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

Выполнил студент гр. 43501/1 Приймак А.Г.

(подпись)

Руководитель Мяснов А. В.

(подпись)

“ ” 2016 г.

Санкт-Петербург

2016

1. **Цель работы.**

Ознакомиться с разработкой клиентских приложений.

1. **Программа работы**

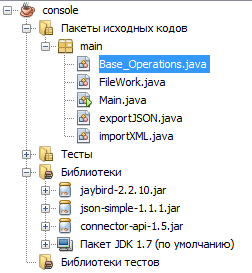
Необходимо создать консольное приложение выполняющее следующие функции:

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

1. **Ход работы.**

Для выполнения работы использовалась IDE NetBeans 8.0.2, в которой было создано консольное приложение Java.

Дерево проекта:



В данном приложении console использовались следующие классы:

* 1. **Main.java**

Это Main-класс программы. Здесь реализована основная работа с консолью и вызовом необходимых функций других классов.

* 1. **Base\_Operations.java**

Это основной класс программы, где реализовано всё взаимодействие с базой данных. Здесь находятся методы подключения к базе, а также взаимодействия с ней: просмотра её таблиц, изменения таблицы, вызова хранимой процедуры.

* 1. **exportJSON.java**

Данный класс предназначен для экспорта данных таблицы в JSON документы.

* 1. **importXML.java**

Данный класс осуществляет импорт данных из файлов в формате XML в таблицу Counties.

**Листинг:**

**Класс Main.java:**

|  |
| --- |
| /\*  Main class for console application for data base "weapon shop"  \*/  package main;  import java.io.FileNotFoundException;  import java.sql.\*;  import org.json.simple.parser.ParseException;  /\*\*  \*  \* @author Dell  \*/  public class Main {  /\*\*  \* @param args the command line arguments  \* @throws java.lang.ClassNotFoundException  \* @throws java.sql.SQLException  \* @throws java.lang.InstantiationException  \* @throws java.lang.IllegalAccessException  \* @throws java.io.FileNotFoundException  \* @throws org.json.simple.parser.ParseException  \*/  public static void main(String[] args) throws ClassNotFoundException,  SQLException, InstantiationException, IllegalAccessException,  FileNotFoundException, ParseException  {  // Выбранный пункт меню  int menu\_item;  // Инициализация  Base\_Operations.initialize();  // Получение списка таблиц БД  Base\_Operations.show\_table();  // Процесс работы с меню  while(true){  print\_menu();  switch(menu\_item = Integer.parseInt(Base\_Operations.sc.nextLine())){  case 1:  Base\_Operations.print\_table();  break;  case 2:  Base\_Operations.countries\_add();  break;  case 3:  exportJSON.export\_JSON();  break;  case 4:  importXML.import\_XML();  break;  case 5:  System.out.println("Завершение работы...");  System.exit(0);  break;    default:  System.err.println("Ошибка! Пункт меню с таким номером"  +"отсутствует!\n Повторите ввод!\n");  break;  }  }  }    // Функция вывода меню  public static void print\_menu(){  System.out.println("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.out.println("| WEAPON SHOP CONSOLE APPLICATION |");  System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.out.println("Выберите одну из следующих операций:");  System.out.println("1. Вывод таблицы.");  System.out.println("2. Добавление записи в таблицу CLUB.");  System.out.println("3. Экспорт в JSON любой таблицы");  System.out.println("4. Импорт из XML в таблицу Classes.");  System.out.println("5. Выход из программы.\n");  }  } |

