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: Норин Евгений Рустамович

Группа: P41142

Преподаватель: Дергачев Андрей Михайлович

Санкт-Петербург

2020

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.

Выполнение работы:

В файле *Article.sql*, код которого представлен в листинге 1.1, присутствует описание таблицы, описывающей научную статье. Листинг 1.2 представляет собой процесс заполнение таблицы:

```
CREATE TABLE Article
                        (
                          article_id   bigserial primary key,
                          author_id    varchar NOT NULL,
                          h_index      BIGINT  NOT NULL,
                          article_name varchar NOT NULL,
                          article_desc varchar NOT NULL,
                          date_added   BIGINT NOT NULL
                        );
```

Листинг 1.1 Содержимое Article.sql

```
INSERT INTO public.article (article_id, author_id, h_index, article_name, article_desc, date_added) VALUES (1, 'author_id', 15, 'article_name', 'article_desc', 123);

INSERT INTO public.article (article_id, author_id, h_index, article_name, article_desc, date_added) VALUES (2, 'author_id_1', 15, 'article_name_1', 'article_desc_1', 124);

INSERT INTO public.article (article_id, author_id, h_index, article_name, article_desc, date_added) VALUES (3, 'author_id_2', 10, 'article_name_2', 'article_desc_2', 124);
```

Листинг 1.2 Заполнение Article.sql

Код сервиса в виде standalone-приложения представлен в листингах 1.3-

1.8. Класс AppStarter.java содержит main метод, и его основная цель – это запустить веб-сервис. ConnectionUtil.java используется для получения JDBC-соединений с базой данных. Article.java – POJO, который соответствует сущности, описанной в таблице Article.sql базы данных. ArticleWebService.java содержит операцию getArticles – этот метод умеет в поиск по любым комбинациям полей (отсутствие параметров в запросе означает полную выборку из таблицы). ArticlesDao.java содержит методы для выборки данных из базы данных. QueryBuilder.java отвечает за построение SQL запросов.

Листинг 1.3 Файл AppStarter.java

```
package ru.itmo.webservices.firstlab.standalone;

import javax.xml.ws.Endpoint;

public class AppStarter {
  public static void main(String[] args) {
    String url = "http://0.0.0.0:8080/ArticleService";
    Endpoint.publish(url, new ArticleWebService());
  }
}
```

Листинг 1.4 Файл ArticleWebService.java

```
package ru.itmo.webservices.firstlab.standalone;

import ru.itmo.webservices.firstlab.standalone.dao.ArticlesDao;
import ru.itmo.webservices.firstlab.standalone.pojo.Article;

import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;
import java.util.List;

@WebService(serviceName = "ArticlesService")
public class ArticleWebService {
  @WebMethod
  public List<Article> getArticles(@WebParam(name = "authorId") String authorId,
                   @WebParam(name = "hIndex") Long hIndex,
                   @WebParam(name = "articleName") String articleName,
                   @WebParam(name = "articleDesc") String articleDesc,
                   @WebParam(name = "dateAdded") Long dateAdded) {
    ArticlesDao dao = new ArticlesDao();
    return dao.getArticles(authorId, hIndex, articleName, articleDesc, dateAdded);
  }
}
```

Листинг 1.5 Файл Article.java

```
package ru.itmo.webservices.firstlab.standalone.pojo;

public class Article {
  private String authorId;
```

```
private Long hIndex;
private String articleName;
private String articleDesc;
private Long dateAdded;

public Article() {
}

public Article(String authorId, Long hIndex, String articleName, String articleDesc, Long dateAdded) {
   this.authorId = authorId;
   this.hIndex = hIndex;
   this.articleName = articleName;
   this.articleDesc = articleDesc;
   this.dateAdded = dateAdded;
}

public String getAuthorId() {
   return authorId;
}

public void setAuthorId(String authorId) {
   this.authorId = authorId;
}

public Long gethIndex() {
   return hIndex;
}

public void sethIndex(Long hIndex) {
   this.hIndex = hIndex;
}

public String getArticleName() {
   return articleName;
}

public void setArticleName(String articleName) {
   this.articleName = articleName;
}

public String getArticleDesc() {
   return articleDesc;
}

public void setArticleDesc(String articleDesc) {
   this.articleDesc = articleDesc;
}

public Long getDateAdded() {
   return dateAdded;
}

public void setDateAdded(Long dateAdded) {
   this.dateAdded = dateAdded;
}

@Override
public String toString() {
   return "Article{" +
         "authorId='" + authorId + '\'' +
         ", hIndex=" + hIndex +
```

```
            ", articleName='" + articleName + '\'' +
            ", articleDesc='" + articleDesc + '\'' +
            ", dateAdded=" + dateAdded +
            '}';
    }
}
```

