Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

**Отчёт о лабораторной работе №8**

**Дисциплина**: Базы данных

**Тема**: Клиентское приложение

Выполнила студентка гр. 43501/1 В.В. Ракова

(подпись)

Руководитель А.В. Мяснов

(подпись)

“ ” 2015 г.

Санкт-Петербург

2015

1. **Цель работы**

Ознакомиться с разработкой клиентских приложений.

1. **Программа работы**

Необходимо создать консольное приложение выполняющее следующие функции:

* соединение с БД, выполнение фиксированного SQL-запроса и получение результатов запроса
* добавление данных в одну из таблиц БД
* выполнение хранимой процедуры
* реализовать импорт данных из JSON файлов
* реализовать экспорт данных в XML

1. **Выполнение работы**

**Листинг:**

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package litpub;  import java.io.FileNotFoundException;  import java.sql.\*;  import java.util.Scanner;  import java.util.Vector;  import org.json.simple.JSONArray;  import org.json.simple.JSONObject;  import org.json.simple.parser.JSONParser;  import org.json.simple.parser.ParseException;  import org.w3c.dom.Document;  import de.jeckle.RS2DOM.RS2DOM;  import java.io.FileOutputStream;  import javax.xml.transform.Transformer;  import javax.xml.transform.TransformerFactory;  import javax.xml.transform.dom.DOMSource;  import javax.xml.transform.stream.StreamResult;  import litpub.FileW;  /\*\*  \*  \* @author  \*/    public class Litpub {    public void err1()  {  System.err.println("Ошибка 1. Недопустимый формат ввода.");  }  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) throws ClassNotFoundException, SQLException, InstantiationException, IllegalAccessException, FileNotFoundException, ParseException {  // TODO code application logic here  Connection connect = null;  java.sql.Statement rqst = null; // для выполнения SQL запросов  Scanner sc = new Scanner(System.in); // класс для работы с консолью  int table\_number = 0; // номер введенной таблицы  Vector<String> vec\_tab = new Vector<String>();  String temp2 = null; // временные строковые переменные  String [] temp3 = null;  StringBuilder sb = new StringBuilder(); // для построения строки  Litpub E = new Litpub(); // для вызова ошибки ввода  int menu\_select = 0; // переменная = выбранный пункт меню  int argz; // входной аргумент для операции INSERT  int argz3; // входной аргумент для операции INSERT  int argz4; // входной аргумент для операции INSERT  int cnt\_col = 0; //для вывода содержимого таблицы  ResultSet res; // Класс для хранения результатов SQL запроса  Document doc = null; //Класс для хранения XML  // Иницализация драйвера  Class.forName("org.firebirdsql.jdbc.FBDriver").newInstance();  //Указание пути к БД  String strPath = "jdbc:firebirdsql://localhost/D:/Litpub/One.fdb";  Class.forName("org.firebirdsql.jdbc.FBDriver").newInstance();  //Подключение к БД  connect = DriverManager.getConnection(strPath, "SYSDBA", "masterkey");  if (connect == null) {  System.err.println("Подключение к БД не выполнено.");}  //Создание класса для выполнения SQL запросов  rqst = connect.createStatement();  System.out.println("Подключение к БД выполнено.");  // Получение списка таблиц БД  DatabaseMetaData metaData = connect.getMetaData();  ResultSet temp=metaData.getTables(temp2, temp2, temp2, temp3);  while(temp.next())  {  temp2=temp.getString(3);  if(!temp2.contains("$"))  vec\_tab.add(temp2);  }  // Меню  while (menu\_select != 6) {  // вывод меню  System.out.println(".....................................");  System.out.println("< Literary Publishing 'RUSBOOK' >");  System.out.println(".....................................");  System.out.println("Возможные операции:");  System.out.println("1.Вывести спискок таблиц");  System.out.println("2.Добавить данные в таблицу REDACTOR");  System.out.println("3.Выполнить хранимую процедуру DEL\_GENRE");  System.out.println("4.Импортировать из JSON данных в REDACTOR");  System.out.println("5.Экспортировать содержимое таблицы в XML");  System.out.println("6.Выйти");  // считывание номера пункта меню  System.out.println("Выберите операцию:");  try{  menu\_select = Integer.parseInt(sc.nextLine());  }catch(NumberFormatException e){  E.err1();  continue;  }  if (menu\_select > 6)  System.err.println("Ошибка 2.НЕсуществующая операция.