**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 № 1**

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

Студент: Калинин Даниил Леонидович

Группа: P41141

Преподаватель: Дергачев Андрей Михайлович

\

Санкт-Петербург

2021

Задание:

В данной работе требуется создать таблицу в БД, содержащую не менее 5 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

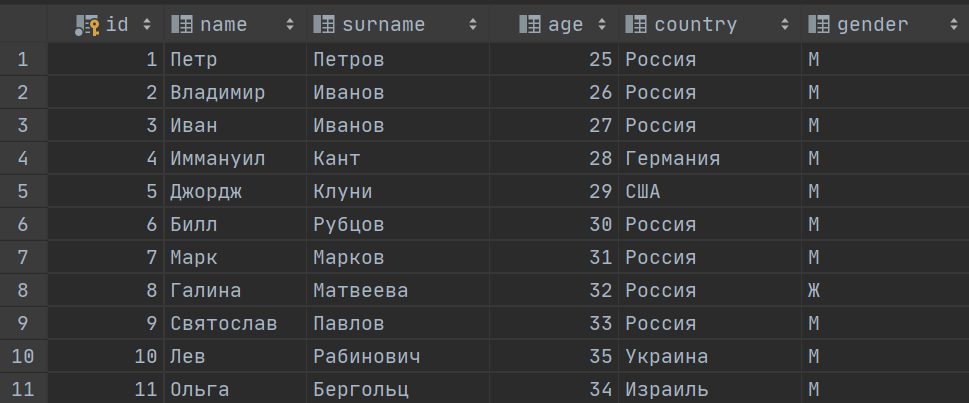
J2EE-приложения. При реализации в виде J2EE-приложения следует на стороне сервера приложений настроить источник данных, и осуществлять его инъекцию в коде сервиса.

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

Этапы выполнения:

Этап 1: Standalone - приложение

При выполнении лабораторной работы за основу был взят код из методического пособия с соответствующими доработками. Существующий код был переработан для подключение к базе данных MariaDB (Аналог MySQL). Была создана таблица для тестирования CRUD-функционала создаваемого приложения. Поля таблицы представлены на рисунке ниже



Поиск по таблице осуществляется при помощи API, реализованного в классе SQLQueryBuilder, принимающего на вход классы, реализующие интерфейс SQLConvertable, обязывающие класс возвращать HashMap, содержащий пары ключ – значение, являющиеся критериями поиска для формирования SQL-запроса, затем SQLQueryBuilder формирует текстовую строку запроса, отправляемого в базу данных через JDBC.

Класс SQLQueryBuilder

package web\_services;  
  
import java.util.HashMap;  
import java.util.Map;  
  
public class SQLQueryBuilder {  
  
 public String buildInsertQuery(SQLConvertable query){  
 HashMap<String, String> map = query.buildMap();  
  
 StringBuilder sqlQuery = new StringBuilder("INSERT INTO persons (");  
 int counter = 0;  
 for (Map.Entry<String, String> e: map.entrySet()){  
 counter++;  
 sqlQuery.append(e.getKey());  
 if (counter != map.size()) sqlQuery.append(", ");  
 }  
 counter = 0;  
 sqlQuery.append(") VALUES (");  
 for (Map.Entry<String, String> e: map.entrySet()){  
 counter++;  
 sqlQuery.append("\'"+e.getValue()+"\'");  
 if (counter != map.size()) sqlQuery.append(", ");  
 }  
 sqlQuery.append(")");  
 return sqlQuery.toString();  
 }  
  
 public String buildUpdateQuery(SQLConvertable query, Query update){  
 HashMap<String, String> map = query.buildMap();  
 HashMap<String, String> updateMap = update.buildMap();  
 StringBuilder sqlQuery = new StringBuilder("UPDATE persons set ");  
 int counter = 0;  
 for (Map.Entry<String, String> e: updateMap.entrySet()){  
 counter++;  
 sqlQuery.append(e.getKey() + " = " + "\'"+e.getValue()+"\'");  
 if (counter != updateMap.size()) sqlQuery.append(", ");  
 }  
 counter = 0;  
 sqlQuery.append(" WHERE ");  
 for (Map.Entry<String, String> e: map.entrySet()){  
 counter++;  
 sqlQuery.append(e.getKey() + " = " + "\'"+e.getValue()+"\'");  
 if (counter != map.size()) sqlQuery.append(" and ");  
 }  
 return sqlQuery.toString();  
 }  
  
 public String buildDeleteQuery(SQLConvertable query){  
 HashMap<String, String> map = query.buildMap();  
 StringBuilder sqlQuery = new StringBuilder("DELETE \* FROM persons WHERE ");  
 int counter = 0;  
 for (Map.Entry<String, String> e: map.entrySet()){  
 counter++;  
 sqlQuery.append(e.getKey() + " = " + "\'"+e.getValue()+"\'");  
 if (counter != map.size()) sqlQuery.append(" and ");  
 }  
 return sqlQuery.toString();  
 }  
  
 public String buildSelectQuery(SQLConvertable query){  
 HashMap<String, String> map = query.buildMap();  
 StringBuilder sqlQuery = new StringBuilder("select \* from persons where ");  
 int counter = 0;  
 for (Map.Entry<String, String> e: map.entrySet()){  
 counter++;  
 sqlQuery.append(e.getKey() + " = " + "\'"+e.getValue()+"\'");  
 if (counter != map.size()) sqlQuery.append(" and ");  
 }  
 return sqlQuery.toString();  
 }  
  
}

Интерфейс SQLConvertable

public interface SQLConvertable {  
 public HashMap<String, String> buildMap();  
}

Этап 2: Клиентское приложение

В клиенте были реализованы всего два класса:

**WebServiceClient**, осуществляющий подключение к развёрнутому J2EE или Standalone-приложению и отправляющий запросы через нужные методы.

**ConsoleInputReader**, осуществляющий считывание параметров запроса из консоли и формирующий на основе этих параметров класс Query, который маршализуется и передается на сервер.

Остальные классы были автоматически сгенерированы средой разработки по wsdl-описанию сервиса, предоставляемому сервером

Этап 3: J2EE-приложение

Доработка приложения для разворачивания на сервере приложений GlassFish не потребовала серьезных переработок. Подключение к базе данных было заменено на Dependency Injection вариант, управляемый через настройки GlassFish.

Выводы:

В ходе выполнения данной лабораторной работы был создан SOAP-сервис, позволяющий выполнять поиск по полям таблицы базы данных. Было реализовано два вариант развертывания сервиса – Standalone и при помощи сервера приложений GlassFish. Для взаимодействия с сервисом было создано клиентское консольное приложение, основная часть кода взаимодействия с сервисом которого была сгенерирована автоматически благодаря входящему в состав технологии SOAP формату описания вэб-сервисов WSDL.

Вопросы:

Ссылка на GitHub:

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