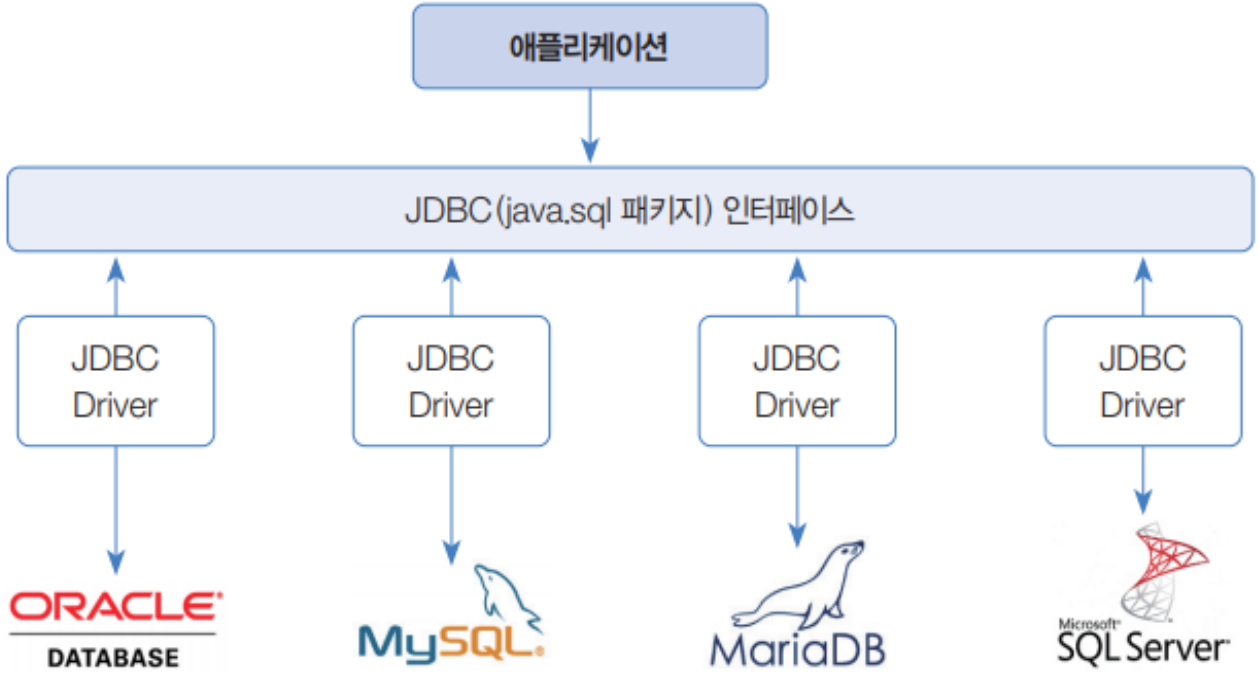
JDBC

사이사이 스터디

중요 내용 리뷰

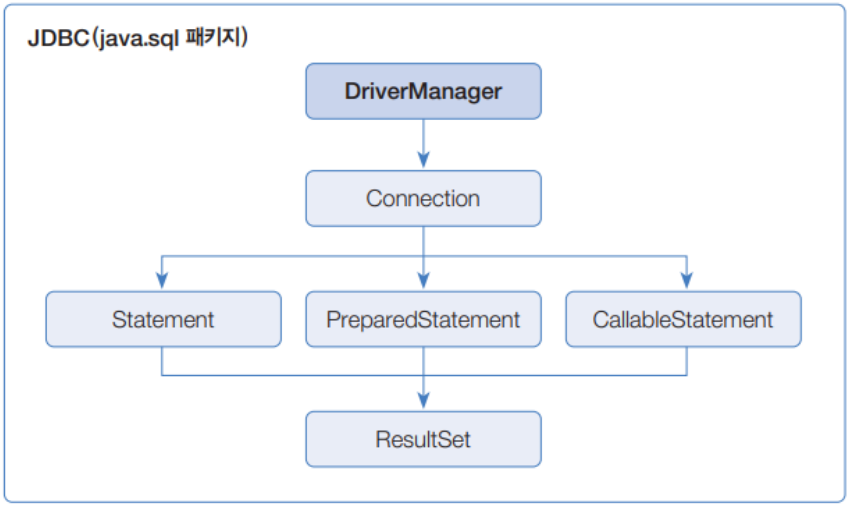
JDBC

- JDBC 라이브러리
  - 자바는 데이터베이스(DB)와 연결해서 데이터 입출력 작업을 할 수 있도록 JDBC 라이브러리 (java.sql 패키지)를 제공
  - JDBC는 데이터베이스 관리시스템(DBMS)의 종류와 상관없이 동일하게 사용할 수 있는 클래스와 인터페이스로 구성



