Федеральное государственное автономное образовательное учреждение высшего образования

Университет ИТМО

Дисциплина: Технологии веб-сервисов

# Лабораторная работа 1

**Выполнили:**
Кривоносов Егор Дмитриевич

**Группа:** P4214

**Преподаватель:**
Сафронов Андрей Геннадьевич

2024 г.

Санкт-Петербург

# Оглавление

# Техническое задание

В данной работе требуется создать таблицу в базе данных, содержащую не менее пяти полей, а также реализовать возможность поиска по любым комбинациям полей с помощью SOAP-сервиса. Данные для поиска должны передаваться в метод сервиса в качестве аргументов.

Веб-сервис необходимо реализовать в виде standalone-приложения и J2EEприложения. При реализации в виде J2EE-приложения следует на стороне сервера приложений настроить источник данных и осуществлять его инъекцию в коде сервиса.

Для демонстрации работы разработанных сервисов следует также разработать и клиентское консольное приложение.

# Постановка задачи

Задача заключалась в разработке SOAP-сервиса для поиска информации в базе данных. Для реализации необходимо было создать три различных модуля: standalone приложение, Java EE приложение, и клиент, позволяющий взаимодействовать с обоими сервисами. Сервис должен обеспечивать выполнение запросов к базе данных по различным критериям, используя интерфейс SOAP, для предоставления данных в структурированном формате.

# Этапы выполнения

1. **Создание базы данных и настройка среды**
    - Разработка базы данных с таблицей persons и необходимыми данными.
    - Настройка сервера приложений WildFly для использования драйвера PostgreSQL и конфигурирование datasource.
    - Создание необходимых таблиц и выполнение инициализационных скриптов с помощью Docker или ручной настройки.

2. **Разработка standalone приложения**
    - Создание модуля standalone, представляющего собой SOAP-сервис, работающий без сервера приложений.
    - Написание основной логики взаимодействия с базой данных с использованием JPA.
    - Создание и настройка EntityManagerFactory для управления соединениями с базой данных PostgreSQL.
3. **Разработка Java EE приложения**
    - Создание Java EE версии SOAP-сервиса с использованием Jakarta и WildFly.
    - Конфигурация и деплой приложения в WildFly с использованием WAR-архива.
    - Подключение datasource через WildFly для взаимодействия с базой данных.
4. **Разработка клиента**
    - Создание Java-клиента, взаимодействующего с обоими сервисами через WSDL.
    - Клиент поддерживает команды для выполнения поиска, получения помощи и выхода из программы.
5. **Запуск и проверка сервисов**
    - Сборка и запуск standalone и Java EE версий сервиса с использованием Maven.
    - Тестирование клиента на корректность выполнения запросов и отображение полученных данных.

# Комментарии к архитектурным и функциональным аспектам реализации

- **SOAP-сервис** реализован с использованием аннотации @WebService, что позволяет удобно описывать операции, доступные через веб-сервис.
- **База данных** реализована с помощью PostgreSQL, а взаимодействие организовано с использованием JPA для удобного управления объектами в базе данных.
- **EntityManagerFactory** настраивается динамически с использованием параметров подключения, что обеспечивает гибкость при настройке сервиса для разных окружений.

- **Архитектурное разделение** на модули (standalone, j2ee, client) позволяет легко адаптировать и тестировать каждую часть приложения, улучшая модульность и удобство разработки.

Пример основного метода веб-сервиса:

```
@WebService(serviceName = "PersonService")
public class PersonWebService {
    private final PersonService personService;

    public PersonWebService(PersonService personService) {
        this.personService = personService;
    }

    @WebMethod
    public List<Person> searchPersons(@WebParam(name = "arg0") PersonListRequestDto
personListRequestDto) {
        if (personListRequestDto == null) {
            System.out.println("Received null PersonListRequestDto");
            return Collections.emptyList();
        }

        int limit = personListRequestDto.getLimit() != null ?
personListRequestDto.getLimit() : 10;
        int offset = personListRequestDto.getOffset() != null ?
personListRequestDto.getOffset() : 0;

        return personService.searchPersons(personListRequestDto.getQuery(), limit,
offset);
    }
}
```

Описание Сущности БД:

```
@Entity
@Table(name = "persons")
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Person {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @Column(name = "name")
    private String name;

    @Column(name = "surname")
    private String surname;

    @Column(name = "age")
    private int age;

    @Column(name = "address")
    private String address;

    @Column(name = "phone_number")
```

```
    private String phoneNumber;
}
```

Подключение к БД в standalone:

```
@Getter
public class EntityManagerFactoryProvider {
    private final EntityManagerFactory entityManagerFactory;

    public EntityManagerFactoryProvider(String url, String username, String
password) {
        entityManagerFactory = Persistence.createEntityManagerFactory(
                "postgres",
                Map.of(
                        "jakarta.persistence.jdbc.url", url,
                        "jakarta.persistence.jdbc.user", username,
                        "jakarta.persistence.jdbc.password", password
                )
        );
    }
}
```

# Ссылка на код

https://github.com/RedGry/TVS-LABS/tree/lab1

# Вывод

В ходе выполнения работы был разработан SOAP-сервис для поиска данных в базе данных PostgreSQL с использованием Java. Были созданы standalone и Java EE версии сервиса, что позволило оценить различия в подходах к разворачиванию и настройке сервисов. Клиентское приложение успешно продемонстрировало возможность взаимодействия с обоими сервисами, позволяя пользователю выполнять сложные запросы для получения необходимой информации. Разделение на модули упростило тестирование и поддержку кода, а также обеспечило гибкость и расширяемость системы.