Выбирите из представленных.");    if (menu\_select == 1)  {  System.out.println("Таблицы базы данных:");  for(int i=1;i<=vec\_tab.size();i++)  {  System.out.printf("%d. %s\n",i,vec\_tab.elementAt(i-1));  }  System.out.println("Введите номер таблицы для отображения ее содержимого или "  + "\n0 для возврата в основное меню:");  try{  table\_number=Integer.parseInt(sc.nextLine());  }catch(NumberFormatException e){  E.err1();  continue;  }  if((table\_number > vec\_tab.size()) || (table\_number < 0)){  System.err.println("Ошибка 2. НЕсуществующая таблица.Выбирите из представленных.");  continue;  }  if(table\_number == 0){  continue;  }  System.out.println();  //Выполнение SQL запроса  res = rqst.executeQuery("SELECT \* from "+ vec\_tab.elementAt(table\_number-1));  // Вывод результата  cnt\_col = res.getMetaData().getColumnCount();  // Вывод содержимого таблицы  // Сначала имена столбцов:  for(int i = 1; i < cnt\_col + 1; i++){  System.out.print(res.getMetaData().getColumnName(i)+  " | ");  }  // Затем сами записи в таблице:  while(res.next())  {  System.out.println();  for (int i = 1;i < cnt\_col + 1;i++)  {  Object obj = res.getObject(i);  if (obj!=null)  {  System.out.print(obj+" \t ");  }  }  }  System.out.println();      continue;  }    if (menu\_select == 2 )  {  if(connect == null) {  System.err.println("Нет соединение с БД.");  continue;  }  System.out.println("Введите ID редактора :");  try{  argz=Integer.parseInt(sc.nextLine());  }catch(NumberFormatException e){  E.err1();  continue;  }  if (argz <= 0)  {  E.err1();  continue;  }    System.out.println("Введите ФИО редактора:");  String argz2 = sc.nextLine();  if (argz2.length()>50 || argz2.isEmpty())  {  E.err1();  continue;  }    System.out.println("Введите опыт работы:");  try{  argz3=Integer.parseInt(sc.nextLine());  }catch(NumberFormatException e){  E.err1();  continue;  }  if (argz3<0)  {  E.err1();  continue;  }    try{  rqst.executeUpdate("insert into REDACTOR values ('"+argz+"','"+argz2+"','"+argz3+"');");  System.out.println("Информация о редакторе добавлена в таблицу.");  }catch (SQLException se){  System.out.println(se.getMessage());  }    continue;  }    if (menu\_select == 3)  {  if(connect == null) {  System.err.println("Нет соединения с БД.");  continue;  }  System.out.println("Хранимая процедура DEL\_GENRE удаляет неиспользуемые жанры \n");  PreparedStatement pstmt = connect.prepareStatement("{call DEL\_GENRE}");  pstmt.execute();  System.out.println("\nХранимая процедура DEL\_GENRE выполнена.");  pstmt.close();  continue;    }  if (menu\_select == 4)  {    JSONParser parser = new JSONParser(); //создание объекта для парсинга  String textjson = FileW.read("D://JSON/redactor.json");  Object obj = parser.parse(textjson);  JSONObject jsonObj = (JSONObject) obj;  JSONArray jo = (JSONArray) jsonObj.get("Redactor");    //Добавление данных в таблицу REDACTOR  for (int i = 0; i<jo.size();i++){  JSONObject element = (JSONObject) jo.get(i);  try{  rqst.executeUpdate("insert into REDACTOR values ('"+element.get("ID\_REDACTOR")+"','"+element.get("REDACTOR\_NAME")+"','"+element.get("EXPERIENCE\_RED")+"');");  }catch (SQLException se){  System.out.println(se.getMessage());  }  }    System.out.println("\nИмпорт данных из JSON файлов в таблицy REDACTOR выполнен.");    continue;      }  if (menu\_select ==5)  {  System.out.println("Список таблиц:");  for(int i=1;i<=vec\_tab.size();i++)  {  System.out.printf("%d. %s\n",i,vec\_tab.elementAt(i-1));  }    System.out.println("Введите номер таблицы для экспорта в XML:");  try{  table\_number=Integer.parseInt(sc.nextLine());  }catch(NumberFormatException e){  E.err1();  continue;  }  if((table\_number > vec\_tab.size()) || (table\_number < 0)){  System.err.println("Ошибка 2. НЕсуществующая таблица.Выбирите из представленных.");  continue;  }  if(table\_number == 0){  continue;  }  System.out.println();  //Выполнение SQL запроса  res = rqst.executeQuery("SELECT \* from "+ vec\_tab.elementAt(table\_number-1));    Document xsd = RS2DOM.ResultSet2XSDDOM(res);  Document d = RS2DOM.ResultSet2DOM(res);  try {  Transformer myTransformer =  (TransformerFactory.newInstance()).newTransformer();  System.out.println(  "Схема, описывающая XML, экспортирована в файл Description.xml");  myTransformer.transform(  new DOMSource(xsd),  new StreamResult(new FileOutputStream("D://Description.xml")));  System.out.println(  "\n\nСодержимое таблицы экспортировано в XML файл Data.xml");  myTransformer.transform(  new DOMSource(d),  new StreamResult(new FileOutputStream("D://Data.xml")));  } catch (Exception e) {  e.printStackTrace();  }  continue;  }  if (menu\_select == 6)  {  System.out.println("Работа с БД завершена. До свидания!!");  continue;  }  }  System.exit(0);  }        } |