Листинг 1.6 Файл QueryBuilder.java

```
package ru.itmo.webservices.firstlab.standalone.dao;

public class QueryBuilder {
    public static String build(String authorId, Long hIndex, String articleName, String articleDesc, Long
dateAdded) {
        StringBuilder builder = new StringBuilder(0);
        if (authorId != null) {
            builder.append(String.format("author_id='%s'", authorId));
        }
        if (hIndex != null) {
            if (builder.length() != 0) {
                builder.append(String.format("and h_index=%d", hIndex));
            } else {
                builder.append(String.format("h_index=%d", hIndex));
            }
        }
        if (articleName != null) {
            if (builder.length() != 0) {
                builder.append(String.format("and article_name='%s'", articleName));
            } else {
                builder.append(String.format("article_name='%s'", articleName));
            }
        }
        if (articleDesc != null) {
            if (builder.length() != 0) {
                builder.append(String.format("and article_desc='%s'", articleDesc));
            } else {
                builder.append(String.format("article_desc='%s'", articleDesc));
            }
        }
        if (dateAdded != null) {
            if (builder.length() != 0) {
                builder.append(String.format("and date_added=%d", dateAdded));
            } else {
                builder.append(String.format("date_added=%d", dateAdded));
            }
        }
        if (builder.length() != 0) {
            return "select * from article where " + builder.toString();
        } else {
            return "select * from article;";
        }
    }
}
```

Листинг 1.7 Файл ConnectionUtil.java

```
package ru.itmo.webservices.firstlab.standalone.dao;

import java.sql.Connection;
```

```
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;

public class ConnectionUtil {
   private static final String JDBC_URL = "jdbc:postgresql://192.168.99.100:32768/mydb";
   private static final String JDBC_USER = "******";
   private static final String JDBC_PASSWORD = "*****";

   static {
      try {
         Class.forName("org.postgresql.Driver");
      } catch (ClassNotFoundException ex) {
         Logger.getLogger(ArticlesDao.class.getName()).log(Level.SEVERE, null, ex);
      }
   }

   public static Connection getConnection() {
      Connection connection = null;
      try {
         connection = DriverManager.getConnection(JDBC_URL, JDBC_USER,
               JDBC_PASSWORD);
      } catch (SQLException ex) {
         Logger.getLogger(ConnectionUtil.class.getName()).log(Level.SEVERE, null, ex);
      }
      return connection;
   }
}
```

Листинг 1.8 Файл ArticlesDao.java

```
package ru.itmo.webservices.firstlab.standalone.dao;

import ru.itmo.webservices.firstlab.standalone.pojo.Article;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

public class ArticlesDao {
   public List<Article> getArticles(String authorId, Long hIndex, String articleName, String articleDesc, Long
dateAdded) {
      List<Article> articles = new ArrayList<>();
      try (Connection connection = ConnectionUtil.getConnection()) {
         Statement stmt = connection.createStatement();
         ResultSet rs = stmt.executeQuery(
               QueryBuilder.build(
                     authorId,
                     hIndex,
                     articleName,
                     articleDesc,
                     dateAdded
               )
         );
```

```
        while (rs.next()) {
          articles.add(
              new Article(
                  rs.getString("author_id"),
                  rs.getLong("h_index"),
                  rs.getString("article_name"),
                  rs.getString("article_desc"),
                  rs.getLong("date_added")
              )
          );
        }
      } catch (SQLException ex) {
        Logger.getLogger(ArticlesDao.class.getName()).log(Level.SEVERE, null, ex);
      }

      return articles;
    }
}
```

Код сервиса в виде J2EE-приложения представлен в листинге 1.9.

Классы QueryBuilder.java, Article.java, ArticlesDao.java аналогичны классам в standalone-приложении. ArticleWebService.java содержит одну операцию: getArticles, семантически аналогичную одноименному методу из standalone приложения. Также содержит инъекцию источника данных, настроенного на стороне сервера приложений glassfish.

Листинг 1.9 Файл ArticleWebService.java

```
import javax.annotation.Resource;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;
import javax.sql.DataSource;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

@WebService(serviceName = "ArticlesService")
public class ArticleWebService {
    @Resource(lookup = "jdbc/mydb")
    private DataSource dataSource;

    @WebMethod
    public List<Article> getArticles(@WebParam(name = "authorId") String authorId,
                                     @WebParam(name = "hIndex") Long hIndex,
                                     @WebParam(name = "articleName") String articleName,
                                     @WebParam(name = "articleDesc") String articleDesc,
                                     @WebParam(name = "dateAdded") Long dateAdded) {
        ArticlesDao dao = new ArticlesDao(getConnection());
        return dao.getArticles(authorId, hIndex, articleName, articleDesc, dateAdded);
```

```
    }

    private Connection getConnection() {
        Connection result = null;
        try {
            result = dataSource.getConnection();
        } catch (SQLException ex) {
            Logger.getLogger(ArticleWebService.class.getName()).log(Level.SEVERE, null, ex);
        }
        return result;
    }
}
```