**Класс Base\_Operations.java:**

|  |
| --- |
| package main;  // Класс для выполнения операций с БД  import java.sql.Connection;  import java.sql.DatabaseMetaData;  import java.sql.DriverManager;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.sql.SQLException;  import java.util.Scanner;  import java.util.Vector;  import java.util.logging.Level;  import java.util.logging.Logger;  public class Base\_Operations {  // Переменные  public static String str\_1 = null;  public static String [] str\_2 = null;  public static int cnt\_col = 0;  public static Connection connect = null;  // Объект для выполнения SQL запросов  public static java.sql.Statement rqst = null;  // Класс для работы с консолью  public static Scanner sc = new Scanner(System.in);  // Номер введенной таблицы  public static int table\_number = 0;  public static Vector<String> vec\_tab = new Vector<String>();  //Объект для построения строки  public static StringBuilder sb = new StringBuilder();  // Класс для хранения результатов SQL запроса  public static ResultSet res;  public static DatabaseMetaData metaData;  public static ResultSet reslt;      // Инициализация базы данных  public static void initialize(){  try {  try {  // Иницализация драйвера  Class.forName("org.firebirdsql.jdbc.FBDriver").newInstance();  } catch (InstantiationException | IllegalAccessException ex) {  Logger.getLogger(Base\_Operations.class.getName()).log(Level.SEVERE, null, ex);  }    //Указание пути к БД  String strPath = "jdbc:firebirdsql://localhost/C:\\DATA2.FDB"; // path to database  try {  Class.forName("org.firebirdsql.jdbc.FBDriver").newInstance();  } catch (InstantiationException | IllegalAccessException ex) {  Logger.getLogger(Base\_Operations.class.getName()).log(Level.SEVERE, null, ex);  }    try {  //Подключение к БД  connect = DriverManager.getConnection(strPath, "SYSDBA", "masterkey");  } catch (SQLException ex) {  Logger.getLogger(Base\_Operations.class.getName()).log(Level.SEVERE, null, ex);  }  if (connect == null) {  System.err.println("Ошибка при подключении к базе данных!");}    try {  //Создание класса для выполнения SQL запросов  rqst = connect.createStatement();  } catch (SQLException ex) {  Logger.getLogger(Base\_Operations.class.getName()).log(Level.SEVERE, null, ex);  }  System.out.println("Подключение к БД выполнено успешно!");  } catch (ClassNotFoundException ex) {  Logger.getLogger(Base\_Operations.class.getName()).log(Level.SEVERE, null, ex);  }  }  // Вывод таблиц  public static void show\_table(){  try {  metaData = connect.getMetaData();  reslt=metaData.getTables(str\_1, str\_1, str\_1, str\_2);  while(reslt.next())  {  str\_1=reslt.getString(3);  if(!str\_1.contains("$"))  vec\_tab.add(str\_1);  }  } catch (SQLException ex) {  Logger.getLogger(Base\_Operations.class.getName()).log(Level.SEVERE, null, ex);  }  }    // Печать таблиц  public static void print\_table(){  try  {  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!");  return;  }  if((table\_number > vec\_tab.size()) || (table\_number < 0)){  System.err.println("Ошибка! Несуществующий номер таблицы!");  return;  }  if(table\_number == 0)  {  return;  }  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  // Вывод названия столбцов:  for(int i = 1; i < cnt\_col + 1; i++){  System.out.print(res.getMetaData().getColumnName(i)+  " | ");  }  // Вывод записей 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();  }  catch(SQLException ex)  {  Logger.getLogger(Base\_Operations.class.getName()).log(Level.SEVERE, null, ex);  }  }  // Добавление в таблицу стран-производителей новой страны  public static void countries\_add(){  // Ввод id страны  System.out.println("Введите id страны");  Integer id = sc.nextInt();  if(id < 0)  {  System.err.println("Ошибка! Проверьте правильность ввода! "  + "Id должен быть больше 0");  return;  }  // Ввод название страны-производителя  System.out.println("Введите название новой страны:");  String insrt1 = sc.nextLine();  if (insrt1.length()>25 || insrt1.isEmpty())  {  System.err.println("Ошибка! Проверьте правильность ввода! "  + "Название должно быть короче 25 символов");  return;  }  } |

