УДК 004.021

ИССЛЕДОВАНИЕ ТЕХНОЛОГИИ ВЗАИМОДЕЙСТВИЯ С БАЗОЙ ДАННЫХ ДЛЯ АВТОМАТИЗИРОВАННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ ТУРИСТИЧЕСКОГО ЦЕНТРА

А.И. ПОПОВ
(Представлено: канд. техн. наук, доц. И. Б. БУРАЧЕНОК)

В данной статье рассмотрены принципы взаимодействия с реляционными базами данных на языке программирования Java посредством технологии JDBC. Приведены преимущества разделения архитектуры приложения непосредственно на саму реализацию приложения и отделённый от него драйвер. Обосновано применение представленной технологии для реализации интерфейса для разрабатываемого приложения.

В каждом современном языке программирования существует собственный набор методов и средств для разработки системы взаимодействия с базами данных (БД). В языке программирования Java, для данной цели, используется такая технология как – Java DataBase Connectivity (JDBC) – платформенно-независимый промышленный стандарт взаимодействия Java-приложений с различными СУБД, реализованный в виде пакета java.sql, входящего в состав Java SE.[1]

Далее подробнее остановимся на особенностях взаимодействия между реализованным приложением и базой данных с использованием стандартного Application Programming Interface (API) JDBC. Если говорить в целом, то JDBC – это библиотека, которая обеспечивает целый набор интерфейсов для доступа к различным БД для решения поставленных задач. Наиболее распространённые цели, для которых может использоваться JDBC:

– создание соединения с БД;
– создание SQL выражений;
– выполнение SQL – запросов;
– просмотр и модификация полученных записей.

В общем виде JDBC состоит из двух слоёв: JDBC API – Обеспечивает соединение «приложение – JDBC Manager». JDBC Driver API – Обеспечивает соединение «JDBC Manager – драйвер». JDBC API использует менеджер драйверов и специальные драйверы БД для обеспечения подключения к различным базам данных. JBDC Manager проверяет соответствие драйвера и конкретной БД. Он поддерживает возможность использования нескольких драйверов одновременно для одновременной работы с несколькими видами БД.

Схематично, JDBC можно представить в виде архитектуры, представленной на рисунке 1:

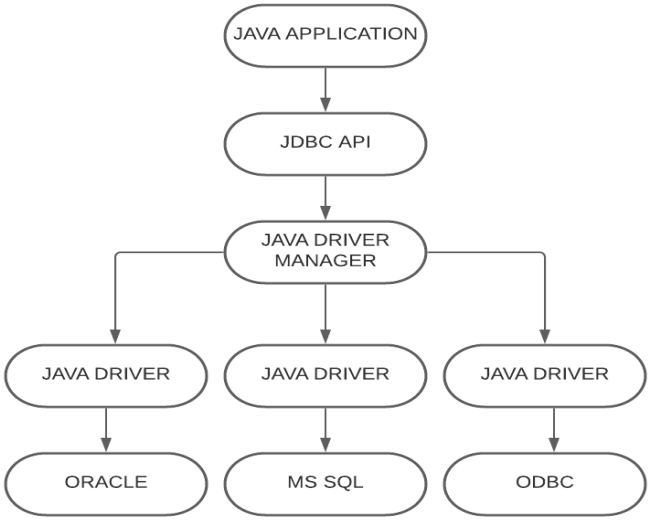

Рисунок 1. – Архитектура взаимодействия JDBC с базой данных

Рассмотрим элементы JDBC по отдельности.

– **Менеджер драйверов (Driver Manager).** Этот элемент управляет списком драйверов БД. Каждой запрос на соединение требует соответствующего драйвера. Первое совпадение даёт нам соединение.

– **Драйвер (Driver).** Этот элемент отвечает за связь с БД. Работать с ним нам приходится крайне редко. Вместо этого чаще используют объекты DriverManager, которые управляют объектами этого типа.

– **Соединение (Connection).** Этот интерфейс обеспечивает нас методами для работы с БД. Все взаимодействия с БД происходят исключительно через Connection.

– **Выражение (Statement).** Для подтверждения SQL-запросов используют объекты, созданные с использованием этого интерфейса.

– **Результат (ResultSet).** Экземпляры этого элемента содержат данные, которые были получены в результате выполнения SQL – запроса. Он работает как итератор и «пробегает» по полученным данным.

– **Исключения (SQL Exception).** Этот класс обрабатывает все ошибки, которые могут возникнуть при работе с БД. [2]

Пример создания подключения к базе данных с использованием JDBC представлен в листинге 1.

Листинг 1 – Пример создания подключения к базе данных с использованием JDBC.

```
public class Connect {
    public static Connection conn = null;
    public static Statement stmt = null;
    public static Connection connectToDatabase() {
        Locale.setDefault(Locale.ENGLISH);
        String url = "jdbc:sqlserver://localhost:1433";
        try {
            if (conn == null) {
                try {
                    SQLServerDriver.class.toString();
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
                } catch (ClassNotFoundException e) {
                    System.out.println(e.getMessage());
                }
                Properties props = new Properties();
                props.setProperty("databaseName", "TyrCentre");
                props.setProperty("user", "tmp");
                props.setProperty("password", "619");
                conn = DriverManager.getConnection(url, props);
                return conn;
            }
        } catch (SQLException e) {
            e.printStackTrace();
            System.out.println(e.getMessage());
        }
        return conn;
    }
    public static Statement getStatement() {
        try {
            if (stmt == null) {
                stmt = conn.createStatement();
                return stmt;
            }
        } catch (SQLException e) {
            System.out.println(e.getMessage());

        }
        return stmt;
```

После установки соединения с базой данных можно взаимодействовать с таблицами при помощи объекта Result. Получение данных из БД с использованием JDBC представлено в листинге 2.

Листинг 2 – Получение данных из БД с использованием JDBC.

```
try {
            ResultSet rs = Connect.stmt.executeQuery("SELECT name\n" +
                    "FROM Place");
            while (rs.next()) {
                list.add(rs.getString(1));
            }
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
        }
```

Таким образом, разделение архитектуры приложения непосредственно на саму реализацию приложения и отделённый от него драйвер имеет ряд преимуществ:

– лёгкость разработки: разработчик может не знать специфики базы данных, с которой работает;

– код практически не меняется, если приложение переходит на другую базу данных (количество изменений зависит исключительно от различий между диалектами SQL);

– нет необходимости устанавливать громоздкую клиентскую программу;

– к любой базе можно подсоединиться через легко описываемый унифицированный указатель ресурса URL (Uniform Resource Locator).

Следовательно, используя данную технологию доступа к базам данных можно организовать взаимодействие приложения с SQL-сервером с минимальными затратами ресурсов. Так как JDBC является частью языка Java, то не потребуется использование сторонних ресурсов. Кроме того, использование данной технологии нивелирует привязанность к какой-то определённой базе данных, приложение может взаимодействовать с другой базой данных с минимальными изменениями в настройках подключения и заменой используемого драйвера подключения при этом основной код приложения останется неизменным.

ЛИТЕРАТУРА

1. Java Database Connectivity. [Электронный ресурс] / Wikipedia. – Режим доступа: https://ru.wikipedia.org/wiki/Java_Database_Connectivity/. – Дата доступа: 19.09.2020.
2. Введение в JDBC. [Электронный ресурс] / Codeflow. – Режим доступа: https://www.codeflow.site/ru/article/java-jdbc/. – Дата доступа: 19.09.2020.