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

	id	name	surname	age	country	gender
1	1	Петр	Петров	25	Россия	М
2	2	Владимир	Иванов	26	Россия	М
3	3	Иван	Иванов	27	Россия	М
4	4	Иммануил	Кант	28	Германия	М
5	5	Джордж	Клуни	29	США	М
6	6	Билл	Рубцов	30	Россия	М
7	7	Марк	Марков	31	Россия	М
8	8	Галина	Матвеева	32	Россия	Ж
9	9	Святослав	Павлов	33	Россия	М
10	10	Лев	Рабинович	35	Украина	М
11	11	Ольга	Бергольц	34	Израиль	М

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:

1. **Что такое JAXB, как аннотации JAXB можно применить в контексте SOAP-сервисов?**

   JAXB-аннотации используются для задания правил сериализации и десериализации JAVA-объектов в формат XML и обратно. В контексте SOAP-сервисов JAXB-аннотации можно применять самым непосредственным образом, управляя сериализацией объектов, передаваемых по сети, для реализации удаленного вызова процедур.

2. **Предположим, что я разработал веб-сервис, но не смог предоставить WSDL разработчикам программного-клиента, а предоставил только XSD. Смогут ли они коммуницировать с моим веб-сервисом в случае, если им известна только XSD-схема?**

   XSD-схема описывает формат данных, участвующих во взаимодействии клиента и сервера SOAP-приложения, но не сами endpoint-ы, таким образом, имея только XSD, мы можем узнать, какие данные мы можем отправлять и получать при взаимодействии с приложением, но не сможем осуществлять взаимодействие, так как не знаем, куда именно отправлять данные.

3. **Предположим, при реализации SOAP веб-сервиса Вам необходимо передавать двоичные данные (аватары пользователей, архивы...). Какие стратегии Вы можете предложить для работы с binary attachments? Реализуйте одну из них.**

   Для работы с двоичными данными можно оперировать напрямую массивом байт в качестве принимаемого/возвращаемого аргумента SOAP-endpoint-ов, однако это не очень удобно и вызывает разрастание передаваемых между клиентом и сервисом XML-файлов, так как в таком случае данные передаются в виде base-64 строки, поэтому для работы с бинарным данными в SOAP лучше использовать MTOM - Message Transmission Optimization Mechanism, позволяющий более эффективно передавать данные за счет сжатия.

Реализация.

Для передачи изображения кота на стороне сервера был создан web-сервис

```java

@StreamingAttachment(parseEagerly=true, memoryThreshold=40000L)
@MTOM
@WebService(name="CatImage")
public class CatImage {

        @XmlMimeType("application/octet-stream")
        @WebMethod
        public DataHandler fileDownload()
        {
            return  new DataHandler(
                    new FileDataSource( name: "src/main/resources/cat.jpg"));
        }
}
```

который затем был опубликован параллельно созданному ранее в лабораторной сервису

```java

public class App {
    public static void main(String[] args) {
        String url = "http://localhost:8080/PersonService";
        String catUrl = "http://localhost:8080/CatImageService";
        System.setProperty("com.sun.xml.ws.fault.SOAPFaultBuilder.disableCaptureStackTrace",
                "false");
        Endpoint.publish(url, new PersonWebService());
        Endpoint.publish(catUrl, new CatImage());
    }
}
```

На клиентской стороне было осуществлено подключение к сервису с помощью сгенерированного на основе WSDL-описания кода.

```java
    public static void main(String[] args) {
        CatImageService catImageService = new CatImageService();
        MTOMFeature feature = new MTOMFeature();
        CatImage port = catImageService.getCatImagePort(feature);
        StreamingDataHandler streamDataHandler = (StreamingDataHandler)port.fileDownload();
        try{
            File file = new File( pathname: "./cat.jpg");
            streamDataHandler.moveTo(file);
            streamDataHandler.close();
        }
        catch(Exception e){
            e.printStackTrace();
        }
}
```

Итог – изображение успешно передается и сохраняется по указанному пути на клиенте

4. В чем заключается роль утилиты wsimport, и можно ли обойтись без нее?

Роль утилиты wsimport состоит в генерации классов для взаимодействия с web-сервисом на основе его WSDL-описания. Данная утилита генерирует набор классов для обращения к службам сервиса, классы исключений, генерируемых сервисом, объекты, участвующие в коммуникации с сервисом и средства асинхронного обращения к сервису. Технически, без данной утилиты можно обойтись, так как коммуникация происходит посредствам протокола HTTP, и мы можем организовать взаимодействие вручную, генерируя и получая xml-файлы в указанном в стандарте формате в случае, если, к примеру, необходимо организовать взаимодействие с SOAP-сервисом на языке, не имеющем средств генерации кода по WSDL-описанию. Однако такое взаимодействие будет крайне неудобным.

5. В чем заключается роль утилиты wsgen, и можно ли обойтись без нее?

Утилита wsgen генерирует WSDL и XSD-описание сервиса на основе анализа классов кода web-сервиса. Снова, технически, обойтись без данной утилиты можно, имея на руках исходный код сервиса, и организовав взаимодействие вручную, однако это будет тяжело и трудозатрато, и, в целом, будет неправильным использованием технологии

Ссылка на GitHub:

https://github.com/KalininDL/web_services_spring