingException.java и был

сгенерирован следующей командой:

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

Исходный код WebServiceClient.java представлен в листинге 3.11. В этом классе последовательно выполняются запросы createPicture, updatePicture и deletePicture. Результат выполнения приведен на рисунке 3.1.

Листинг 3.11 – WebServiceClient.java

package com.sumingjie.client;

import java.net.MalformedURLException;

import java.net.URL;

import java.util.Collections;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import javax.xml.ws.BindingProvider;

import javax.xml.ws.handler.MessageContext;

import org.postgresql.util.Base64;

public class WebServiceClient {

private static final String USERNAME = "admin";

private static final String PASSWORD = "admin";

private 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 {

String authToken = "Basic " + Base64.encodeBytes((USERNAME + ":" + PASSWORD).getBytes());

URL url = new URL("http://localhost:8080/PictureService?wsdl");

PictureService pictureService = new PictureService(url);

PictureWebService pictureWebService = pictureService.getPictureWebServicePort();

Map<String, List<String>> credentials;

credentials = new HashMap<>();

credentials.put("Authorization", Collections.singletonList(authToken));

BindingProvider bindingProvider = (BindingProvider) pictureWebService;

bindingProvider.getRequestContext().put(MessageContext.HTTP\_REQUEST\_HEADERS, credentials);

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

getStatus(pictureWebService);

System.out.println("Query with auth: createPicture?name=Богатыри&author=Виктор Михайлович Васнецов");

int creatingPictureId = 0;

try {

creatingPictureId = pictureWebService.createPicture(

"Богатыри",

"Виктор Михайлович Васнецов",

0,

"",

0,

0);

} catch (InsertingException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

} catch (InvalidCreatingParametersException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

} catch (AuthException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

}

System.out.println("Response: " + creatingPictureId);

System.out.println();

credentials.remove("Authorization");

System.out.println("Query without auth: createPicture?name=Богатыри&author=Виктор Михайлович Васнецов&" +

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

int id = 0;

try {

id = pictureWebService.createPicture(

"Богатыри",

"Виктор Михайлович Васнецов",

1881,

"Маслянные краски",

295.3f,

446);

} catch (InsertingException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

} catch (InvalidCreatingParametersException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

} catch (AuthException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

}

System.out.println("Response: " + id);

System.out.println();

getStatus(pictureWebService);

credentials.put("Authorization", Collections.singletonList(authToken));

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

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

id = 0;

try {

id = pictureWebService.createPicture(

"Богатыри",

"Виктор Михайлович Васнецов",

1881,

"Маслянные краски",

295.3f,

446);

} catch (InsertingException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

} catch (InvalidCreatingParametersException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

} catch (AuthException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

}

System.out.println("Response: " + id);

System.out.println();

getStatus(pictureWebService);

System.out.println("Query with auth: updatePicture?id=10");

MyRequest myRequest = new MyRequest();

try {

creatingPictureId = pictureWebService.updatePicture(id, myRequest);

} catch (IllegalIdException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

} catch (IllegalQException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

} catch (InvalidEntityException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

} catch (AuthException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

}

System.out.println("Response: " + creatingPictureId);

System.out.println();

System.out.println("Query with auth: updatePicture?id=111&name=My own picture&author=ITMO&year=2018");

myRequest.init();

myRequest.setName("My own picture");

myRequest.setAuthor("ITMO");

myRequest.setYear(2018);

try {

creatingPictureId = pictureWebService.updatePicture(111, myRequest);

} catch (IllegalIdException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

} catch (IllegalQException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

} catch (InvalidEntityException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

} catch (AuthException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

}

System.out.println("Response: " + creatingPictureId);

System.out.println();

credentials.remove("Authorization");

System.out.println("Query without auth: updatePicture?id=10&name=My own picture&author=ITMO&year=2018");

try {

creatingPictureId = pictureWebService.updatePicture(10, myRequest);

