Федеральное государственное автономное образовательное учреждение высшего образования

Университет ИТМО

Дисциплина: Технологии веб-сервисов

Лабораторная работа 2

Выполнили:
Кривоносов Егор Дмитриевич

Группа: P4214

Преподаватель:
Сафронов Андрей Геннадьевич

2024 г.

Санкт-Петербург

Оглавление

Техническое 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.

Постановка задачи

Задача заключалась в расширении существующего SOAP-сервиса для обеспечения полного CRUD-функционала (Create, Read, Update, Delete). Дополнительно требовалось адаптировать клиентское приложение для взаимодействия с новыми методами сервиса, позволяя пользователю создавать, обновлять и удалять записи в базе данных через консольный интерфейс.

Этапы выполнения

1. **Расширение SOAP-сервиса**
    - Добавлены методы createPerson, updatePerson, deletePersonById и findPersonById в сервис для реализации CRUD-операций.
    - Методы реализованы с использованием JPA для управления данными в базе данных PostgreSQL.
    - Метод findPersonById позволяет получать информацию по конкретному персону.
    - Добавлена проверка на существование персона перед обновлением и удалением.
    - Удален модуль J2EE из проекта.
2. **Изменения в клиентском приложении**
    - В клиентское приложение добавлены команды create, update, delete, и findById для взаимодействия с новыми методами сервиса.

3. **Тестирование и отладка**
    - Проведено тестирование нового функционала сервиса и клиентского приложения на корректность выполнения CRUD-операций.
    - Проверены сценарии успешного и неуспешного выполнения операций, такие как попытки обновления или удаления несуществующих персонов.

Комментарии к архитектурным и функциональным аспектам реализации

- **SOAP-сервис** реализован с использованием аннотаций @WebService и @WebMethod, что позволило удобно описывать новые методы для работы с персонами.
- **Модульное разделение**: каждый модуль приложения (standalone, client) изолирован, что облегчает тестирование и поддержку кода.

Пример новых методов веб-сервиса для CRUD:

```
    @WebMethod
    public Person findPersonById(@WebParam(name = "id") int id) {
        return personService.readPerson(id);
    }

    @WebMethod
    public int createPerson(@WebParam(name = "personDto") PersonDto personDto) {
        return personService.createPerson(personDto);
    }

    @WebMethod
    public boolean updatePerson(@WebParam(name = "id") int id, @WebParam(name =
"personDto") PersonDto personDto) {
        return personService.updatePerson(id, personDto);
    }

    @WebMethod
    public boolean deletePersonById(@WebParam(name = "id") int id) {
        return personService.deletePersonById(id);
    }
```

Выполнение через entityManager взаимодействия с БД (такие запросы как CREATE, UPDATE, DELETE выполняются как транзакции):

```
    public Person readPerson(int id) {
        try (EntityManager entityManager =
entityManagerFactory.createEntityManager()) {
            return entityManager.find(Person.class, id);
        }
    }
```

```
    public int createPerson(Person person) {
        try (EntityManager entityManager =
entityManagerFactory.createEntityManager()) {
            entityManager.getTransaction().begin();
            entityManager.persist(person);
            entityManager.getTransaction().commit();
            return person.getId();
        }
    }

    public boolean updatePerson(int id, Person newDetails) {
        try (EntityManager entityManager =
entityManagerFactory.createEntityManager()) {
            entityManager.getTransaction().begin();
            Person existingPerson = entityManager.find(Person.class, id);
            if (existingPerson == null) {
                return false;
            }
            existingPerson.setName(newDetails.getName());
            existingPerson.setSurname(newDetails.getSurname());
            existingPerson.setAge(newDetails.getAge());
            existingPerson.setAddress(newDetails.getAddress());
            existingPerson.setPhoneNumber(newDetails.getPhoneNumber());
            entityManager.getTransaction().commit();
            return true;
        }
    }

    public boolean deletePersonById(int id) {
        try (EntityManager entityManager =
entityManagerFactory.createEntityManager()) {
            entityManager.getTransaction().begin();
            Person person = entityManager.find(Person.class, id);
            if (person == null) {
                return false;
            }
            entityManager.remove(person);
            entityManager.getTransaction().commit();
            return true;
        }
    }
```

Дополнительно был создан PersonMapper, чтобы удобно делать преобразование DTO в Entity и обратно:

```
public class PersonMapper {

    public static PersonDto toDto(Person person) {
        if (person == null) {
            return null;
        }

        return PersonDto.builder()
                .name(person.getName())
                .surname(person.getSurname())
                .age(person.getAge())
                .address(person.getAddress())
```

```
                .phoneNumber(person.getPhoneNumber())
                .build();
    }

    public static Person toEntity(PersonDto personDto) {
        if (personDto == null) {
            return null;
        }

        return Person.builder()
                .name(personDto.getName())
                .surname(personDto.getSurname())
                .age(personDto.getAge())
                .address(personDto.getAddress())
                .phoneNumber(personDto.getPhoneNumber())
                .build();
    }
}
```

Ссылка на код

https://github.com/RedGry/TVS-LABS/tree/lab2

Вывод

В данной лабораторной работе успешно реализованы методы для создания, обновления и удаления записей в базе данных с использованием SOAP-сервиса. Клиентское приложение продемонстрировало корректное взаимодействие с новым функционалом.