WSDL сервиса представлен в листинге 1.10 и доступен по адресу http://localhost:8080/ArticleService?wsdl :

Листинг 1.10 WSDL сервиса статей

```
<definitions
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
    xmlns:wsp="http://www.w3.org/ns/ws-policy"
    xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy"
    xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="http://standalone.firstlab.webservices.itmo.ru/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://standalone.firstlab.webservices.itmo.ru/" name="ArticlesService">
    <types>
        <xsd:schema>
            <xsd:import namespace="http://standalone.firstlab.webservices.itmo.ru/"
schemaLocation="http://localhost:8080/ArticleService?xsd=1"/>
        </xsd:schema>
    </types>
    <message name="getArticles">
        <part name="parameters" element="tns:getArticles"/>
    </message>
    <message name="getArticlesResponse">
        <part name="parameters" element="tns:getArticlesResponse"/>
    </message>
    <portType name="ArticleWebService">
        <operation name="getArticles">
            <input
wsam:Action="http://standalone.firstlab.webservices.itmo.ru/ArticleWebService/getArticlesRequest" message="tns:getArticles"/>
            <output
wsam:Action="http://standalone.firstlab.webservices.itmo.ru/ArticleWebService/getArticlesResponse" message="tns:getArticlesResponse"/>
        </operation>
```

```
    </portType>
    <binding name="ArticleWebServicePortBinding" type="tns:ArticleWebService">
        <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
        <operation name="getArticles">
            <soap:operation soapAction=""/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
        </operation>
    </binding>
    <service name="ArticlesService">
        <port name="ArticleWebServicePort" binding="tns:ArticleWebServicePortBinding">
            <soap:address location="http://localhost:8080/ArticleService"/>
        </port>
    </service>
</definitions>
```

Для работы клиента была сгенерирована клиентская часть при помощи утилиты wsimport следующим образом –
/usr/lib/jvm/jdk1.8.0_211/bin/wsimport -Xnocompile http://localhost:8080/ArticleService?wsdl -d /mnt/c/Users/evgeniy/IdeaProjects/webservices/webservices/firstlab-client/src/main/java

Были сгенерированы следующие классы:
Article.java, ArticlesService.java, ArticleWebService.java, GetArticles.java, GetArticlesResponse.java, ObjectFactory.java, package-info.java
Для сгенерированного Article.java был переопределен toString для более красивого вывода.

В листинге 1.11 приведен код клиента, служащий для демонстрации работы сервиса:

Листинг 1.11 Код клиента

```
public class FirstLabClient {

    public static void main(String[] args) throws MalformedURLException {
        URL url = new URL("http://localhost:8080/ArticleService?wsdl");
        ArticlesService articlesService = new ArticlesService(url);
        ArticleWebService articlesWebService = articlesService.getArticleWebServicePort();


        System.out.println("Querying all existing Articles");
```

```
        List<Article> articles = articlesWebService.getArticles(null, null, null, null, null);
        for (Article picture : articles) {
            System.out.println(picture.toString());
        }

        System.out.println("Querying all Articles by specific fields");
        List<Article> articles2 = articlesWebService.getArticles(null, 15L, null, "article_desc_1", null);
        for (Article picture : articles2) {
            System.out.println(picture.toString());
        }

    }
}
```

Результат работы клиента представлен на Рисунке 1.1

Рисунок 1.1. Результат работы клиента

```
Querying all existing Articles
Article{articleDesc='article_desc', articleName='article_name', authorId='author_id', dateAdded=123, hIndex=15}
Article{articleDesc='article_desc_1', articleName='article_name_1', authorId='author_id_1', dateAdded=124, hIndex=15}
Article{articleDesc='article_desc_2', articleName='article_name_2', authorId='author_id_2', dateAdded=124, hIndex=10}
Querying all Articles by specific fields
Article{articleDesc='article_desc_1', articleName='article_name_1', authorId='author_id_1', dateAdded=124, hIndex=15}

Process finished with exit code 0
```

Такой вывод ожидаем – передача всех параметров со значениями null вынуждает сервис вернуть полную выборку таблицы Article.sql (см. Листинг 1.1)

Именно так было задумано в QueryBuilder.java (см. Листинг 1.6)

Вывод: в ходе выполнения работы была создана и заполнена таблица Article, а также реализована возможность поиска по любым комбинациям полей с помощью SOAP-сервиса в виде standalone-приложения и J2EE приложения. Для демонстрации работы разработанных сервисов было разработано клиентское консольное приложение.