**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 № 2**

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

Студент: Калинин Даниил Леонидович

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

Этапы выполнения:

1. Для обработки ошибок были созданы несколько исключений:

EmptyRequestException – при получении от клиента объекта Person, в котором все поля пусты

@WebFault(faultBean = "web\_services.errors.faultbeans.PersonServiceFault")  
public class EmptyRequestException extends ServerException {

private final PersonServiceFault fault;  
  
 public EmptyRequestException(String message, PersonServiceFault fault) {  
 super(message);  
 this.fault = fault;  
 }  
  
 public EmptyRequestException(String message) {  
 super(message);  
 this.fault = new PersonServiceFault();  
 }  
  
 public EmptyRequestException(String message, PersonServiceFault fault, Throwable cause) {  
 super(message, cause);  
 this.fault = fault;  
 }  
 public PersonServiceFault getFaultInfo() {  
 return fault;  
 }  
}

PersonDoesNotExistException – при попытке удалить или изменить несуществующую запись

@WebFault(faultBean = "web\_services.errors.faultbeans.PersonServiceFault")  
public class PersonDoesNotExistException extends ServerException {

private final PersonServiceFault fault;  
  
 public PersonDoesNotExistException(String message, PersonServiceFault fault) {  
 super(message);  
 this.fault = fault;  
 }  
 public PersonDoesNotExistException(String message, PersonServiceFault fault, Throwable cause) {  
 super(message, cause);  
 this.fault = fault;  
 }  
   
 public PersonServiceFault getFaultInfo() {  
 return fault;  
 }  
}

SQLConvertException – для ошибки при конвертации объекта в SQL-запрос

@WebFault(faultBean = "web\_services.errors.faultbeans.PersonServiceFault")  
public class SQLConvertException extends ServerException {  
   
 private final PersonServiceFault fault;  
  
 public SQLConvertException(String message) {  
 super(message);  
 this.fault = new PersonServiceFault();  
 }  
 public SQLConvertException(String message, PersonServiceFault fault, Throwable cause) {  
 super(message, cause);  
 this.fault = fault;  
 }  
 public PersonServiceFault getFaultInfo() {  
 return fault;  
 }  
}

Все они наследуются от ServerException, что упрощает их конечную обработку

@WebFault(faultBean = "web\_services.errors.faultbeans.PersonServiceFault")  
public class ServerException extends Exception {  
  
 private static final long *serialVersionUID* = -6647544772732631047L;  
 private final PersonServiceFault fault;  
  
 public ServerException(String message) {  
 super(message);  
 this.fault = new PersonServiceFault();  
 }  
 public ServerException(String message, PersonServiceFault fault, Throwable cause) {  
 super(message, cause);  
 this.fault = fault;  
 this.initCause(cause);  
 }  
 public PersonServiceFault getFaultInfo() {  
 return fault;  
 }  
}

1. В DAO также были добавлены методы для выполнения update – операций

public class MariaDBDAO {  
  
 private Person[] processQuery(ResultSet rs){  
 Person[] persons\_array = new Person[0];  
 List<Person> persons = new ArrayList<>();  
 try {  
 while (rs.next()) {  
 String name = rs.getString("name");  
 String surname = rs.getString("surname");  
 int age = rs.getInt("age");  
 String country = rs.getString("country");  
 String gender = rs.getString("gender");  
 int id = rs.getInt("id");  
 Person person = new Person(name, surname, age, country, gender);  
 person.setId(id);  
 persons.add(person);  
 }  
 persons\_array = new Person[persons.size()];  
 persons.toArray(persons\_array);  
 return persons\_array;  
 } catch (SQLException ex) {  
 Logger.*getLogger*(MariaDBDAO.class.getName()).log(Level.*SEVERE*, null, ex);  
 }  
 return persons\_array;  
 }  
  
 public Person[] getPersons() {  
 Person[] persons\_array = new Person[0];  
 try (Connection connection = ConnectionUtil.*getConnection*()) {  
 Statement stmt = connection.createStatement();  
 ResultSet rs = stmt.executeQuery("select \* from persons");  
 persons\_array = processQuery(rs);  
 } catch (SQLException ex) {  
 Logger.*getLogger*(MariaDBDAO.class.getName()).log(Level.*SEVERE*, null, ex);  
 }  
 return persons\_array;  
 }  
  
 public String executeUpdateQuery(String sqlQuery, String message){  
 try (Connection connection = ConnectionUtil.*getConnection*()) {  
 Statement stmt = connection.createStatement();  
 int res = stmt.executeUpdate(sqlQuery);  
 String result = "Successfully " + message + " " + res + " person(s)";  
 System.*out*.println(result);  
 return result;  
 } catch (Exception ex) {  
 Logger.*getLogger*(MariaDBDAO.class.getName()).log(Level.*SEVERE*, null, ex);  
 return ("Error! " + ex.getMessage());  
 }  
 }  
  
