JDBC & ODBC

Dept. of Computer Science

Hanyang University

Accessing SQL from a Programming Language

- A database programmer must have access to a general-purpose programming language for at least two reasons
  - Not all queries can be expressed in SQL, since SQL does not provide the full expressive power of a general-purpose language.
  - Non-declarative actions -- such as printing a report, interacting with a user, or sending the results of a query to a graphical user interface -- cannot be done from within SQL.
- There are two approaches to accessing SQL from a general-purpose programming language
  - Dynamic SQL: a general-purpose program can connect to and communicate with a database server using a collection of functions
    - Allows the program to construct an SQL query as a character string at runtime, submit the query, and retrieve the result into program variables a tuple at a time
    - Examples: JDBC, ODBC
  - Embedded SQL: embeds SQL statements in a program to interact with a DB server.
    - The SQL statements are translated at compile time into function calls.
    - At runtime, these function calls connect to the database using an API that provides dynamic SQL facilities.

JDBC

- A Java API for communicating with database systems supporting SQL.
- JDBC supports a variety of features for querying and updating data, and for retrieving query results.
- JDBC also supports metadata retrieval, such as querying about relations present in the database and the names and types of relation attributes.
- Model for communicating with the database:
  - Open a connection
  - Create a "statement" object
  - Execute queries using the statement object to send queries and fetch results
  - Exception mechanism to handle errors

- JDBC Basics Tutorial
  - https://docs.oracle.com/javase/tutorial/jdbc/index.html

JDBC Driver

- JDBC Drivers Downloads
  - https://www.oracle.com/kr/database/technologies/appdev/jdbc-downloads.html

- MySQL JDBC Connector jar Downloads
  - https://dev.mysql.com/downloads/connector/j/

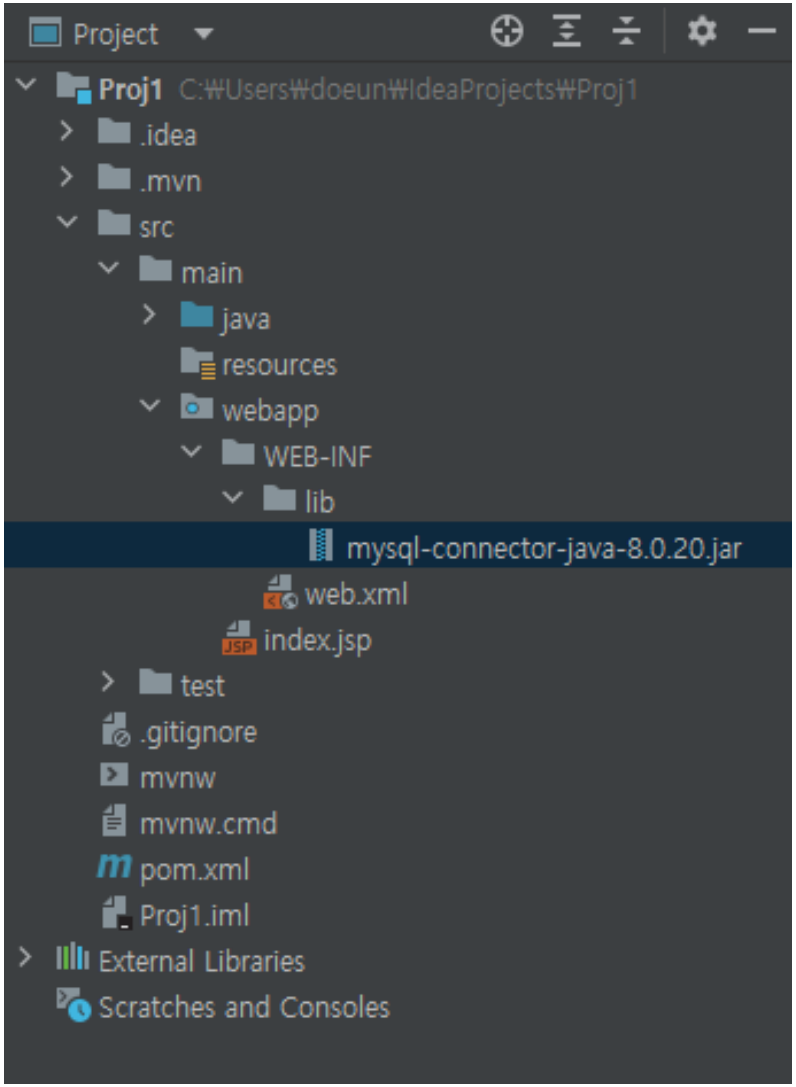
윈도우는 올린파일로 하
면됨

Connector/J Version	JDBC Version	MySQL Server Version	JRE Supported	JDK Required for Compilation	Status
8.0	4.2	5.7, 8.0	8.x	8.x	