JDBC Driver

- JDBC 인터페이스를 구현한 것으로, DBMS마다 별도로 다운로드받아 사용
- DriverManager : JDBC Driver를 관리하며 DB와 연결해서 Connection 구현 객체를 생성
- Connection 인터페이스: Statement, PreparedStatement, CallableStatement 구현 객체를 생성하며, 트랜잭션 처리 및 DB 연결을 끊을 때 사용
- Statement 인터페이스: SQL의 DDL과 DML 실행 시 사용
- PreparedStatement: SQL의 DDL, DML 문 실행 시 사용.
  매개변수화된 SQL 문을 써 편리성과 보안성 유리
- CallableStatement: DB에 저장된 프로시저를 호출
- ResultSet: DB에서 가져온 데이터를 읽음



JDBC Driver 설치

- 로컬 PC에 Oracle을 설치하면 JDBC Driver 파일 찾을 수 있음

DB 연결

- Class.forName() 메소드는 문자열로 주어진 JDBC Driver 클래스를 BuildPath에서 찾고, JDBC Driver를 메모리로 로딩

```
Class.forName("oracle.jdbc.OracleDriver");
```

```
Connection conn = DriverManager.getConnection("연결 문자열", "사용자", "비밀번호");
```

JDBC Driver

```
String sql = "" +
  "INSERT INTO users (userid, username, userpassword, userage, useremail) " +
  "VALUES (?, ?, ?, ?, ?)";
```

```
PreparedStatement pstmt = conn.prepareStatement(sql);
```

```
pstmt.setString(1, "winter");
pstmt.setString(2, "한겨울");
pstmt.setString(3, "12345");
pstmt.setInt(4, 25);
pstmt.setString(5, "winter@mycompany.com");
```

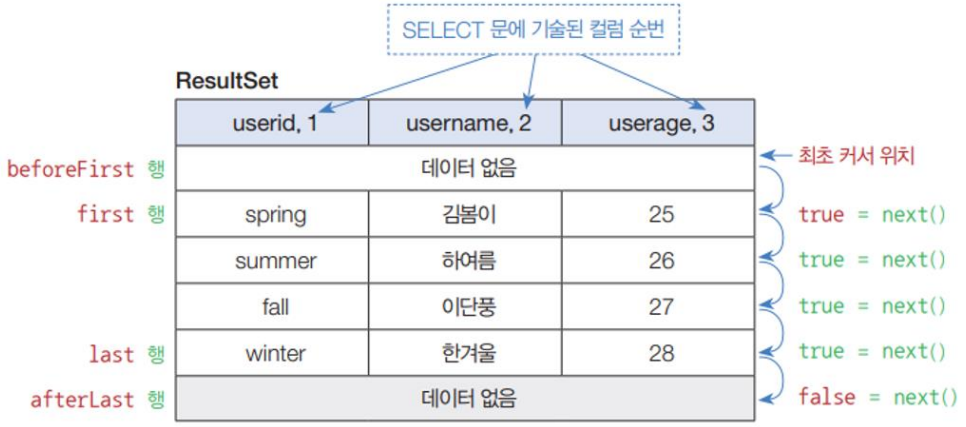
```
int rows = pstmt.executeUpdate();
```

```
String sql = "DELETE FROM boards WHERE bwriter=?";
PreparedStatement pstmt = conn.prepareStatement(sql);
pstmt.setString(1, "winter");
int rows = pstmt.executeUpdate();
```

ResultSet

- SELECT 문에 기술된 컬럼으로 구성된 행(row)의 집합

```
SELECT userid, username, userage FROM users
```



- 커서cursor가 있는 행의 데이터만 읽을 수 있음
- first 행을 읽으려면 next() 메소드로 커서 이동

```
boolean result = rs.next();
```

1개의 데이터 행만 가져올 경우

```
ResultSet rs = pstmt.executeQuery();
if(rs.next()) {
  //첫 번째 데이터 행 처리
} else {
  //afterLast 행으로 이동했을 경우
}
```

n개의 데이터 행을 가져올 경우

```
ResultSet rs = pstmt.executeQuery();
while(rs.next()) {
  //last 행까지 이동하면서 데이터 행 처리
}
//afterLast 행으로 이동했을 경우
```

