



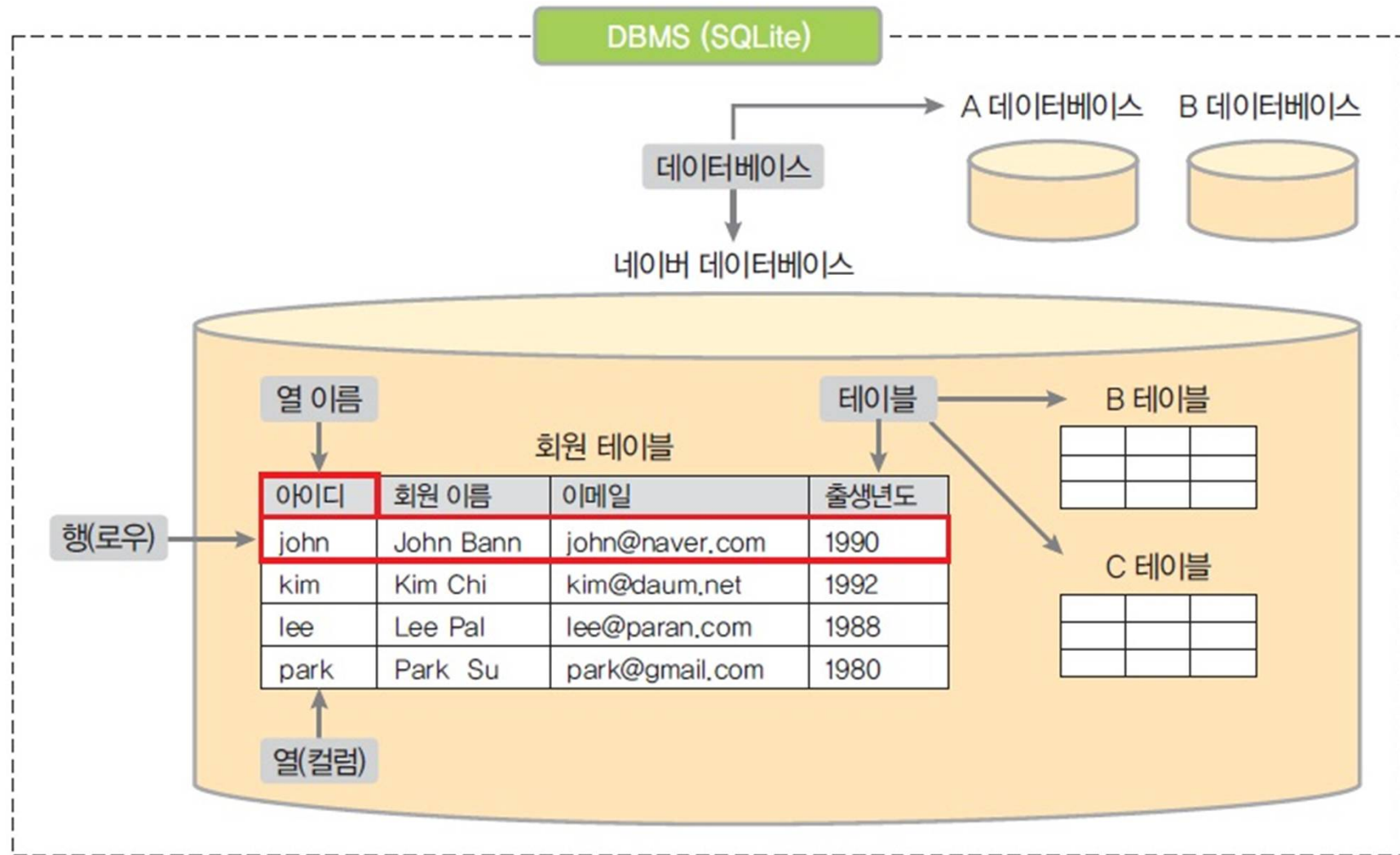
# 데이터베이스

- 용어 정리
  - DBMS란?
  - 데이터베이스란?
  - 테이블이란?
  - 열이란?
  - 행이란?
- SQL = DML(데이터 조작) + DDL(테이블 조작) + DCL(제어)
- DML : CRUD2SG
  - C: INSERT
  - R: SELECT
  - U: UPDATE
  - D: DELETE
  - S: SELECT ~ WHERE
  - S: SELECT ~ ORDER BY
  - G: GROUP BY ~ HAVING

데이터베이스  
를 사용하는  
방법을  
학습합니다.



