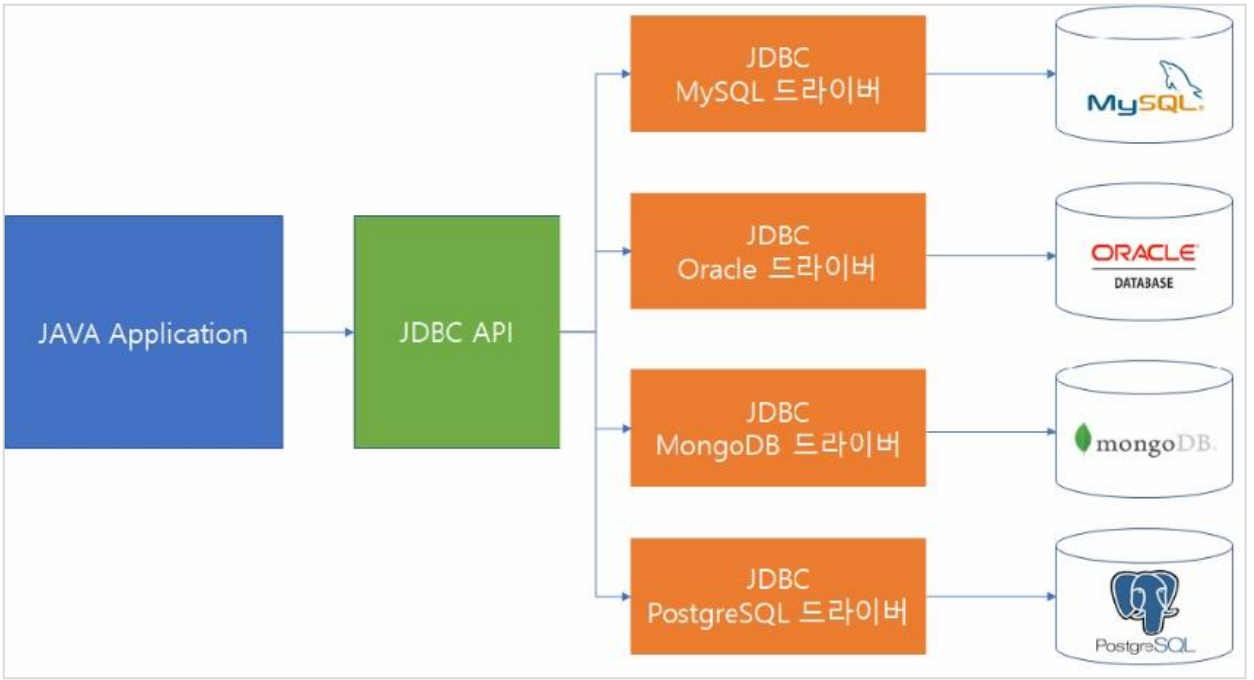
13장. JDBC Connection

오라클 DB 연동

JDBC(Java Database Connectivity)

◆ JDBC(Java Database Connectivity) 정의와 사용

- 자바 애플리케이션에서 DBMS에 연결해주는 기능을 갖는 API이다.
- 오라클, MySQL, MS-SQL 개발사가 구현한 '드라이버'가 존재함

