**JDBC**

**Java Database Connectivity (JDBC)** – это встроенная библиотека в Java для взаимодействия с различными СУБД (реализован в виде библиотеки java.sql).

Для соединения приложения и СУБД необходим драйвер (посредник), для каждой СУБД свой драйвер, его необходимо скачать. Для загрузки драйвера в память JVM используется метод: Class.forName (“полное имя к классу драйвера”), но в современных версиях java это можно опустить, загрузчик сам подгрузит драйвер.

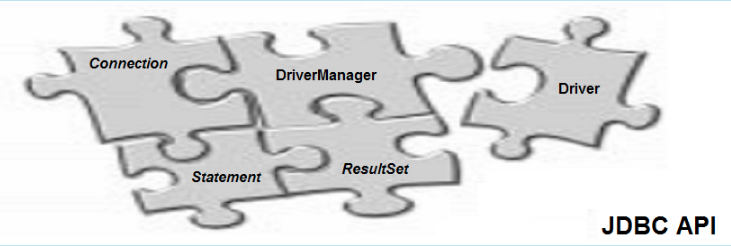
Минусы JDBC (решение Spring + hibernate):

- Не информативные сообщения (sqlexaption в основном)

- Низкоуровневый способ взаимодействия с БД

- Большое кол-во самописного кода

- отсутствие ORM



**DriverManager** – это класс который управляет списком загруженных драйверов.

**Driver** – это интерфейс (java.sql.Driver) содержащий различные методы для соединения с определённой БД, этот интерфейс обязан наследовать - конкретный драйвер определённой БД (oracle.jdbc.driver.OracleDriver).

**Connection** – это интерфейс, представляет в JDBC сеанс работы с базой данных, всё взаимодействие с БД происходит через него, даёт методы создавать Statement, PreparedStatement, CallableStatement.

**Statement** – интерфейс, используется для выполнения запросов в БД, от него наследуются такой интерфейс как PreparedStatement <- CallableStatement.

**ResultSet** – это интерфейс, содержит результирующий набор базы данных. По нему можно идти вперёд и назад, получая данные.

**Для работы с Postgresql в Java:**

1) Первое что необходимо сделать это скачать драйвер (jar) проще это сделать прописав maven зависимость либо, создать папку lib в web -> WEB-INF (и туда закинуть архив драйвера jar - скачать).

2) Далее в коде необходимо подгрузить класс в JVM:

Class.forName(“org.postgresql.Driver”);

3) Получаем соединение с БД:

Connection connection = DriverManager.getConnection (URL, name, password);

URL для MySQL: jdbc:mysql://localhost:3306/park?serverTimezone=UTC

URL для PostgresSQL: Connection conn = DriverManager.getConnection

("jdbc:postgresql://localhost:5432/db\_java\_app", "postgres", "Qwert070123");

**Есть 3 типа обьектов для выполнения SQL запросов:**

Наследственность: Statement <- PreparedStatement <- CallableStatement;

**1)** Statement statement = connection.**createStatement**();

Используется для простых запросов – без параметров. Сам SQL запрос передаётся в методы executeQuery() или executeUpdate().

ResultSet resultSet = statement.executeQuery("sql запрос");

**2)** PreparedStatement prepStatement = connection.**prepareStatement**("sql запрос");

Предназначен для выполнения как простых запросов, так и подготовленных с параметрами. SQL запрос передаётся в метод создания PreparedStatement, и по вызову метода executeQuery() или executeUpdate() выполняется запрос.

prepStatement.setString(1, login);

prepStatement.setString(2, email);

ResultSet resultSet = statement.executeQuery();

**3)** CallableStatement callStatement = **connection**. prepareCall("{call setGoodsData(?, ?)}");

Используется для вызовов хранимых процедур. Почти нигде не используется.

**Statement предоставляет 3 основных метода для выполнения SQL-выражений:**

**executeQuery()** – используется в запросах, результатом которых является один единственный набор значений (SELECT). Возвращает объект ResultSet.

**executeUpdate()** - используется в остальных SQL выражениях (INSERT, UPDATE, DELETE, CREATE TABLE, DROP TABLE). И возвращает int число затронутых строк.

**execute()** - используется, когда операторы SQL возвращают более одного набора данных, более одного счетчика обновлений или и то, и другое. Такая возможность редко используется программистами.