# 데이터베이스 기본 개념



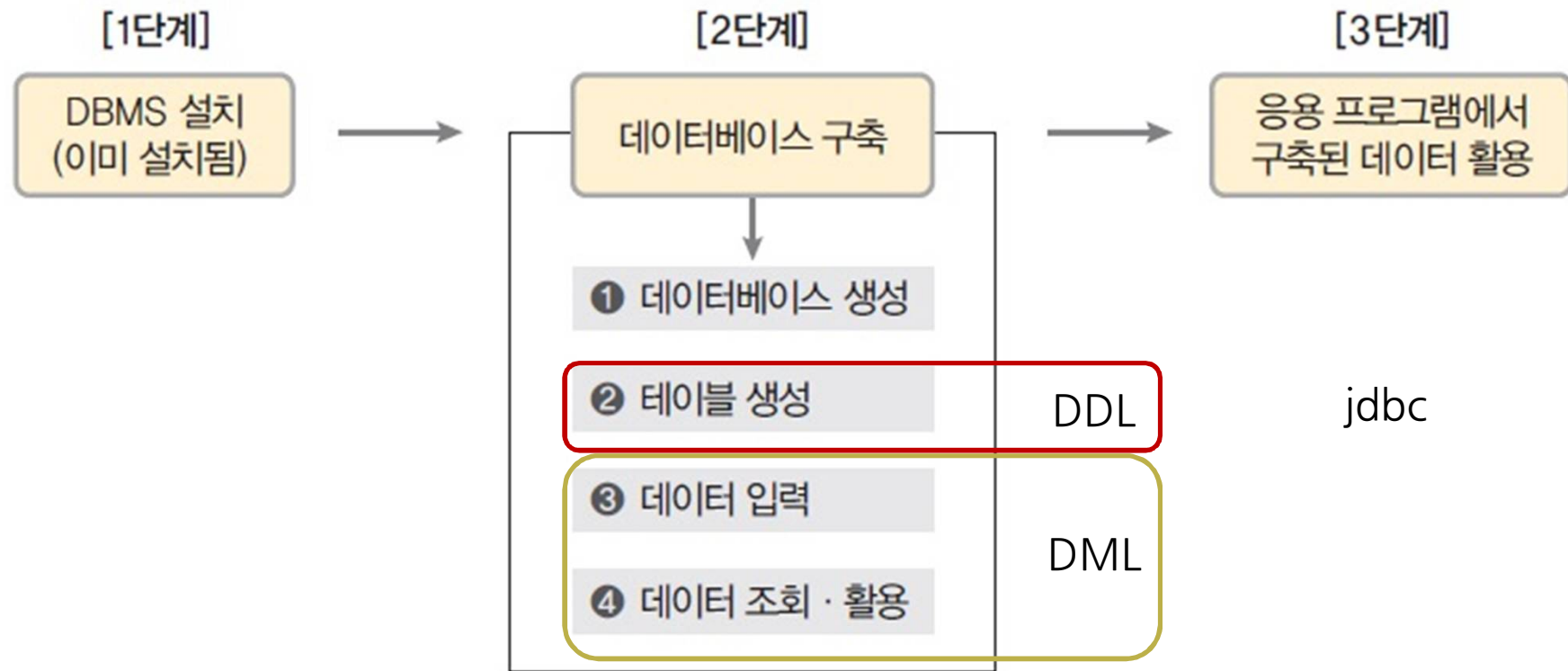


# 데이터베이스 기본 개념

- 데이터베이스 관련 용어
  - **데이터** : 하나하나의 단편적인 정보를 뜻함
  - **DBMS** : 데이터베이스를 관리하는 소프트웨어를 말함  
Oracle, MS-SQL, MySQL, SQLite 소프트웨어가 이에 해당
  - **데이터베이스(DB)** : 테이블이 저장되는 장소로 원통 모양으로 표현  
각 데이터베이스는 서로 다른 고유한 이름이 있어야 함
  - **테이블** : 데이터가 표 형태로 표현된 것
  - **열(컬럼 또는 필드)** : 각 테이블은 1개 이상의 열로 구성됨
  - **열 이름** : 각 열을 구분하는 이름, 열 이름은 각 테이블 안에서는 중복되지 않아야 함
  - **데이터 형식** : 열의 데이터 형식을 뜻함  
**숫자, 문자, 날짜**  
테이블을 생성할 때 열 이름과 함께 지정해줘야 함
  - **행(로우)** : 실제 데이터를 뜻함
  - **SQL** : 사용자와 DBMS와 소통하기 위해서 사용하는 언어  
 $\text{SQL} = \text{DML(CRUD2SG)} + \text{DDL} + \text{DCL}$



# 데이터베이스 구축





# Introduction

- DBMS
  - Database management system
  - Storing and organizing data
- Database
  - Collection of data
- Table
- Column
- Row, Record
- SQL
  - Relational database
  - Structured Query Language
- JDBC
  - Java Database Connectivity