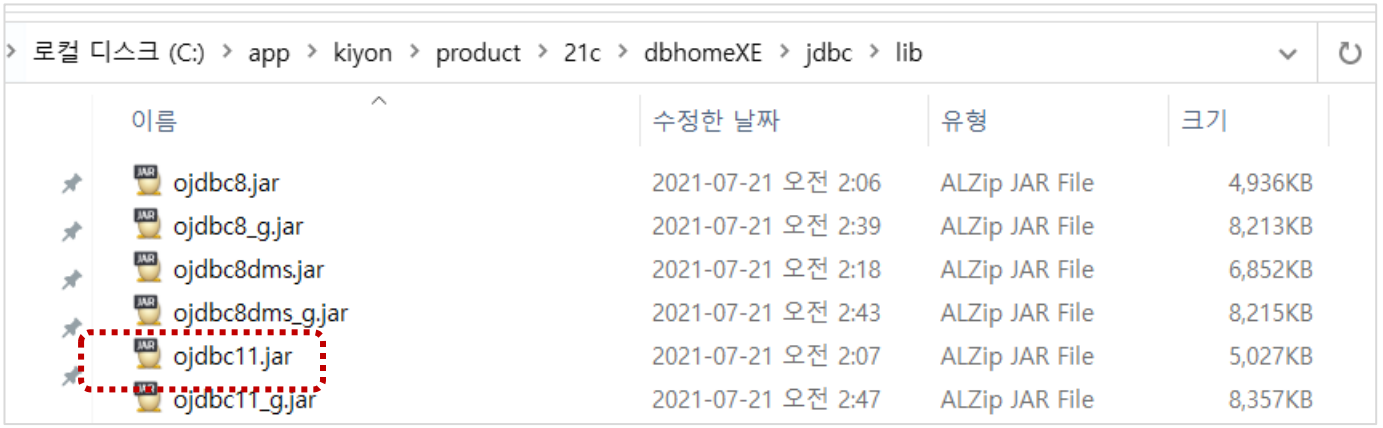


JDBC(Java Database Connectivity)

