Санкт-Петербургский Государственный Политехнический Университет

Институт Информационных Технологий и Управления

Кафедра Компьютерных Систем и Программных Технологий

**ОТЧЁТ О ЛАБОРАТОРНОЙ РАБОТЕ №8**

Дисциплина: **Базы данных**

Тема: **Клиентское приложение**

Н.В. Тышковец

А.В. Мяснов

Выполнил студент гр. № 43501/1

Преподаватель

Санкт-Петербург

2015

## Цели работы

## Ознакомиться с разработкой клиентских приложений.

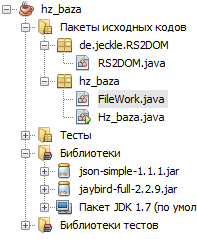
## Программа работы

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

**Ход работы**

Дерево проекта:



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

Листинг Tourfirm.java:

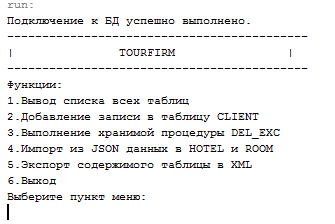
|  |
| --- |
| package hz\_baza;  import java.io.FileNotFoundException;  import java.sql.\*;  import java.util.Scanner;  import java.util.Vector;  import org.json.simple.JSONArray;  import org.json.simple.JSONObject;  import org.json.simple.parser.JSONParser;  import org.json.simple.parser.ParseException;  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;  public class Tourfirm {  /\*\*  \*  \* @param args  \*/  public static void main(String[] args) throws ClassNotFoundException,  SQLException, InstantiationException, IllegalAccessException,  FileNotFoundException, ParseException  {  Connection connect = null;  java.sql.Statement rqst = null; // объект 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(); //объект для построения строки  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; //переменная 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;  strPath = "jdbc:firebirdsql://localhost/C:/tourfirm3/hz.fdb";  Class.forName("org.firebirdsql.jdbc.FBDriver").newInstance();  //Подключение к БД  connect = DriverManager.getConnection(strPath, "SYSDBA", "masterkey");  if (connect == null) {  System.err.println("Невозможно подключиться к БД.");}  //Создание класса для выполнения SQL запросов  rqst = connect.createStatement();  System.out.println("Подключение к БД успешно 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("| TOURFIRM |");  System.out.println("-------------------------------------------");  System.out.println("Функции:");  System.out.println("1.Вывод списка всех таблиц");  System.out.println("2.Добавление записи в таблицу CLIENT");  System.out.println("3.Выполнение хранимой процедуры DEL\_EXC");  System.out.println("4.Импорт из JSON данных в HOTEL и ROOM");  System.out.println("5.Экспорт содержимого таблицы в XML");  System.out.println("6.Выход");  // считывание номера пункта меню  System.out.println("Выберите пункт меню:");  try{  menu\_select = Integer.parseInt(sc.nextLine());  }catch(NumberFormatException e){  System.err.println("Ошибка! Вводите только цифры.");  continue;  }  if (menu\_select > 6)  System.err.println("Ошибка! Пункт меню с таким номером отсутствует.");  if (menu\_select == 1)  {  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 отображения ее содержимого или "  + "\n0 для возврата в основное меню:");  try{  table\_number=Integer.parseInt(sc.nextLine());  }catch(NumberFormatException e){  System.err.println("Ошибка! Номер должен быть числом!");  continue;  }  if((table\_number > vec\_tab.size()) || (table\_number < 0)){  System.err.println("Ошибка! Таблица с таким номером отсутсвует.");  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("Соединение с БД не установлено.");  continue;  }  // ВВОД АРГУМЕНТОВ ДЛЯ ОПЕРАЦИИ INSERT В ТАБЛИЦУ ТИПОВ БИЛЕТОВ  System.out.println("Введите ID клиента:");  try{  argz=Integer.parseInt(sc.nextLine());  }catch(NumberFormatException e){  System.err.println("Ошибка! ID не является числом или превышает 9 символов.");  continue;  }  if (argz <= 0)  {  System.err.println("Ошибка! ID не может быть отрицательным или равным нулю.");  continue;  }    System.out.println("Введите имя клиента:");  String argz2 = sc.nextLine();  if (argz2.length()>32 || argz2.isEmpty())  {  System.err.println("Ошибка! Название типа не может быть пустым или больше 32 символов.");  continue;  }    try{  rqst.executeUpdate("insert into CLIENT values ('"+argz+"','"+argz2+"',null);");  System.out.println("Запись добавлена в таблицу.");  }catch (SQLException se){  System.out.println(se.getMessage());  }    continue;  }  if (menu\_select == 3)  {  if(connect == null) {  System.err.println("Соединение с БД не установлено.");  continue;  }  System.out.println("Хранимая процедура DEL\_EXC удаляет неиспользуемые экскурсиии \n");  PreparedStatement pstmt = connect.prepareStatement("{call DEL\_EXC}");  pstmt.execute();  System.out.println("\nХранимая процедура DEL\_TOUR\_MIN выполнена.");  pstmt.close();  continue;    }    if (menu\_select == 4)  {  JSONParser parser = new JSONParser(); //создание объекта для парсинга  String textjson = FileWork.read("D://Николя/БД/Консольное/JSON/hotel.json");  Object obj = parser.parse(textjson);  JSONObject jsonObj = (JSONObject) obj;  JSONArray jo = (JSONArray) jsonObj.get("HOTEL");    //Добавление данных в таблицу Hotel  for (int i = 0; i<jo.size();i++){  JSONObject element = (JSONObject) jo.get(i);  try{  rqst.executeUpdate("insert into HOTEL values ('"+element.get("ID")+"','"+element.get("HOTELNAME")  +"','"+element.get("COUNTRY\_ID")+"','"+element.get("STARS")+"','"+element.get("ROOM\_ID")+"');");  }catch (SQLException se){  System.out.println(se.getMessage());  }  }  //Добавление данных в таблицу Room  textjson = FileWork.read("D://Николя/БД/Консольное/JSON/room.json");  obj = parser.parse(textjson);  jsonObj = (JSONObject) obj;  jo = (JSONArray) jsonObj.get("ROOM");    for (int i=0; i<jo.size();i++){  JSONObject element = (JSONObject) jo.get(i);  try{  rqst.executeUpdate("insert into ROOM values"  + "('"+element.get("ID")+"','"+element.get("HOTEL\_ID")  +"',"+element.get("ROOM\_NUMBER")+");");  }catch (SQLException se){  System.out.println(se.getMessage());  }  }    System.out.println("\nИмпорт данных из JSON файлов в таблицы HOTEL и ROOM 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){  System.err.println("Ошибка! Номер должен быть числом!");  continue;  }  if((table\_number > vec\_tab.size()) || (table\_number < 0)){  System.err.println("Ошибка! Таблица с таким номером отсутсвует.");  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://Николя/БД/Консольное/XML/Description.xml")));  System.out.println(  "\n\nСодержимое таблицы экспортировано в XML файл Data.xml");  myTransformer.transform(  new DOMSource(d),  new StreamResult(new FileOutputStream("D://Николя/БД/Консольное/XML/Data.xml")));  } catch (Exception e) {  e.printStackTrace();  }  continue;  }    if (menu\_select == 6)  {  System.out.println("Goodbye");  continue;  }  }  System.exit(0);  }  } |