 public Person[] getPersonsBySqlQuery(String sqlQuery) {  
 Person[] persons\_array = new Person[0];  
 try (Connection connection = ConnectionUtil.*getConnection*()) {  
 Statement stmt = connection.createStatement();  
 ResultSet rs = stmt.executeQuery(sqlQuery);  
 persons\_array = processQuery(rs);  
 } catch (SQLException ex) {  
 Logger.*getLogger*(MariaDBDAO.class.getName()).log(Level.*SEVERE*, null, ex);  
 }  
 return persons\_array;  
 }  
  
 public Boolean checkIfPersonExists(String sqlQuery) {  
 try (Connection connection = ConnectionUtil.*getConnection*()) {  
 Statement stmt = connection.createStatement();  
 ResultSet rs = stmt.executeQuery(sqlQuery);  
 return processQuery(rs).length > 0;  
 } catch (SQLException ex) {  
 Logger.*getLogger*(MariaDBDAO.class.getName()).log(Level.*SEVERE*, null, ex);  
 }  
 return false;  
 }

1. В клиенте были добавлены диалоги для ввода пользовательских данных и вызова новых операций на сервере
2. public class OperationHandler {  
     
    public OperationHandler(PersonService personService) {  
    this.personService = personService;  
    this.inputReader = new ConsoleInputReader();  
    }  
     
    ConsoleInputReader inputReader;  
    PersonService personService;  
     
    public void operationSelectDialog() throws EmptyRequestException, SQLConvertException, PersonDoesNotExistException {  
    BufferedReader reader = new BufferedReader(new InputStreamReader(System.*in*));  
    System.*out*.println("Select operation: \n"  
    + "\t1 - search\n" +  
    "\t2 - add\n" +  
    "\t3 - delete\n" +  
    "\t4 - update\n" +  
    "\t5 - exit\n");  
    try {  
    int choice = Integer.*parseInt*(reader.readLine());  
    switch (choice){  
    case 1:  
    selectQuery();  
    break;  
    case 2:  
    insertQuery();  
    break;  
    case 3:  
    deleteQuery();  
    break;  
    case 4:  
    updateQuery();  
    break;  
    case 5:  
    System.*exit*(0);  
    default:  
    throw new IllegalStateException("Unexpected value: " + choice);  
    }  
    } catch (IOException ioException){  
    System.*out*.println("IO Exception!");  
    System.*exit*(-1);  
    } catch (NumberFormatException |  
    IllegalStateException illegalStateException){  
    System.*out*.println("Incorrect choice! Please try again");  
    operationSelectDialog();  
    }  
    }  
     
    public void selectQuery() throws EmptyRequestException, SQLConvertException, PersonDoesNotExistException {  
    Person query = inputReader.readSearchInput();  
    PersonArray persons = personService.getPersonWebServicePort().selectPersons(query);  
    for (Person person : persons.getItem()) {  
    System.*out*.println("name: " + person.getName() +  
    ", surname: " + person.getSurname() + ", age: " + person.getAge() +  
    ", country: " + person.getCountry() + ", gender: " + person.getGender());  
    }  
    System.*out*.println("Total persons: " + persons.getItem().size());  
    }  
     
    public void insertQuery() throws SQLConvertException, EmptyRequestException {  
    Person query = inputReader.readInsertInput();  
    String result = personService.getPersonWebServicePort().insertPerson(query);  
    if (result != null)  
    System.*out*.println(result);  
    else  
    System.*out*.println("Server error");  
    }  
     
    public void updateQuery() throws EmptyRequestException, SQLConvertException, PersonDoesNotExistException {  
    int id = inputReader.readNum("Enter the ID of the person you want to change");  
    System.*out*.println("Insert updated information");  
    Person newPerson = inputReader.readInsertInput();  
    String result = personService.getPersonWebServicePort().updatePersonByID(id, newPerson);  
    if (result != null)  
    System.*out*.println(result);  
    else  
    System.*out*.println("Server error");  
    }  
     
    public void deleteQuery() throws SOAPFaultException, EmptyRequestException, SQLConvertException, PersonDoesNotExistException {  
    int id = inputReader.readNum("Enter the ID of the person you want to delete");  
    String result = personService.getPersonWebServicePort().deletePersonByID(id);  
    if (result != null)  
    System.*out*.println(result);  
    else  
    System.*out*.println("Server error");  
    }  
   }

public class ConsoleInputReader {  
  
 BufferedReader reader = new BufferedReader(new InputStreamReader(System.*in*));  
  
 public int readNum(String message){  
 System.*out*.println(message);  
 try {  
 String age = reader.readLine();  
 if (age.equals("")) return -1;  
 else {  
 try {  
 return Integer.*parseInt*(age);  
 } catch (NumberFormatException e) {  
 System.*out*.println("Number is incorrect!");  
 return readNum(message);  
 }  
 }  
 } catch (IOException e){  
 System.*out*.println("Something went wrong, please try again");  
 return readNum(message);  
 }  
 }  
  
