Java Database Connectivity (JDBC)

JDBC is an application programming interface (API) [https://en.wikipedia.org/wiki/Java_Database_Connectivity] for the programming language Java to access SQL databases.

A JDBC connection is established by a client application in Java to a relational database server. It is necessary to provide a JDBC driver for particular database server (usually provided by its vendor).

Oracle

Oracle Database JDBC driver [https://www.oracle.com/technetwork/database/application-development/jdbc/downloads/] is in the `ojdbc8.jar` file.

Import the JDBC API and Oracle JDBC Driver

```
import oracle.jdbc.pool.OracleDataSource;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
```

Configure OracleDataSource to a Database Server

- OracleDataSource [https://docs.oracle.com/en/database/oracle/oracle-database/18/jajdb/oracle/jdbc/datasource/OracleDataSource.html] documentation

```
OracleDataSource ods = new OracleDataSource();
ods.setURL("jdbc:oracle:thin:@//gort.fit.vutbr.cz:1521/orclpdb");
ods.setUser("xlogin12");
ods.setPassword("myPasswordToOracleServer");
```

Create a Connection to the Server

- Connection [https://docs.oracle.com/en/java/javase/11/docs/api/java.sql/java/sql/Connection.html] documentation
- SQLException [https://docs.oracle.com/en/java/javase/11/docs/api/java.sql/java/sql/SQLException.html] documentation

```
try (Connection conn = ods.getConnection()) {
  // ... a usage of the connection
} catch (SQLException sqlEx) {
  System.err.println("SQLException: " + sqlEx.getMessage());
}
```

PostgreSQL

PostgreSQL JDBC driver [https://jdbc.postgresql.org/download.html] is in the `postgresql-*.jar` file.

Import the JDBC API

Applications do not need to explicitly import or load the `org.postgresql.Driver` class because the pgjdbc driver jar supports the Java Service Provider mechanism.

```
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
```

Create a Connection to the Server

- connection parameters [https://jdbc.postgresql.org/documentation/head/connect.html]
- DriverManager [https://docs.oracle.com/en/java/javase/11/docs/api/java.sql/java/sql/DriverManager.html] documentation
- Connection [https://docs.oracle.com/en/java/javase/11/docs/api/java.sql/java/sql/Connection.html] documentation
- SQLException [https://docs.oracle.com/en/java/javase/11/docs/api/java.sql/java/sql/SQLException.html] documentation

```
Properties props = new Properties();
props.setProperty("user", "xlogin12");
props.setProperty("password", "myPasswordToPgSqlServer");
props.setProperty("ssl","true");
try (Connection conn =
DriverManager.getConnection("jdbc:postgresql://gort.fit.vutbr.cz/stud
 {
  // ... a usage of the connection
} catch (SQLException sqlEx) {
  System.err.println("SQLException: " + sqlEx.getMessage());
}
```

Generic (Vendor Indepedent)

Please note that `dual` table is a table provided by Oracle.

Create a Statement and Execute a SQL Query

Plain Statement

- Statement [https://docs.oracle.com/en/java/javase/11/docs/api/java.sql/java/sql/Statement.html] documentation

```
try (Statement stmt = conn.createStatement()) {
  // select something from the system's dual table
  try (ResultSet rset = stmt.executeQuery("select 1+2.0 as col1,
'foo' as col2 from dual")) {
    // ... a usage of the result set
  }
}
```

Prepared Statement

- PreparedStatement [https://docs.oracle.com/en/java/javase/11/docs/api/java.sql/java/sql/PreparedStatement.html] documentation

Use the premared statements to protect against SQL Injection attacks [https://en.wikipedia.org/wiki/SQL_injection] and to optimize a repeating execution of the same statement on different values.

```
// select something from the system's dual table
try (PreparedStatement pstmt = conn.prepareStatement("select ?+? as
col1, ? as col2 from dual")) {
  pstmt.setInt(1, 1);
  pstmt.setDouble(2, 2.0);
  pstmt.setString(3, "foo");
  try (ResultSet rset = pstmt.executeQuery()) {
    // ... a usage of the result set
  }
}
```

Process Query Results

- ResultSet [https://docs.oracle.com/en/java/javase/11/docs/api/java.sql/java/sql/ResultSet.html] documentation

```
while (rset.next()) {
  System.out.println("col1: '" + rset.getDouble(1) + "'\tcol2: '" +
rset.getString(2) + "'");
}
```

Data Manipulation Language (DML) Statements

DML statements do not return any data, so there is no `ResultSet`. Such statements should be executed by `.executeUpdate()` method.

```
try (PreparedStatement pstmt = conn.prepareStatement("insert values
(?,?) into myproduct")) {
  for (int i = 0; i <= 10; i++) {
    pstmt.setInt(1, i);
    pstmt.setString(2, "foo" + i);
    pstmt.executeUpdate();
  }
}
```

Transactions

```
boolean previousAutoCommit = conn.getAutoCommit();
conn.setAutoCommit(false);
try {
  // ... executing SQL statements
  Savepoint svpt1 = conn.setSavepoint("SVTP1");
  // ... executing another SQL statements after the save-point
  if (someCondition) {
    conn.rollback(svtp1);
    // ... rollback to the save-point and try another/different SQL
statements
  }
  conn.commit();
} finally {
  conn.setAutoCommit(previousAutoCommit);
}
```