◆ JDBC를 이용한 데이터베이스 연동하기

ojdbc 드라이버 구하기

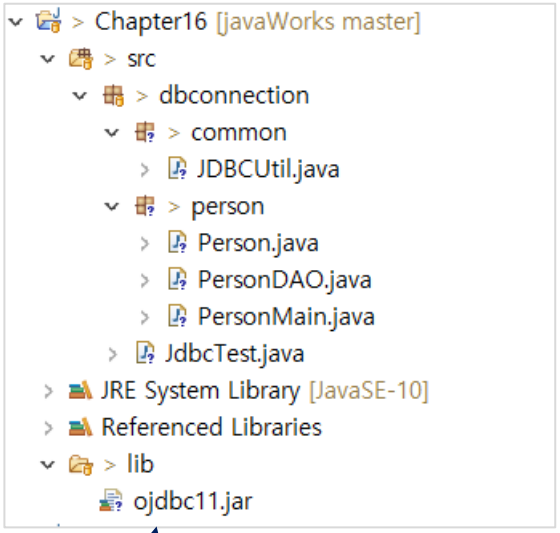
1. 오라클 설치 경로 2. sql developer 설치 경로

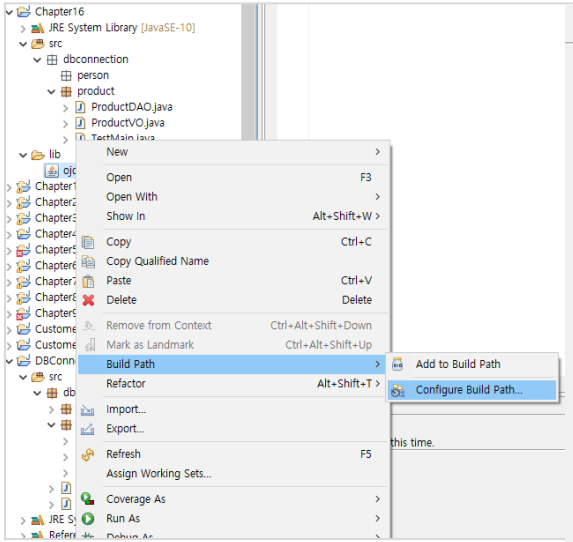


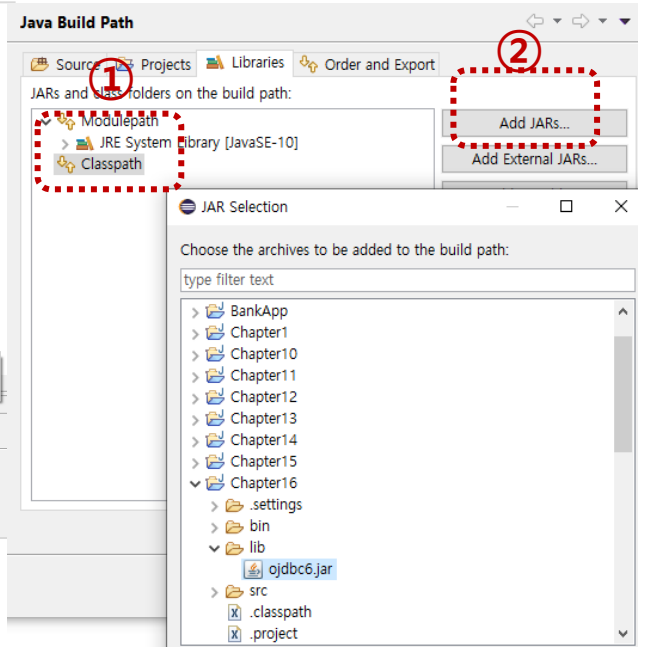
JDBC(Java Database Connectivity)