  - JDBC driver



# SQL이란?

- SQL = Structured Query Language
- 데이터베이스에서 사용하기 위하여 설계된 언어
- DDL + DML + DCL
  - DDL : Data Definition Language : DB, 테이블 조작
  - DML : Data Manipulation Language : 테이블에서 데이터 조작
  - DCL : Data Control Language : DB 제어



# DDL이란?

데이터베이스나 테이블을 생성하거나 수정하는 데 사용되는 언어.

- 생성(CREATE),
- 변경(ALTER),
- 제거(DROP)
- 권한 부여(GRANT)
- 권한 박탈(REVOKE)

구분	명령어	설명
데이터 정의 명령어 (Data Definition Language)	CREATE	사용자가 제공하는 컬럼 이름을 가지고 테이블을 생성한다. 사용자는 컬럼의 데이터 타입도 지정하여야 한다. 데이터 타입은 데이터베이스에 따라 달라진다. CREATE TABLE은 보통 DML보다 적게 사용된다. 왜냐하면 이미 테이블이 만들어져 있는 경우가 많기 때문이다.
	ALTER	테이블에서 컬럼을 추가하거나 삭제한다.
	DROP	테이블의 모든 레코드를 제거하고 테이블의 정의 자체를 데이터베이스로부터 삭제하는 명령어이다.
	USE	어떤 데이터베이스를 사용하는지를 지정



# 데이터베이스 접속하기

HS 세션 관리자

세션 이름 ▲  
localhost

설정 고급 통계

네트워크 유형: MySQL (TCP/IP)

호스트명 / IP: localhost

☐ 자격 증명 프롬프트  
☐ Windows 인증 사용

사용자: root

암호: ●●●●

포트: 3306

☐ 압축된 클라이언트/서버 프로토콜

데이터베이스: 세미콜론으로 구분

코멘트:

신규 ▼ 저장 삭제 열기 취소 더 보기 ▼





# 데이터베이스 생성하기

-- 데이터베이스 삭제

```
DROP DATABASE IF EXISTS book_db;
```

-- 데이터베이스 생성

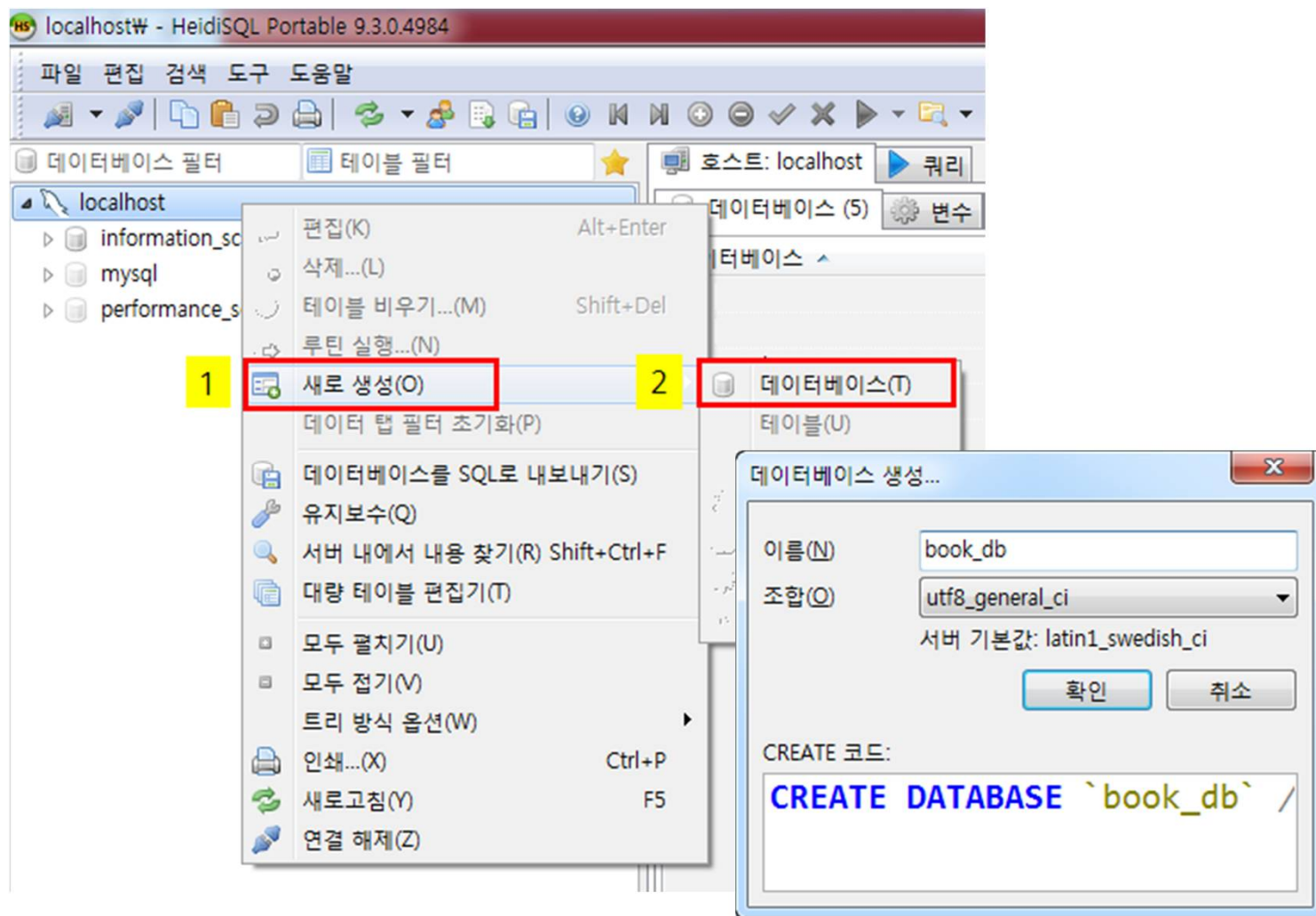
```
CREATE DATABASE book_db COLLATE 'utf8_general_ci';
```