**FileW.java**

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package litpub;  import java.io.BufferedReader;  import java.io.File;  import java.io.FileNotFoundException;  import java.io.FileReader;  import java.io.IOException;  /\*\*  \*  \* @author  \*/  public class FileW {  public static String read(String fileName) throws FileNotFoundException {  //Этот спец. объект для построения строки  //Определяем файл  File file = new File(fileName);    StringBuilder sb = new StringBuilder();    exists(fileName);    try {  //Объект для чтения файла в буфер  BufferedReader in = new BufferedReader(new FileReader( file.getAbsoluteFile()));  try {  //В цикле построчно считываем файл  String s;  while ((s = in.readLine()) != null) {  sb.append(s);  sb.append("\n");  }  } finally {  //Также не забываем закрыть файл  in.close();  }  } catch(IOException e) {  throw new RuntimeException(e);  }    //Возвращаем полученный текст с файла  return sb.toString();  }  private static void exists(String fileName) throws FileNotFoundException {  File file = new File(fileName);  if (!file.exists()){  throw new FileNotFoundException(file.getName());  }  }  private static String fileName;  } |

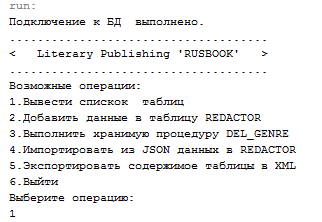
**Redactor.json**

|  |
| --- |
| {  "Redactor" :  [  {  "ID\_REDACTOR" : 10,  "REDACTOR\_NAME" : "Gert D.F.",  "EXPERIENCE\_RED" : 7    },  {  "ID\_REDACTOR" : 11,  "REDACTOR\_NAME" : "Gert D.D.",  "EXPERIENCE\_RED" : 5  }  ]} |

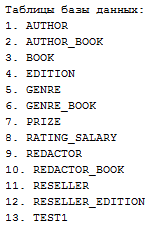
Для импорта данных из файлов JSON использовалась библиотека json-simple. Для экспорта данных в XML использовался класс RS2DOM.

**Примеры работы программы:**

При запуске программы выводим меню:

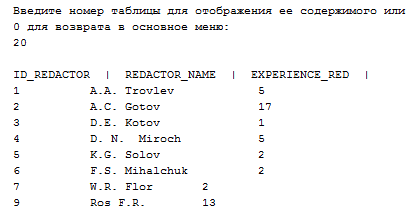
****

При выборе первого пункта меню: Вывести список всех таблиц.

****

При выборе второго пункта меню:

Таблица до добавления данных:



Добавляем данные:

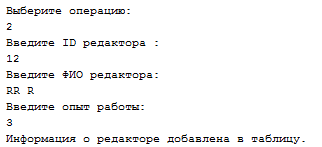
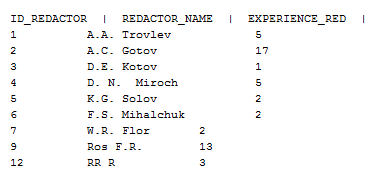
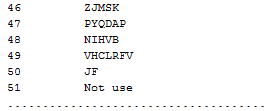


Таблица после добавления данных:

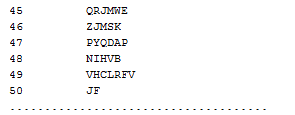


Выполнение процедуры DEL\_GENRE:

Список жанров до выполнения процедуры:



Список жанров после выполнения процедуры:



1. **Вывод**

В данной лабораторной работе было разработано клиентское приложение, для работы с БД.

JDBC (Java DataBase Connectivity) – это платформенно-независимый промышленный стандарт взаимодействия Java-приложений с различными [СУБД](https://ru.wikipedia.org/wiki/%D0%A1%D0%A3%D0%91%D0%94), реализованный в виде пакета java.sql, входящего в состав [Java SE](https://ru.wikipedia.org/wiki/Java_SE" \o "Java SE).

Преимуществами JDBC являются:

* 1. Сочетание JAVA API и JDBC API делает создание приложений лёгким и эффективным.
  2. Код приложения подвержен наименьшим изменениям в случае, если происходит смена базы данных.
  3. Лёгкость подсоединения к базе через легко описываемый URL.
  4. JDBC API полностью предоставляет доступ к метаданным, что позволяет писать сложные приложения
  5. Нет необходимости установки специального программного обеспечения, как на стороне клиента, так и на стороне сервера.
  6. Драйвера JDBC могут загружаться динамически.