} catch (IllegalIdException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

} catch (IllegalQException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

} catch (InvalidEntityException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

} catch (AuthException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

}

System.out.println("Response: " + creatingPictureId);

System.out.println();

getStatus(pictureWebService);

credentials.put("Authorization", Collections.singletonList(authToken));

System.out.println("Query with auth: updatePicture?id=10&name=My own picture&author=ITMO&year=2018");

try {

creatingPictureId = pictureWebService.updatePicture(10, myRequest);

} catch (IllegalIdException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

} catch (IllegalQException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

} catch (InvalidEntityException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

} catch (AuthException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

}

System.out.println("Response: " + creatingPictureId);

System.out.println();

getStatus(pictureWebService);

System.out.println("Query with auth: deletePicture?id=111");

try {

creatingPictureId = pictureWebService.deletePicture(111);

} catch (IllegalIdException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

} catch (InvalidEntityException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

} catch (AuthException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

}

System.out.println("Response: " + creatingPictureId);

System.out.println();

credentials.remove("Authorization");

System.out.println("Query without auth: deletePicture?id=" + id);

try {

creatingPictureId = pictureWebService.deletePicture(id);

} catch (IllegalIdException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

} catch (InvalidEntityException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

} catch (AuthException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

}

System.out.println("Response: " + creatingPictureId);

System.out.println();

getStatus(pictureWebService);

credentials.put("Authorization", Collections.singletonList(authToken));

System.out.println("Query with auth: deletePicture?id=" + id);

try {

creatingPictureId = pictureWebService.deletePicture(id);

} catch (IllegalIdException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

} catch (InvalidEntityException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

} catch (AuthException e) {

System.out.println("Message: " + e.getMessage());

System.out.println("FaultInfo: " + e.getFaultInfo().getMessage());

}

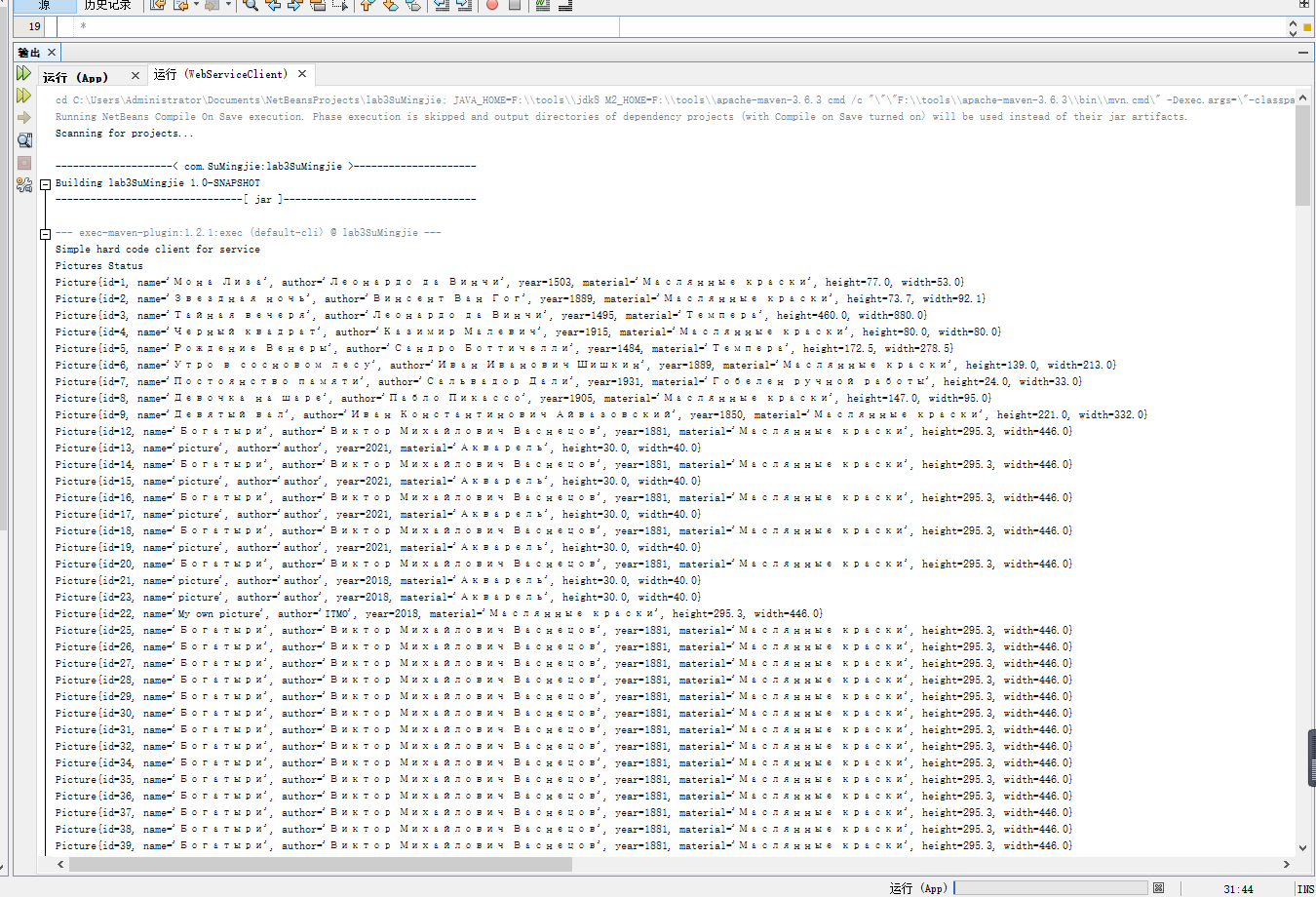
System.out.println("Response: " + creatingPictureId);