◆ JDBC를 이용한 데이터베이스 연동하기

ojdbc 이클립스 프로젝트에 복사하기





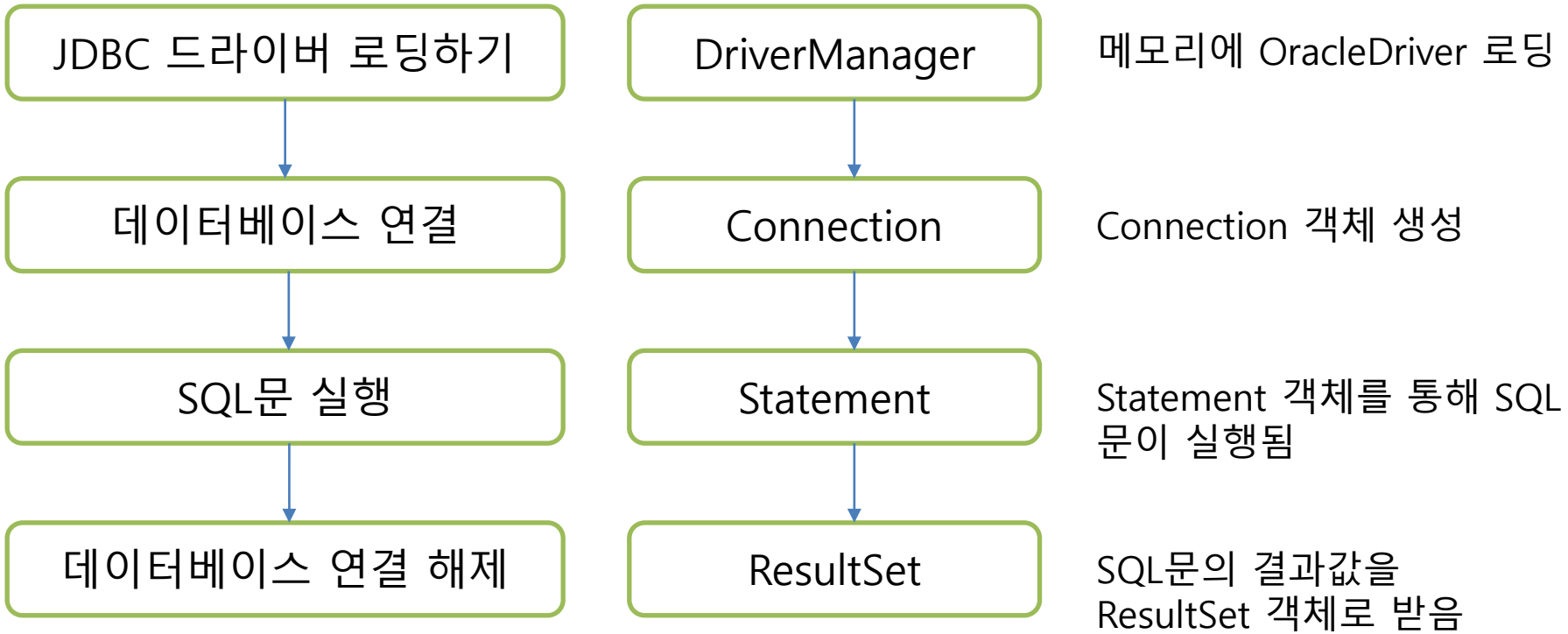


1. 프로젝트 만든후, lib폴더 만들기
2. 오라클 드라이버 .jar파일 복사
3. 클래스 패스 설정

프로젝트 > 우측마우스 > Build Path > Cofigure Build Path > Libraries(Classpath) > Add JARs

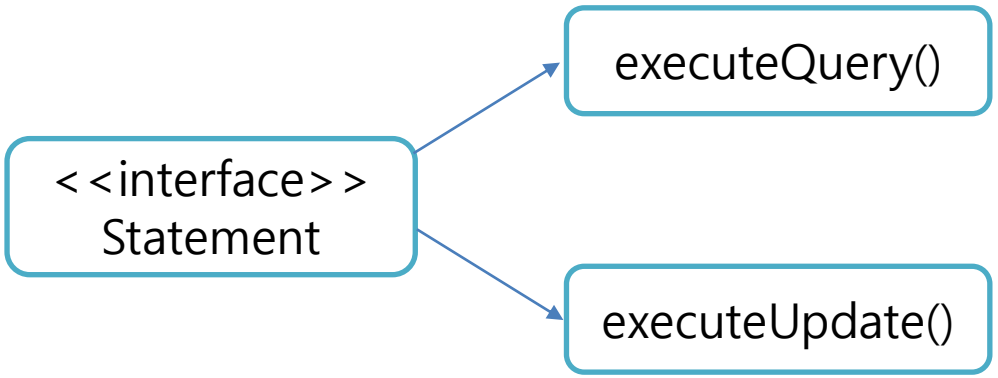
JDBC(Java Database Connectivity)

➢ 데이터베이스 연결 순서



JDBC(Java Database Connectivity)