Листинг FileWork.java

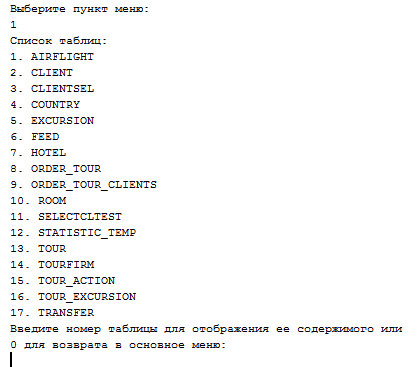
|  |
| --- |
| package hz\_baza;  import java.io.BufferedReader;  import java.io.File;  import java.io.FileNotFoundException;  import java.io.FileReader;  import java.io.IOException;  public class FileWork {  public static String read(String fileName) throws FileNotFoundException {  //Этот спец. объект 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;  } |

*Примеры работы программы:*

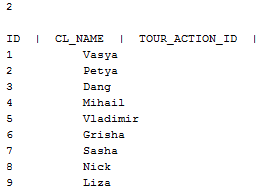
1) Подключение к БД



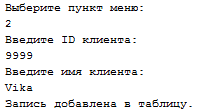
2) Выполнение пункта «Вывод списка всех таблиц».



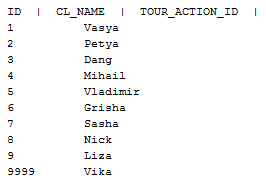
3) Отображение содержимого таблицы №2 «CLIENT».