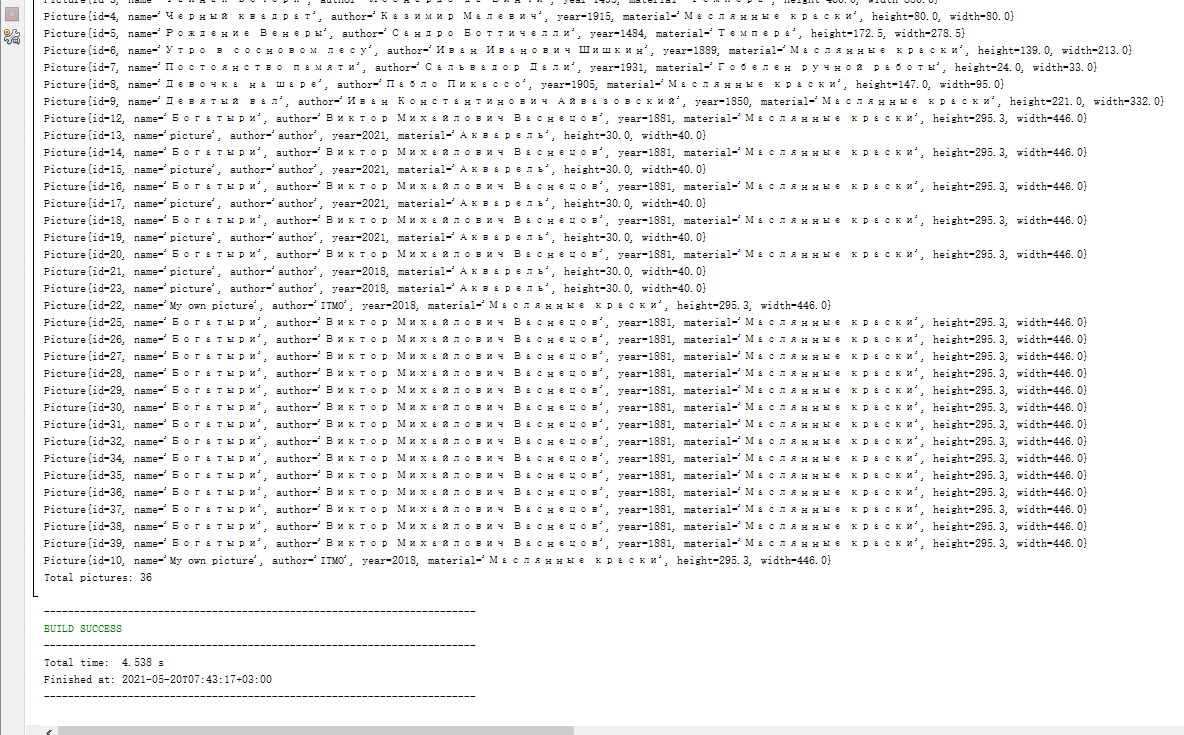
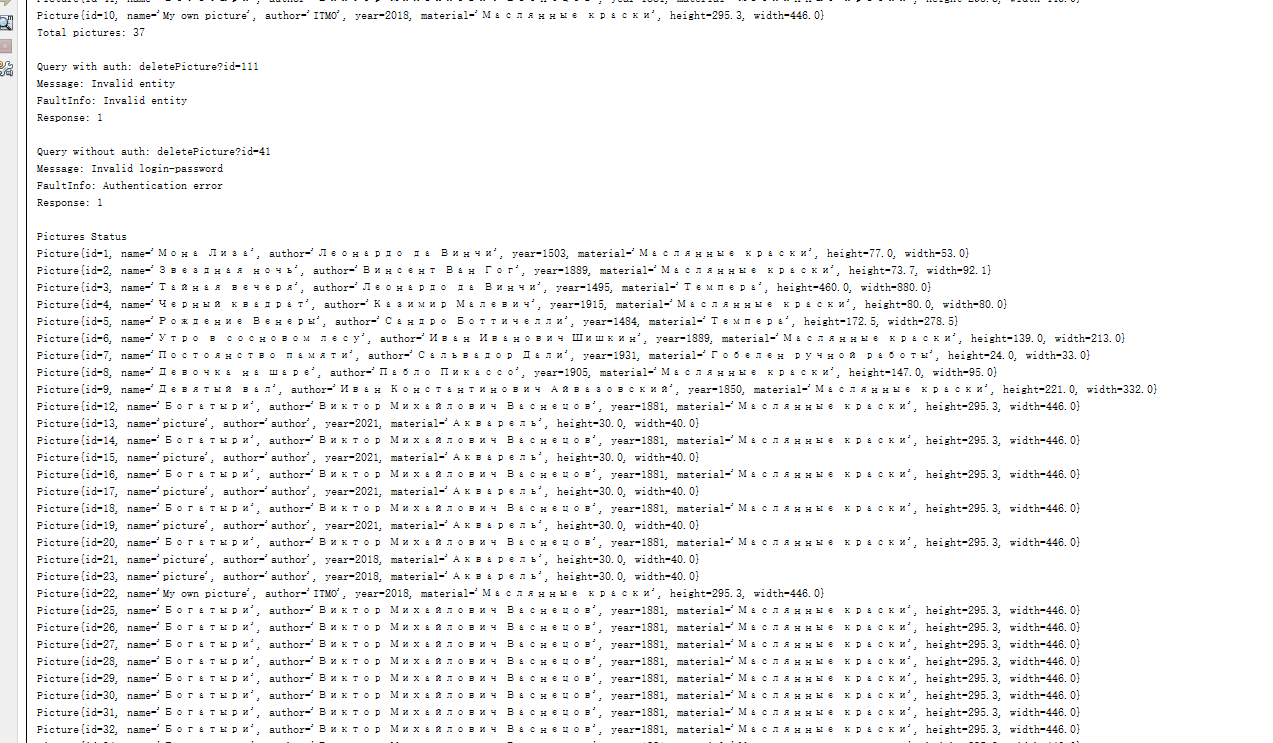
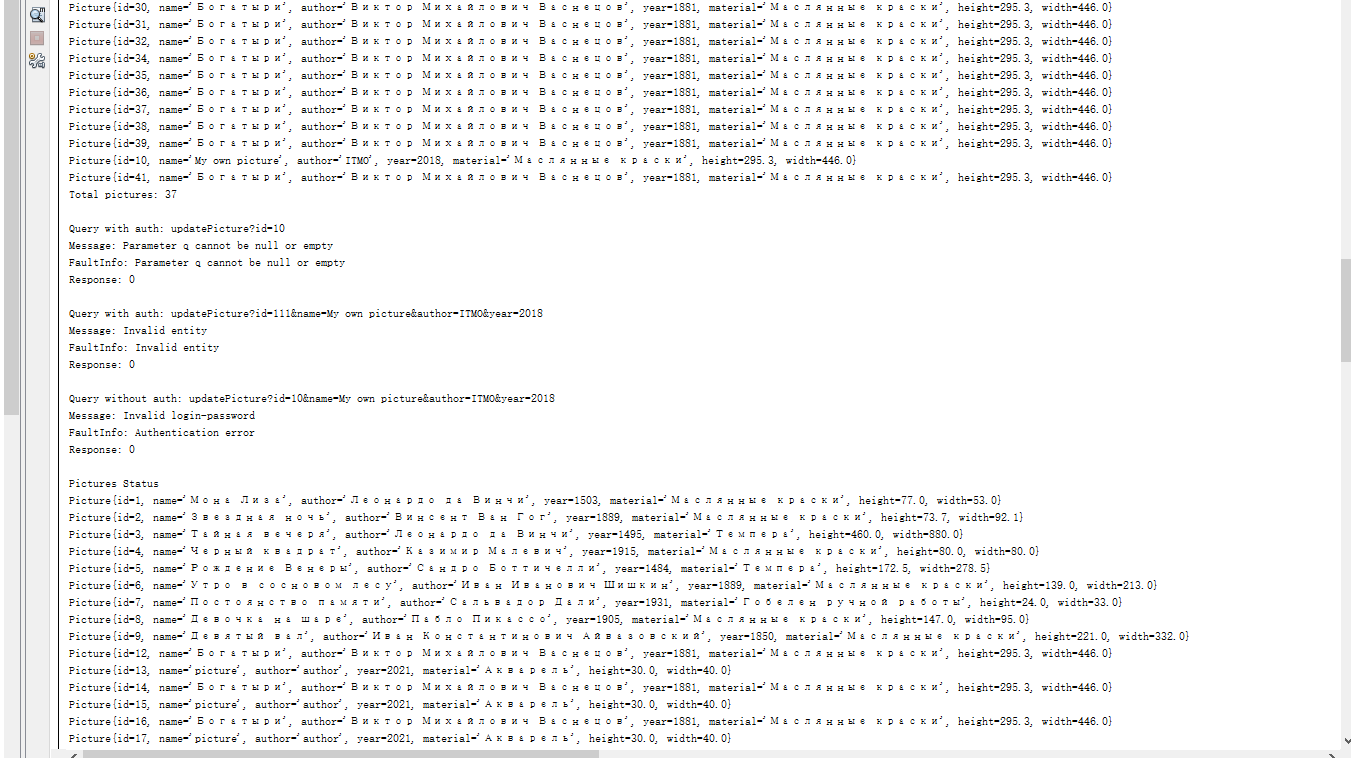
System.out.println();

getStatus(pictureWebService);

}

}





**Вывод**: в ходе выполнения работы, основываясь на информации из раздела 2.8, была добавлена обработка ошибок в сервис. В соответствии с изменениями сервиса также было обновлено и клиентское приложение.