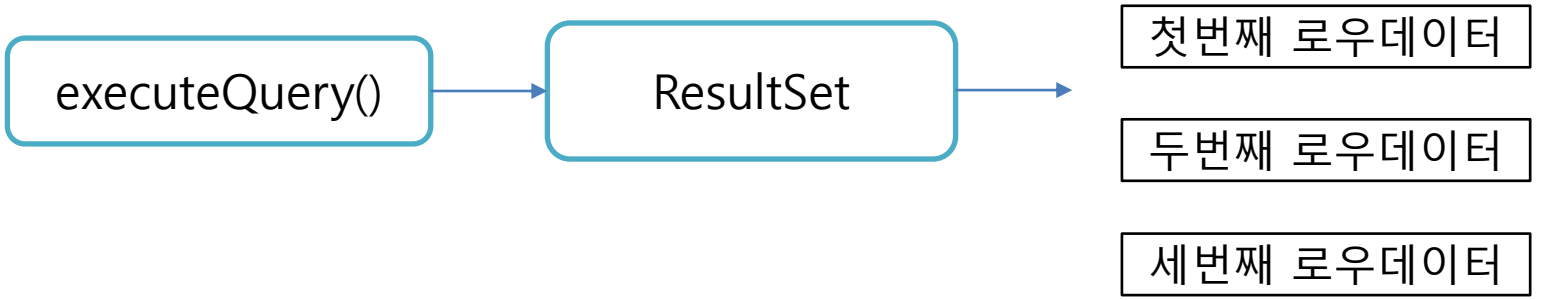
➢ Statement 객체 살펴보기



SQL문 실행 후 여러 개의 결과값 생기는 경우 사용. 예)select

SQL문 실행 후 테이블의 내용만 변경되는 경우 사용. 예)insert, delete, update

executeQuery() 실행 후 반환 되는 레코드셋



연결 테스트(Connection Test)

```java
public class JdbcTest {

    private static String driverClass = "oracle.jdbc.OracleDriver";       //오라클 드라이버
    private static String url = "jdbc:oracle:thin:@localhost:1521:xe";    //db 경로 포트-1521
    private static String username = "system";                            //사용자 이름
    private static String password = "12345";                             //사용자 비밀번호


    public static void main(String[] args) {
        //연결 객체 선언
        Connection conn = null;

        try {
            Class.forName(driverClass);
            System.out.println("Oracle 드라이버 로딩");
            conn = DriverManager.getConnection(url, username, password);
            System.out.println("Connection 객체 생성 : " + conn);
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        }finally {
            try {
                conn.close();
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }
}
```

```
Oracle 드라이버 로딩
Connection 객체 생성 : oracle.jdbc.driver.T4CConnection@5aa9e4eb
```

SQL 디벨로퍼 – 테이블 생성

◆ 테이블 만들기(생성)

```
-- person 테이블 생성
CREATE TABLE person(
    userId     VARCHAR2(10) PRIMARY KEY,
    userPw     VARCHAR2(10) NOT NULL,
    name       VARCHAR2(20) NOT NULL,
    age        NUMBER(3)
)

-- 자료 삽입
INSERT INTO person(userId, userPw, name, age)
VALUES ('cloud', 'cloud123', '구름이', 120);

-- 커밋 완료
COMMIT;
```

DAO와 VO 정의와 사용법

VO(Value Object)의 정의와 사용법

- 여러 다른 타입의 데이터를 다른 클래스로 전달할 때 사용
- 만드는 방법

> DB 테이블의 필드명을 속성으로 선언한다.
> 생성자를 구현한다.
> 각 속성에 대한 getter/setter 메서드를 구현한다.

DAO(Data Access Object)의 정의와 사용법

- 자바 프로그램에서 데이터베이스 작업만 수행하는 코드
- 하나의 클래스 안에 코드가 많아져서 개발이나 유지관리가 힘들어진다.
- 화면기능, 데이터베이스 연동 기능 등을 각각 담당하는 클래스로 나누어 프로그램을 구현한다.

DAO와 VO 정의와 사용법

```java
package dbconnection.person;

public class Person {
    //필드
    private String userId;
    private String userPw;
    private String name;
    private int age;

    //setter, getter
    public String getUserId() {
        return userId;
    }
    public void setUserId(String userId) {
        this.userId = userId;
    }
    public String getUserPw() {
        return userPw;
    }
    public void setUserPw(String userPw) {
        this.userPw = userPw;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
}
```

JDBCUtil – DB 연결

```java
package dbconnection.common;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class JDBCUtil {
        private static String driverClass = "oracle.jdbc.OracleDriver";
        private static String url = "jdbc:oracle:thin:@localhost:1521:xe";
        private static String username = "system";
        private static String password = "12345";

        //DB 연결 메서드
        public static Connection getConnention() {
            try {
                Class.forName(driverClass);
                return DriverManager.getConnection(url, username, password);
            } catch (Exception e) {
                e.printStackTrace();
            }
            return null;
        }
```

JDBCUtil – 연결 종료

```
//DB 연결 종료 메서드
public static void close(Connection conn, PreparedStatement pstmt) {
    if(pstmt != null) {
        try {
            pstmt.close();
        } catch (Exception e) {
            e.printStackTrace();
        }finally {
            pstmt = null;
        }
    }

    if(conn != null) {
        try {
            conn.close();
        } catch (Exception e) {
            e.printStackTrace();
        }finally {
            conn = null;
        }
    }
}
```