-- 데이터베이스 사용

```
USE book_db;
```



# 데이터베이스 생성하기



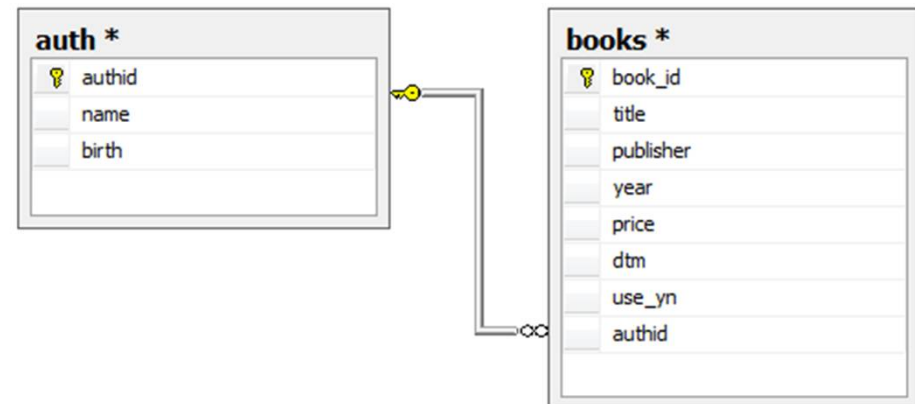


# 테이블 생성하기

-- 테이블 생성

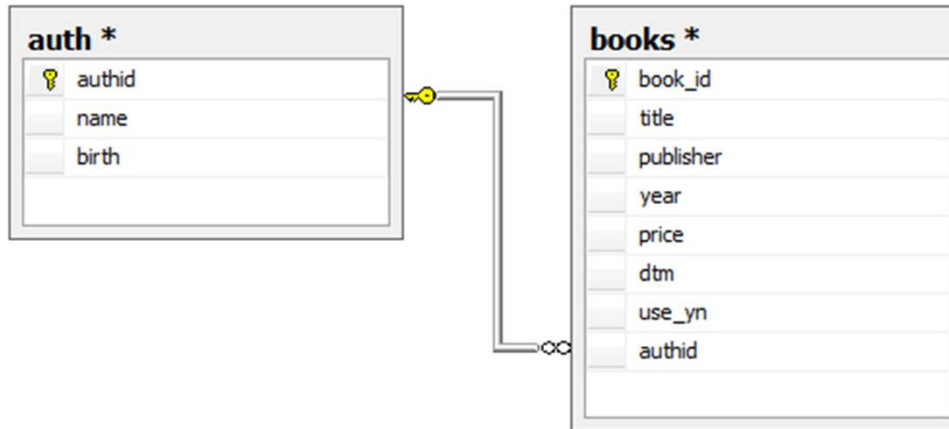
```
DROP TABLE IF EXISTS book;
```

```
CREATE TABLE book (  
    bookid      INT NOT NULL auto_increment  
    , bookname  VARCHAR(50)  
    , publisher VARCHAR(30)  
    , year      VARCHAR(10)  
    , price     INT  
    , dtm       DATE  
    , use_yn    BIT  
    , authid    INT NOT NULL  
  
    , PRIMARY KEY(bookid)  
)  
COLLATE='utf8_general_ci'  
ENGINE=InnoDB ;
```





# 테이블 생성하기



-- 테이블 생성

```
DROP TABLE IF EXISTS auth;
```

```
CREATE TABLE auth (
    authid      INT NOT NULL auto_increment
    , name      VARCHAR(50)
    , birth     VARCHAR(10)

    , PRIMARY KEY(authid)
)
COLLATE='utf8_general_ci'
ENGINE=InnoDB ;
```



# 테이블 생성하기

1

2

테이블: [Untitled]