4) Выполнение пункта «Добавление данных в таблицу CLIENT



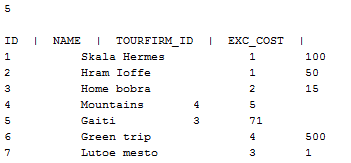
5)Проверка содержимого таблицы после добавления строки.



Как видно из двух предыдущих скриншотах, в таблицу CLIENT добавлена новая запись.

4) Выполнение хранимой процедуры DEL\_EXC (Удаление неиспользуемых экскурсий).

Перед выполнением запроса выведем все имеющиеся экскурсии.



Выполнение запроса:

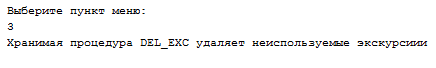
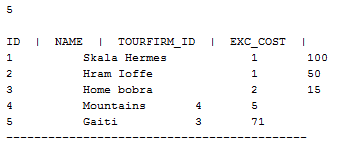


Таблица EXCURSION:



Как видно из скриншотов, из таблицы EXCURSION были удалены две неиспользуемые экскурсии.

5) Четвёртое действие – импорт данных из JSON.

Файл данных hotel.json:

|  |
| --- |
| {  "HOTEL":[  {  "ID" : 100,  "HOTELNAME" : "Rich",  "COUNTRY\_ID" : 2,  "STARS": 5,  "ROOM\_ID" : 2  },  {  "ID" : 101,  "HOTELNAME" : "Dzen",  "COUNTRY\_ID" : 3,  "STARS": 4,  "ROOM\_ID" : 5  }]  } |

Файл данных room.json:

|  |
| --- |
| {  "ROOM":[  {  "ID" : 200,  "HOTEL\_ID" : 7,  "ROOM\_NUMBER" : 34  },  {  "ID" : 201,  "HOTEL\_ID" : 3,  "ROOM\_NUMBER" : 67  },  {  "ID" : 202,  "HOTEL\_ID" : 5,  "ROOM\_NUMBER" : 2  }]  } |

Результат выполнения запроса:

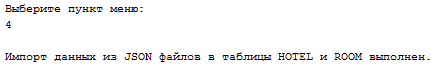


Таблица HOTEL:

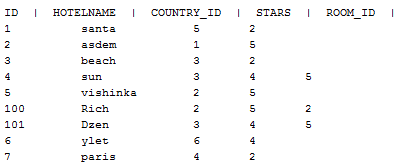
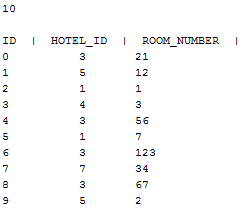


Таблица ROOM:

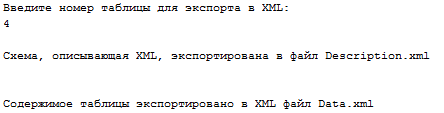


Как видно из скриншотов, данные из файлов JSON успешно импортированы в таблицу.

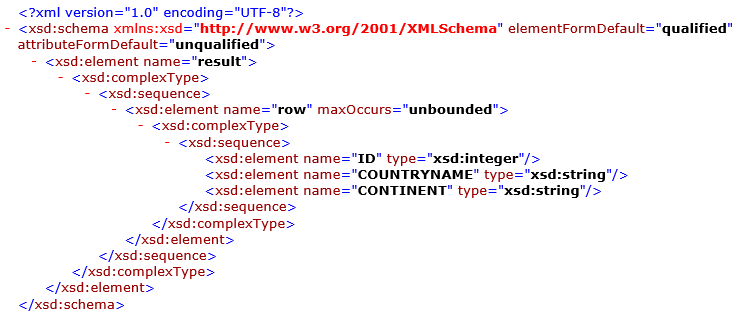
В таблицу ROOM добавлены записи с ID не соответствующими room.json из-за срабатывания триггера, задающего ID новой записи как инкремент предыдущего максимального ID.

6) Пятое действие – экспорт таблицы в XML:

Экспорт таблицы COUNTRY



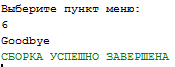
Содержимое Description.xml:



Содержимое Data.xml:



7) Выполнение пункта «Выход»



**Вывод**

В ходе выполнения данной лабораторной работы мы ознакомились с принципами создания простейшего консольного приложения для работы с базой данных, выполняющего определённые функции.

Выполнение данного задания как завершающего в данном цикле лабораторных работ логично, т.к. оно объединяет в себе две изученных платформы для программирования: пройденный ранее язык Java и язык SQL.

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). JDBC позволяет получать доступ к БД, а также выполнять к ней SQL запросы.

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.