ResultSet

- 커서가 있는 데이터 행에서 각 컬럼의 값은 Getter 메소드로 읽음
- SELECT 문에 연산식이나 함수 호출이 포함되어 있다면 컬럼 이름 대신에 컬럼 순번으로 읽어야 함

컬럼 이름으로 읽기

```
String userId =
  rs.getString("userid");
String userName =
  rs.getString("username");
int userAge = rs.getInt("userage");
```

컬럼 순번으로 읽기

```
String userId = rs.getString(1);
String userName = rs.getString(2);
int userAge = rs.getInt(3);
```

```
SELECT userid, userage - 1
FROM users
```

```
String userId =
  rs.getString("userid");
int userAge = rs.getInt(2);
```

ResultSet

- SELECT 문 prepareStatement() 메소드로부터 PreparedStatement를 얻고, ?에 값을 지정

```
String sql = "" +
  "SELECT userid, username, userpassword, userage, useremail " +
  "FROM users " +
  "WHERE userid=?";
```

```
PreparedStatement pstmt = conn.prepareStatement(sql);
pstmt.setString(1, "winter");
```

- executeQuery() 메소드로 SELECT 문을 실행해서 ResultSet을 얻음.
- if 문을 이용해서 next() 메소드가 true를 리턴할 경우에는 데이터 행을 User 객체에 저장하고 출력

```
ResultSet rs = pstmt.executeQuery();
if(rs.next()) {                          //1개의 데이터 행을 가져왔을 경우
  User user = new User();
  user.setUserId(rs.getString("userid"));
  user.setUserName(rs.getString("username"));
  user.setUserPassword(rs.getString("userpassword"));
  user.setUserAge(rs.getInt(4));         //컬럼 순번을 이용해서 컬럼 지정
  user.setUserEmail(rs.getString(5));    //컬럼 순번을 이용해서 컬럼 지정
  System.out.println(user);
} else {                                 //데이터 행을 가져오지 않았을 경우
  System.out.println("사용자 아이디가 존재하지 않음");
}
```

Java 프로젝트 유형 – 빌드 도구

- Ant 프로젝트 : build.xml에 추가할 라이브러리(의존성), 빌드 방법 등에 대한 설정을 지정
  . 외부 라이브러리 파일 (jar)를 포함 → 프로젝트 파일이 커짐
  . IntelliJ 기본 프로젝트

- Maven : pom.xml 파일에 추가할 라이브러리(의존성), 빌드 방법 등에 대한 설정을 지정
  . 외부 라이브러리 파일을 직접 포함하기 보다 Remote Repository에서 찾아서 빌드시 포함
  → 프로젝트 파일이 작아 짐

- Gradle : build.gradle 파일에 추가할 라이브러리(의존성), 빌드 방법 등에 대한 설정을 지정
  . 외부 라이브러리 파일을 직접 포함하기 보다 Remote Repository에서 찾아서 빌드시 포함
  . Maven에 비해 캐싱을 사용하여 빌드 속도가 빠름
  . 설정 파일이 Xml 문법이 아니고 groovy 언어로 되어 있음

Java 프로젝트 유형 – 빌드 도구

- Ant : build.xml , ant.xml …

```
<target depends="init,deps-jar,-pre-pre-compile,-pre-compile,-do-compile,-post-compile" description="Compile project." name="compile"/>
  <target depends="init,deps-jar,-pre-pre-compile" name="-do-compile-single">
      <fail unless="javac.includes">Must select some files in the IDE or set javac.includes</fail>
      <webproject2:javac excludes="" gensrcdir="${build.generated.sources.dir}" includes="${javac.includes}"/>
      <copy todir="${build.classes.dir}">
          <fileset dir="${src.dir}" excludes="${build.classes.excludes},${excludes}" includes="${includes}"/>
      </copy>
  </target>
</target>
```

- Maven : pom.xml

```
<build>
<plugins>
  <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.1</version>
                <configuration>
                  <source>1.8</source>
                  <target>1.8</target>
                  <compilerArguments>
                                <endorseddirs>${endorsed.dir}</endorseddirs>
                  </compilerArguments>
                  <debug>false</debug>
                </configuration>
  </plugin>
</build>
```

Java 프로젝트 유형 – 빌드 도구

- Gradle : build.gradle

```
android {
    compileSdk 32

    defaultConfig {
        applicationId "net.stayonbox.soxbeacon"
        minSdk 31
        targetSdk 32
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }
}
dependencies {
    implementation 'androidx.appcompat:appcompat:1.4.1'
    implementation 'com.google.android.material:material:1.7.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
}
```