1

2

추가

#	이름	데이터 유형	길이/설정	부호 없음	NULL 허용	0으로 채움	기본값
---	----	--------	-------	-------	---------	--------	-----

The image shows a screenshot of a database management tool interface. The top part displays a tree view with 'book\_db', 'mysql', and 'performance' folders. A context menu is open for 'book\_db', showing options like '편집(K)', '삭제...(L)', '테이블 비우기...(M)', '루틴 실행...(N)', '새로 생성(O)', '데이터 탭 필터 초기화(P)', '데이터베이스를 SQL로 내보내기(S)', '유지보수(Q)', '서버 내에서 내용 찾기(R)', and '대량 테이블 편집기(T)'. The '새로 생성(O)' option is highlighted with a red box and a yellow '1'. A sub-menu is open for '새로 생성(O)', showing options like '데이터베이스(T)', '테이블(U)', '테이블 복사(V)', '뷰(W)', '저장 루틴(X)', '트리거(Y)', and '이벤트(Z)'. The '테이블(U)' option is highlighted with a red box and a yellow '2'. Below the menu, the '테이블: [Untitled]' tab is selected. The main window shows the '기본' (Basic) tab for creating a table. The '이름:' (Name) field is highlighted with a red box and a yellow '1', with the text '테이블 이름 입력' (Enter table name) inside. The '열:' (Columns) section is highlighted with a yellow '2', and the '추가' (Add) button is highlighted with a red box. Below the '열:' section, there is a table with columns: '#', '이름', '데이터 유형', '길이/설정', '부호 없음', 'NULL 허용', '0으로 채움', and '기본값'.



# 테이블 Primary Key 생성하기

기본 옵션 인덱스 외래 키 분할 CREATE 코드 ALTER 코드

이름: tb\_saved

열: + 추가 - 제거 ▲ 위로 ▼ 아래로

#	이름	데이터 유형	길이/설정	부호 없음	NULL 허용	0으로 채움	기본값
1	saved_id			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREMENT
2	name			<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
3	expression			<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL

Context menu for row 1:

- 복사(C) Ctrl+C
- 선택한 열 복사(S)
- 열 붙여넣기(T)
- 열 추가(U) Ctrl+Ins
- 열 제거(V) Ctrl+Del
- 위로(W) Ctrl+U
- 아래로(X) Ctrl+D
- 새 인덱스 생성(Y)
  - PRIMARY
  - KEY
  - UNIQUE
  - FULLTEXT
  - SPATIAL
- 인덱스에 추가(Z)



# DML이란?

데이터를 실질적으로 처리하는 데 사용되는 언어입니다. (SUDI)

- SELECT(검색)
- UPDATE(수정)
- INSERT(삽입)
- DELETE(삭제)
  
- COMMIT(트랜잭션)
- ROLLBACK(트랜잭션)

구분	명령어	설명
데이터 조작 명령어 (Data Manipulation Language)	SELECT	데이터베이스로부터 데이터를 쿼리하고 출력한다. SELECT 명령어들은 결과 집합에 포함시킬 컬럼을 지정한다. SQL 명령어 중에서 가장 자주 사용된다.
	INSERT	새로운 레코드를 테이블에 추가한다. INSERT는 새롭게 생성된 테이블을 채우거나 새로운 레코드를 이미 존재하는 테이블에 추가할 때 사용된다.
	DELETE	지정된 레코드를 테이블로부터 삭제한다.
	UPDATE	테이블에서 레코드에 존재하는 값을 변경한다.



# 레코드 검색하기

```
SELECT * FROM book;
```

```
SELECT * FROM book WHERE bookname like '%SQL%';
```

```
SELECT * FROM book WHERE bookname = 'JAVA';
```

```
SELECT * FROM book WHERE 30700 <= price and price <50000;
```

```
SELECT * FROM book WHERE price BETWEEN 30700 AND 50000;
```

```
SELECT * FROM book WHERE price <= 30700 OR 58000 < price ;
```

```
SELECT bookname, publisher, price, authid FROM book;
```





## 레코드 정렬하기

```
SELECT * FROM book ORDER BY price ASC;
```

```
SELECT * FROM book ORDER BY price DESC;
```

```
SELECT * FROM book ORDER BY publisher ASC, price DESC;
```



## 레코드 추가하기

```
INSERT 테이블명( 컬럼명1, 컬럼명2, 컬럼명3, .. , 컬럼  
명N )
```

```
VALUES( 값1      ,      값2,      값3, .. , 값
```

```
INSERT INTO book (bookname      , publisher, year      , price, authid )  
VALUES('System', 'Wiley'      , '2003', 30700, 1      ) ;
```

```
INSERT INTO auth (authid, name      , birth      )  
VALUES(1      , 'bob'      , '1970.05.01' ) ;
```