 public Person readInsertInput() {  
 Person query = new Person();  
 try {  
 System.*out*.println("Enter name: ");  
 query.setName(reader.readLine());  
 System.*out*.println("Enter surname: ");  
 query.setSurname(reader.readLine());  
 query.setAge(readNum("Enter age: "));  
 System.*out*.println("Enter country: ");  
 query.setCountry(reader.readLine());  
 System.*out*.println("Enter gender: ");  
 query.setGender(reader.readLine());  
 return query;  
 }  
 catch (IOException e) {  
 System.*out*.println(e.toString());  
 }  
 return query;  
 }  
  
 public Person readSearchInput() {  
 Person query = new Person();  
 try {  
 query.setId(readNum("Enter ID: "));  
 System.*out*.println("Enter name: ");  
 query.setName(reader.readLine());  
 System.*out*.println("Enter surname: ");  
 query.setSurname(reader.readLine());  
 query.setAge(readNum("Enter age: "));  
 System.*out*.println("Enter country: ");  
 query.setCountry(reader.readLine());  
 System.*out*.println("Enter gender: ");  
 query.setGender(reader.readLine());  
 return query;  
 }  
 catch (IOException e) {  
 System.*out*.println(e.toString());  
 }  
 return query;  
 }  
}

На стороне клиента была добавлена обработка всех генерируемых сервером исключений

package console\_client;  
  
import wsdl\_autogenerated.\*;  
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
  
public class OperationHandler {  
  
 public OperationHandler(PersonService personService) {  
 this.personService = personService;  
 this.inputReader = new ConsoleInputReader();  
 }  
  
 ConsoleInputReader inputReader;  
 PersonService personService;  
  
 public void operationSelectDialog() throws ServerException {  
 BufferedReader reader = new BufferedReader(new InputStreamReader(System.*in*));  
 System.*out*.println("Select operation: \n"  
 + "\t1 - search\n" +  
 "\t2 - add\n" +  
 "\t3 - delete\n" +  
 "\t4 - update\n" +  
 "\t5 - exit\n");  
 try {  
 int choice = Integer.*parseInt*(reader.readLine());  
 switch (choice){  
 case 1:  
 selectQuery();  
 break;  
 case 2:  
 insertQuery();  
 break;  
 case 3:  
 deleteQuery();  
 break;  
 case 4:  
 updateQuery();  
 break;  
 case 5:  
 System.*exit*(0);  
 default:  
 throw new IllegalStateException("Unexpected value: " + choice);  
 }  
 } catch (IOException ioException){  
 System.*out*.println("IO Exception!");  
 System.*exit*(-1);  
 } catch (NumberFormatException |  
 IllegalStateException illegalStateException){  
 System.*out*.println("Incorrect choice! Please try again");  
 operationSelectDialog();  
 }  
 }  
  
 public void selectQuery() throws ServerException{  
 Person query = inputReader.readSearchInput();  
 PersonArray persons = personService.getPersonWebServicePort().selectPersons(query);  
 for (Person person : persons.getItem()) {  
 System.*out*.println("name: " + person.getName() +  
 ", surname: " + person.getSurname() + ", age: " + person.getAge() +  
 ", country: " + person.getCountry() + ", gender: " + person.getGender());  
 }  
 System.*out*.println("Total persons: " + persons.getItem().size());  
 }  
  
 public void insertQuery() throws ServerException{  
 Person query = inputReader.readInsertInput();  
 String result = personService.getPersonWebServicePort().insertPerson(query);  
 if (result != null)  
 System.*out*.println(result);  
 else  
 System.*out*.println("Server error");  
 }  
  
 public void updateQuery() throws ServerException{  
 int id = inputReader.readNum("Enter the ID of the person you want to change");  
 System.*out*.println("Insert updated information");  
 Person newPerson = inputReader.readInsertInput();  
 String result = personService.getPersonWebServicePort().updatePersonByID(id, newPerson);  
 if (result != null)  
 System.*out*.println(result);  
 else  
 System.*out*.println("Server error");  
 }  
  
 public void deleteQuery() throws ServerException {  
 int id = inputReader.readNum("Enter the ID of the person you want to delete");  
 String result = personService.getPersonWebServicePort().deletePersonByID(id);  
 if (result != null)  
 System.*out*.println(result);  
 else  
 System.*out*.println("Server error");  
 }  
}

Дополнительно:

Выводы:

В ходе выполнения данной лабораторной работы были реализована обработка ошибок, возникающих на сервере, с соответствующим оповещением на клиентской стороне

Вопросы:

Ссылка на GitHub:

https://github.com/KalininDL/web\_services\_spring