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:

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.

Этапы выполнения:

1. Для выполнения задание в сервис были добавлены endpoint-ы для новых операций:

```
@WebMethod(operationName = "deletePersonByID")
public String deletePersonByID(int id) throws PersonDoesNotExistException,
EmptyRequestException, SQLConvertException {
    try {
        if (!dao.checkIfPersonExists(sqlQueryBuilder.buildSelectQuery(new
Person(id))))
            throw new PersonDoesNotExistException("Person with specified
parameters does not exist!", fault);
        return dao.executeUpdateQuery(sqlQueryBuilder.buildDeleteQuery(id),
"deleted");
    } catch (EmptyRequestException | SQLConvertException e){
        throw e;
    }
}

@WebMethod(operationName = "insertPerson")
public String insertPerson(Person person) throws SQLConvertException,
EmptyRequestException {
    try {
        String result =
dao.executeUpdateQuery(sqlQueryBuilder.buildInsertQuery(person), "added");
        Person[] created =
dao.getPersonsBySqlQuery(sqlQueryBuilder.buildSelectQuery(person));
        return "New person with id: " + created[0].getId() + " was
successfully created!";
    } catch (SQLConvertException | EmptyRequestException e){
        throw e;
    }
}

@WebMethod(operationName = "updatePersonByID")
public String updatePersonByID(int id, Person updated) throws
PersonDoesNotExistException, EmptyRequestException, SQLConvertException {
    try {
        if (!dao.checkIfPersonExists(sqlQueryBuilder.buildSelectQuery(new
Person(id))))
            throw new PersonDoesNotExistException("Person with specified
```

```
parameters does not exist!", fault);
        return dao.executeUpdateQuery(sqlQueryBuilder.buildUpdateQuery(id,
updated), "updated");
    } catch (EmptyRequestException | SQLConvertException e){
        throw e;
    }
}
```

2. 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;
    }
```

3. В клиенте были добавлены диалоги для ввода пользовательских данных и вызова новых операций на сервере

```
4. 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;
    }
}
```

Выводы:

В ходе выполнения данной лабораторной работы был дополнен код программы, созданной в ходе первой лабораторной работы, для выполнения CRUD-операций над таблицей в базе данных

Вопросы:

Ссылка на GitHub:

https://github.com/KalininDL/web_services_spring