**Класс exportJSON.java:**

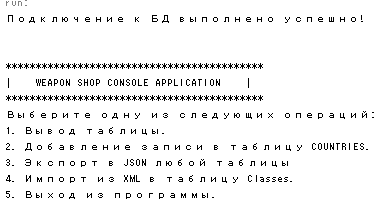
|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package main;  import java.io.FileNotFoundException;  import java.io.File;  import java.io.FileWriter;  import java.io.IOException;  import java.sql.SQLException;  import java.sql.ResultSet;  import java.sql.ResultSetMetaData;  import java.util.logging.Level;  import java.util.logging.Logger;  import static main.Base\_Operations.res;  import static main.Base\_Operations.rqst;  import org.json.simple.JSONArray;  import org.json.simple.JSONObject;  import org.json.simple.parser.JSONParser;  import org.json.simple.parser.ParseException;  import static main.Base\_Operations.sc;  import static main.Base\_Operations.table\_number;  import static main.Base\_Operations.vec\_tab;  /\*\*  \*  \* @author Dell  \*/  public class exportJSON {  // Объекты класса JSON  public static JSONParser parser = new JSONParser();  public static Object obj;  public static String textjson;  public static JSONObject jsonObj;  public static JSONArray jo;  // Адреса файлов JSON  private static final String json\_adr = "C:\\Users\\Dell\\Desktop\\data.json";    // Функция импорта файлов JSON  public static void export\_JSON(){  JSONArray json = new JSONArray();  try {  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!");  return;  }  if((table\_number > vec\_tab.size()) || (table\_number < 0)){  System.err.println("Ошибка! Таблица с таким номером отсутсвует.");  return;  }  if(table\_number == 0){  return;  }  System.out.println();  //Выполнение SQL запроса  res = rqst.executeQuery("SELECT \* from " + vec\_tab.elementAt(table\_number-1));  //making JSON data    ResultSetMetaData rsmd = res.getMetaData();  while(res.next()) {  int numColumns = rsmd.getColumnCount();  JSONObject obj = new JSONObject();  for (int i=1; i<numColumns+1; i++) {  String column\_name = rsmd.getColumnName(i);  if(rsmd.getColumnType(i)==java.sql.Types.ARRAY){  obj.put(column\_name, res.getArray(column\_name));  }  else if(rsmd.getColumnType(i)==java.sql.Types.BIGINT){  obj.put(column\_name, res.getInt(column\_name));  }  else if(rsmd.getColumnType(i)==java.sql.Types.BOOLEAN){  obj.put(column\_name, res.getBoolean(column\_name));  }  else if(rsmd.getColumnType(i)==java.sql.Types.BLOB){  obj.put(column\_name, res.getBlob(column\_name));  }  else if(rsmd.getColumnType(i)==java.sql.Types.DOUBLE){  obj.put(column\_name, res.getDouble(column\_name));  }  else if(rsmd.getColumnType(i)==java.sql.Types.FLOAT){  obj.put(column\_name, res.getFloat(column\_name));  }  else if(rsmd.getColumnType(i)==java.sql.Types.INTEGER){  obj.put(column\_name, res.getInt(column\_name));  }  else if(rsmd.getColumnType(i)==java.sql.Types.NVARCHAR){  obj.put(column\_name, res.getNString(column\_name));  }  else if(rsmd.getColumnType(i)==java.sql.Types.VARCHAR){  obj.put(column\_name, res.getString(column\_name));  }  else if(rsmd.getColumnType(i)==java.sql.Types.TINYINT){  obj.put(column\_name, res.getInt(column\_name));  }  else if(rsmd.getColumnType(i)==java.sql.Types.SMALLINT){  obj.put(column\_name, res.getInt(column\_name));  }  else if(rsmd.getColumnType(i)==java.sql.Types.DATE){  obj.put(column\_name, res.getDate(column\_name));  }  else if(rsmd.getColumnType(i)==java.sql.Types.TIMESTAMP){  obj.put(column\_name, res.getTimestamp(column\_name));  }  else{  obj.put(column\_name, res.getObject(column\_name));  }  }  json.add(obj);  }  } catch(SQLException ex) {  Logger.getLogger(Base\_Operations.class.getName()).log(Level.SEVERE, null, ex);  }  try {    // Writing to a file  File file=new File(json\_adr);  file.createNewFile();  FileWriter fileWriter = new FileWriter(file);  System.out.println("Writing JSON object to file");  System.out.println("-----------------------");    fileWriter.write(json.toJSONString());  fileWriter.flush();  fileWriter.close();    } catch (IOException e) {  e.printStackTrace();  }  }  } |