https://dev.mysql.com/doc/connector-j/8.0/en/connector-j-versions.html

JDBC Driver

- JDBC 드라이버 설치한 모습



Implementation

[1] JDBC Driver Loading

static Class<?> forName(String className);

ex.

```
Class.forName("com.mysql.jdbc.Driver");
```

[2] DBMS Server Connection USERDAO

static Connection getConnection(String url, String user, String pwd)

ex.

```
Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3307/abc", dbID, dbPassword);
```

mysql protocol 서버주소 서버포트 dbName

[3] Statement Object

Statement createStatement()

ex.

```
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery("SELECT * FROM A WHERE a=?");
```

[4] PreparedStatement Object (vs. Statement Object)

PreparedStatement prepareStatement(String sql)

ex.

```
PreparedStatement pstmt = conn.prepareStatement("insert into test values(?,?)");
pstmt.setString(1, userID); //pstmt의 첫 번째 물음표에 id 변수 값을 문자열 타입으로 설정
pstmt.setString(2, userPassword); //pstmt의 두 번째 물음표에 pwd 변수 값을 문자열 타입으로 설정
pstmt.executeUpdate();
```

Implementation

SQL 문 실행

int executeUpdate(String sql) **vs.** **ResultSet executeQuery(String sql)**

```
SQL > update test set pwd='55' where id='aa';

1 row updated.

SQL > delete from test;

5 row deleted.

SQL > insert into test values<'aa', '11'>;

1 row created.

SQL > insert into test value<'bb', '22'>;

1 row created.

SQL > _
```

```
SQL > select * from test;

ID                  PWD
------------------  ------------------
aa                  11
bb                  22

SQL > _
```

Implementation

SQL 문 실행

int executeUpdate(String sql) vs. **ResultSet executeQuery(String sql)**

```
SQL > update test set pwd='55' where id='aa';

1 row updated.

SQL > delete from test;

5 row deleted.

SQL > insert into test values<'aa', '11'>;

1 row created.

SQL > insert into test value<'bb', '22'>;

1 row created.

SQL > _
```

```
SQL > select * from test;

ID              PWD
--------------- ---------------
aa              11
bb              22

SQL > _
```

ResultSet

Implementation

SQL 문 실행

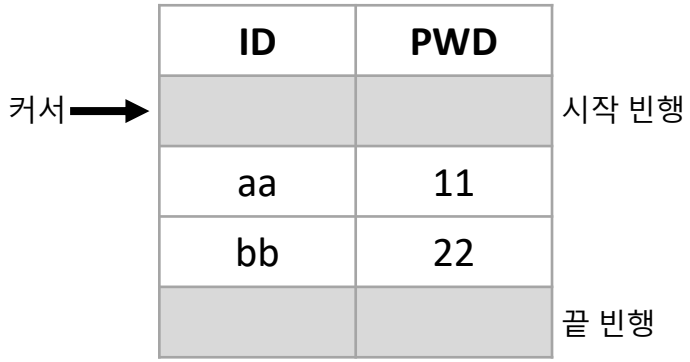
ResultSet executeQuery(String sql)

```
SQL > select * from test;

ID                  PWD
------------------  ------------------
aa                  11
bb                  22

SQL > _
```

ResultSet

```
ResultSet rs = stmt.executeQuery("SELECT * FROM test");
```

위와 같은 SQL 명령문을 실행했을 때 ResultSet 객체 rs가 가지는 값은



이와 같습니다

(cf.) ODBC

- Open DataBase Connectivity (ODBC) standard
  - standard for application program to communicate with a database server.
  - application program interface (API) to
    - open a connection with a database,
    - send queries and updates,
    - get back results.

- Applications such as GUI, spreadsheets, etc. can use ODBC