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 № 4

ПО ДИСЦИПЛИНЕ «ТЕХНОЛОГИИ ВЕБ-СЕРВИСОВ»

Студент: Норин Евгений Рустамович

Группа: P41142

Преподаватель: Дергачев Андрей Михайлович

Санкт-Петербург

2020

Задание:

Необходимо выполнить задание из первой лабораторной работы, но с использованием REST-сервиса. Таблицу базы данных, а также код для ее работы с ней можно оставить без изменений.
Веб-сервис необходимо реализовать в виде standalone-приложения и J2EE-приложения. 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, 'authordId', 1, 'articleName', 'articleDesc', 1);

INSERT INTO public.article (article_id, author_id, h_index, article_name,
article_desc, date_added)
VALUES (2, 'authordId2', 2, 'articleName2', 'articleDesc2', 2);

INSERT INTO public.article (article_id, author_id, h_index, article_name,
article_desc, date_added)
VALUES (3, 'authordId3', 3, 'articleName3', 'articleDesc3', 3);
```

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. ArticleResource.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.fourthlab.standalone;

import com.sun.jersey.api.container.grizzly2.GrizzlyServerFactory;
import com.sun.jersey.api.core.ClassNamesResourceConfig;
import com.sun.jersey.api.core.ResourceConfig;
import org.glassfish.grizzly.http.server.HttpServer;

import java.io.IOException;
import java.net.URI;

public class AppStarter {
    private static final URI BASE_URI = URI.create("http://localhost:8080/");

    public static void main(String[] args) {
        HttpServer server = null;
        try {
            ResourceConfig resourceConfig = new
ClassNamesResourceConfig(ArticleResource.class);
            server = GrizzlyServerFactory.createHttpServer(BASE_URI, resourceConfig);
            server.start();
            System.in.read();
            stopServer(server);
        } catch (IOException e) {
            e.printStackTrace();
            stopServer(server);
        }
    }

    private static void stopServer(HttpServer server) {
        if (server != null)
            server.stop();
    }
}
```

Листинг 1.4 Файл ArticleResource.java

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

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

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.QueryParam;
import javax.ws.rs.core.MediaType;
import java.util.List;

@Path("/articles")
@Produces({MediaType.APPLICATION_JSON})
public class ArticleResource {

    @GET
    @Path("/find")
    public List<Article> getArticles(@QueryParam("authorId") String authorId,
                                     @QueryParam("hIndex") Long hIndex,
```

```
                                        @QueryParam("articleName") String articleName,
                                        @QueryParam("articleDesc") String articleDesc,
                                        @QueryParam("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.fourthlab.standalone.pojo;

import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement
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.fourthlab.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.fourthlab.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://localhost:32768/mydb";
    private static final String JDBC_USER = "myuser";
    private static final String JDBC_PASSWORD = "mypass";

    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.fourthlab.standalone.dao;

import ru.itmo.webservices.fourthlab.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. ArticleResource.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 ArticleResource.java

```
package ru.itmo.webservices.fourthlab.j2ee;

import javax.annotation.Resource;
import javax.enterprise.context.RequestScoped;
import javax.jws.WebParam;
import javax.sql.DataSource;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.QueryParam;
import javax.ws.rs.core.MediaType;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

@RequestScoped
@Path("/articles")
@Produces({MediaType.APPLICATION_JSON})
public class ArticleResource {
    @Resource(lookup = "jdbc/mydb")
    private DataSource dataSource;

    @GET
    @Path("/find")
    public List<Article> getArticles(@QueryParam("authorId") String authorId,
                                     @QueryParam("hIndex") Long hIndex,
                                     @QueryParam("articleName") String articleName,
```

```
                                        @QueryParam("articleDesc") String articleDesc,
                                        @QueryParam("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(ArticleResource.class.getName()).log(Level.SEVERE, null,
ex);
        }
        return result;
    }
}
```

В листинге 1.10 приведен код клиента, служащий для демонстрации работы сервиса:

Листинг 1.10 Код клиента

```
package ru.itmo.webservices.fourthlab.client;

import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Set;

import com.sun.jersey.api.client.Client;
import com.sun.jersey.api.client.ClientResponse;
import com.sun.jersey.api.client.GenericType;
import com.sun.jersey.api.client.WebResource;

import javax.ws.rs.core.MediaType;

public class FourthLabClient {

    public static void main(String[] args) {

        Client client = Client.create();
        System.out.println("Query: /articles");

        String  url = "http://localhost:8080/articles/find";

        System.out.println("Querying all existing Articles");
        display(findArticles(client, url, ""));
        System.out.println();

        System.out.println("Querying all existing Articles with authorId=authorId");
        display(findArticles(client, url, "authorId=authorId"));
        System.out.println();

        System.out.println("Querying all existing Articles with hIndex=3");
        display(findArticles(client, url, "hIndex=3"));
        System.out.println();

        System.out.println("Querying all existing Articles with
articleName=articleName2");
        display(findArticles(client, url, "articleName=articleName2"));
    }
```

```
    private static List<Article> findArticles(Client client, String url, String query)
{
        WebResource webResource = client.resource(url);
        if (!query.isEmpty()) {
            Map<String, String> map = getQueryMap(query);

            Set<String> keys = map.keySet();
            for (String key : keys) {
                webResource = webResource.queryParam(key, map.get(key));
            }
        }

        ClientResponse response =
webResource.accept(MediaType.APPLICATION_JSON).get(ClientResponse.class);
        if (response.getStatus() != ClientResponse.Status.OK.getStatusCode()) {
            throw new IllegalStateException("Request failed");
        }

        GenericType<List<Article>> type = new GenericType<List<Article>>() {
        };

        return response.getEntity(type);
    }

    private static Map<String, String> getQueryMap(String query) {
        String[] params = query.split("&");
        Map<String, String> map = new HashMap<String, String>();
        for (String param : params) {
            String name = param.split("=")[0];
            String value = param.split("=")[1];
            map.put(name, value);
        }
        return map;
    }

    private static void display(List<Article> articles) {
        for (Article article : articles) {
            System.out.println(article);
        }
    }
}
```

Результат работы клиента представлен на Рисунке 1.1

Рисунок 1.1. Результат работы клиента

```
Querying all existing Articles
Article{authorId='authorId', hIndex=1, articleName='articleName', articleDesc='articleDesc', dateAdded=1}
Article{authorId='authorId2', hIndex=2, articleName='articleName2', articleDesc='articleDesc2', dateAdded=2}
Article{authorId='authorId3', hIndex=3, articleName='articleName3', articleDesc='articleDesc3', dateAdded=3}

Querying all existing Articles with authorId=authorId
Article{authorId='authorId', hIndex=1, articleName='articleName', articleDesc='articleDesc', dateAdded=1}

Querying all existing Articles with hIndex=3
Article{authorId='authorId3', hIndex=3, articleName='articleName3', articleDesc='articleDesc3', dateAdded=3}

Querying all existing Articles with articleName=articleName2
Article{authorId='authorId2', hIndex=2, articleName='articleName2', articleDesc='articleDesc2', dateAdded=2}

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 реализована возможность поиска по любым комбинациям полей с помощью REST-сервиса в виде standalone-приложения и J2EE-приложения. Для демонстрации работы разработанных сервисов было разработано клиентское консольное приложение.