**Класс ImportXML:**

|  |
| --- |
| package main;  import java.io.FileNotFoundException;  import java.io.File;  import java.io.FileWriter;  import java.io.IOException;  import java.lang.Object;  import javax.xml.parsers.DocumentBuilder;  import javax.xml.parsers.DocumentBuilderFactory;  import javax.xml.parsers.ParserConfigurationException;  import org.xml.sax.SAXException;  import org.w3c.dom.Document;  import org.w3c.dom.Element;  import org.w3c.dom.NodeList;  import static main.Base\_Operations.rqst;  import java.sql.SQLException;  /\*\*  \*  \* @author Dell  \*/  public class importXML {          public static void import\_XML(){  Document dom = null;  Element root = null;  Element id = null;  Element country = null;  DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();    try {  File f = new File("C:\\Users\\Dell\\Desktop\\res.xml");    //Using factory get an instance of document builder  DocumentBuilder builder = factory.newDocumentBuilder();  //parse using builder to get DOM representation of the XML file  Document document = builder.parse(f);  root = document.getDocumentElement();  }  catch(ParserConfigurationException | IOException | SAXException pce)  {  pce.printStackTrace();  }    // для простоты сразу берем message  NodeList cntrs = root.getElementsByTagName("country");  for (int i = 0; i < cntrs.getLength(); ++i) {  id = (Element) root.getElementsByTagName("id").item(i);  country = (Element) root.getElementsByTagName("name").item(i);  try  {  String s = "insert into COUNTRIES values ('" + id.getTextContent()  + "','" + country.getTextContent() + "');";  rqst.executeUpdate(s);  }  catch (SQLException se)  {  System.out.println(se.getMessage());  }    }  }    } |

Взаимодействие приложения с пользователем:

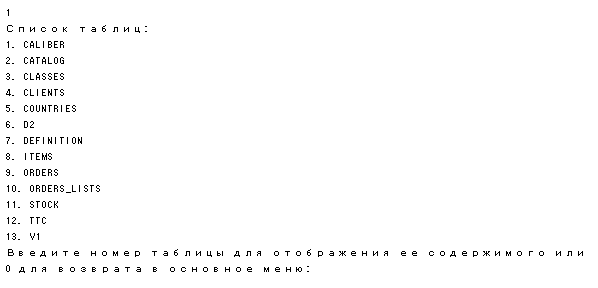
При запуске программы выводится меню возможных действий с базой данных:



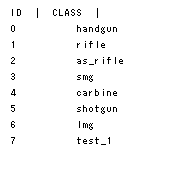
Далее пользователь должен ввести номер команды, которую он хочет выполнить.

**Рассмотрим действия, которые может выполнять созданная программа:**

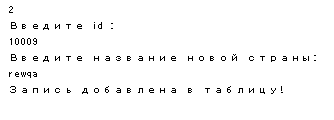
* + 1. Вывод списка всех таблиц базы данных:



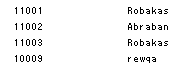
3



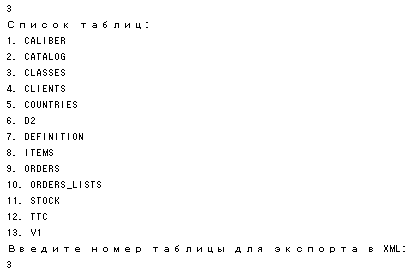
Добавление записи в таблицу COUNTRIES:



Выведем конец таблицы countries:



Экспорт в JSON





Содержимое созданного файла data.json:

|  |
| --- |
| [{"CLASS":"handgun  ","ID":0},{"CLASS":"rifle ","ID":1},{"CLASS":"as\_rifle ","ID":2},{"CLASS":"smg ","ID":3},{"CLASS":"carbine ","ID":4},{"CLASS":"shotgun ","ID":5},{"CLASS":"lmg ","ID":6},{"CLASS":"test\_1 ","ID":7}] |

Экспорт выполнен успешно

Импорт содержимого файла XML:

Файл res.xml:

|  |
| --- |
| <countries>  <country>  <id>11002</id>  <name>Abraban</name>  </country>  <country>  <id>11003</id>  <name>Robakas</name>  </country>  </countries> |

Далее выведем таблицу countries и в конце таблицы получим:

|  |
| --- |
|  |

Следовательно, импорт прошел успешно

**Вывод:**

В результате выполнения лабораторной работы было разработано клиентское приложение, осуществляющее работу с базой данных, созданной ранее.

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.

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).