JDBCUtil – 연결 종료

```java
//연결 종료(ResultSet이 있는 경우)
public static void close(Connection conn, PreparedStatement pstmt, ResultSet rs) {
    if(rs != null) {
        try {
            rs.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    if(pstmt != null) {
        try {
            pstmt.close();
        } catch (Exception e) {
            e.printStackTrace();
        }finally {
            pstmt = null;
        }
    }

    if(conn != null) {
        try {
            conn.close();
        } catch (Exception e) {
            e.printStackTrace();
        }finally {
            conn = null;
        }
    }
}
```

PersonDAO – 자료 삽입

```java
public class PersonDAO {
    //JDBC 관련 변수
    private Connection conn = null;
    private PreparedStatement pstmt = null;
    private ResultSet rs = null;

    //자료 삽입
    public void insertPerson(Person person) {
        try {
            conn = JDBCUtil.getConnention();  //DB 연결 메서드 호출
            //SQL - DML 언어, 동적 쿼리 - ?기호에 순서대로 대응
            String sql = "INSERT INTO person (userId, userPw, name, age) VALUES (?, ?, ?, ?)";
            pstmt = conn.prepareStatement(sql);       //예외 처리
            pstmt.setString(1, person.getUserId());
            pstmt.setString(2, person.getUserPw());
            pstmt.setString(3, person.getName());
            pstmt.setInt(4, person.getAge());

            pstmt.executeUpdate();     //db에 저장
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            JDBCUtil.close(conn, pstmt);  //연결 종료 메서드 호출
        }
    }
```

PersonDAO – 자료 목록

```java
// 자료 목록 조회
public ArrayList<Person> getPersonList(){
    ArrayList<Person> personList = new ArrayList<>();
    try {
        conn = JDBCUtil.getConnention();
        String sql = "SELECT * FROM person";
        pstmt = conn.prepareStatement(sql);
        rs = pstmt.executeQuery();      //쿼리 실행
        while(rs.next()) {              //자료가 있다면 계속 반복
            Person person = new Person();
            person.setUserId(rs.getString("userId"));
            person.setUserPw(rs.getString("userPw"));
            person.setName(rs.getString("name"));
            person.setAge(rs.getInt("age"));

            personList.add(person);
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        JDBCUtil.close(conn, pstmt, rs);
    }
    return personList;
}
```

PersonDAO – 자료 수정

```
//자료 수정
public void updatePerson(Person person) {
    try {
        conn = JDBCUtil.getConnention();
        String sql = "UPDATE person SET userPw = ?, name = ?, age = ? WHERE userId = ?";
        pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, person.getUserPw());
        pstmt.setString(2, person.getName());
        pstmt.setInt(3, person.getAge());
        pstmt.setString(4, person.getUserId());
        pstmt.executeUpdate();
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        JDBCUtil.close(conn, pstmt);
    }
}
```

PersonDAO – 자료 삭제

```java
//자료 삭제
public void deletePerson(Person person) {
    try {
        conn = JDBCUtil.getConnention();
        String sql = "DELETE FROM person WHERE userId = ?";
        pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, person.getUserId());
        pstmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        JDBCUtil.close(conn, pstmt);
    }
}
```

PersonDAO – 특정 자료 검색

```
//특정 자료 검색
public Person getPerson(String userId) {
    Person person = new Person();
    try {
        conn = JDBCUtil.getConnention();
        String sql = "SELECT * FROM person WHERE userId = ?";
        pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, userId);
        rs = pstmt.executeQuery();
        if(rs.next()) {    //자료가 있다면
            person.setUserId(rs.getString("userId"));
            person.setUserPw(rs.getString("userPw"));
            person.setName(rs.getString("name"));
            person.setAge(rs.getInt("age"));
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        JDBCUtil.close(conn, pstmt, rs);
    }
    return person;
}
```