JDBC

Drivers

* 4 types of drivers:
  1. JDBC-ODBC bridge driver
  2. Native API driver (partially Java driver)
  3. Network Protocol driver (fully Java driver)
  4. Thin driver (fully Java driver) – most often used
     + The thin driver converts JDBC calls directly into the vendor-specific database protocol
     + That is why it is known as thin driver
     + It is fully written in Java language

Steps to connect Java app to Database

1. Load the dependency into pom.xml file
   * Dependencies can usually be found by typing in “postgres jdbc maven dependency”
     + In our case it’s postgres but just type in the name of whatever database you are using
2. Load an appropriate driver (by using maven)
   * Search google for “PostgreSQL jdbc maven driver”
   * This will go in between the <dependencies></dependencies> tag in the pom.xml file
   * In your Java class you will write Class.forName("org.postgresql.Driver");
     + This will throw a checked exception, either use ‘throws’ or a try/catch block
3. Create a connection
   * Use the Connection interface to create a connection
     + In you Java class write Connection conn = DriverManager(url,username,password);
     + This will throw a checked exception, either use ‘throws’ or a try/catch block
4. Create the statement
   * Statement:
     + Statement stmt = conn.createStatement();
   * PreparedStatement
     + PreparedStatement pstmt = conn.prepareStatement();
5. Execute SQL Statements (Statement) / Pre-compiled SQL Statements (PreparedStatement)
   * Statement
     + For SELECT statement execute ‘executeQuery()’ method (DQL)
     + For INSERT / UPDATE statement execute ‘executeUpdate()’ method (DML)
     + For CREATE / ALTER / DROP statement execute ‘execute()’ method (DDL)
   * PreparedStatement
     + Execute ‘prepareStatement()’ method
       - Then execute ‘pstmt.method(Placeholder position, value to insert)’
       - Then execute ‘pstmt.executeUpdate()’ method
   * Methods and return types
     + ResultSet stat.executeQuery() -DQL
     + int stat.executeUpdate() -DML
     + boolean stat.execute() -DDL
6. Store the results
   * ResultSet res = stmt.executeQuery(SELECT \* FROM public.product);
7. Loop through the result
   * while(res.next()) {

System.out.print(res.getInt(columnIndex:1 + “ “));

System.out.print(res.getString(columnIndex3 + “ “));

System.out.print(res.getString(columnIndex:3 + “ “));

}

1. Close the connection
   * stmt.close();
   * conn.close();

Difference between Statement and PreparedStatement

Both are used for accessing the database

Statement PreparedStatement

Use if SQL query executed only once Use if SQL query executed multiple times

Can’t pass parameters at runtime Can pass parameters at runtime

Used for CREATE/ALTER/DROP Used for queries to be executed multiple times

Used for DDL statements Used for any SQL query

Low performance Better performance than Statement

Base interface Extends Statement interface

Used for normal SQL queries Used for dynamic SQL queries

Can’t read binary data Can read binary data

Can’t write binary data Can write binary data

No binary protocol used for communication Binary protocol used for communication