# 레코드 수정하기

```
/* UPDATE 테이블명  
    SET 컬럼명1=값1  
    , ..  
    , 컬럼명N=값N  
    WHERE 컬럼명=값 ; */
```

```
UPDATE book  
    SET year = '2016'  
    WHERE bookname like '%SQL%' ;
```

```
UPDATE book  
    SET year = '2010'  
    , price = 20000  
    WHERE bookname = 'JAVA' ;
```



# 레코드 삭제하기

```
/* DELETE FROM 테이블명  
    WHERE 컬럼명=값 ; */
```

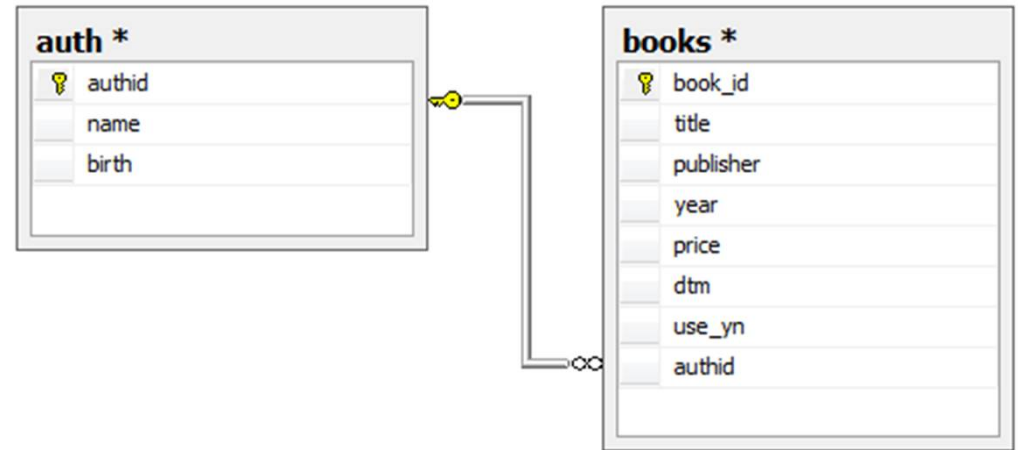
```
DELETE FROM book  
    WHERE bookname = 'JAVA' ;
```

```
DELETE FROM book  
    WHERE publisher ='Wiley' AND year < '2015' ;
```



# 레코드 조인

- 조인이란?
  - 테이블 연결



- 조인의 종류
  - *inner join*
  - *left join = left outer join*
  - *right join = right outer join*
  - *cross join*

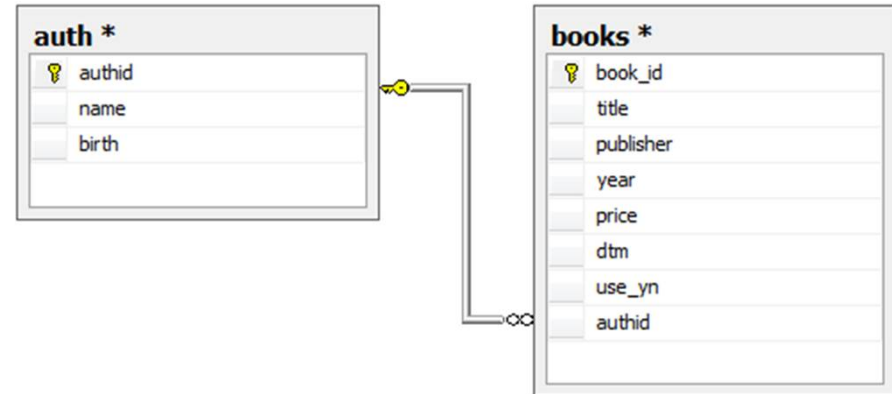


# 조인하기

*-- inner join*

```
select book.*, auth.*  
from book
```

```
inner join auth on book.authid = auth.authid ;
```



*-- left join*

```
select book.*, auth.*  
from book
```

```
left join auth on book.authid = auth.authid ;
```



# JDBC 프로그래밍

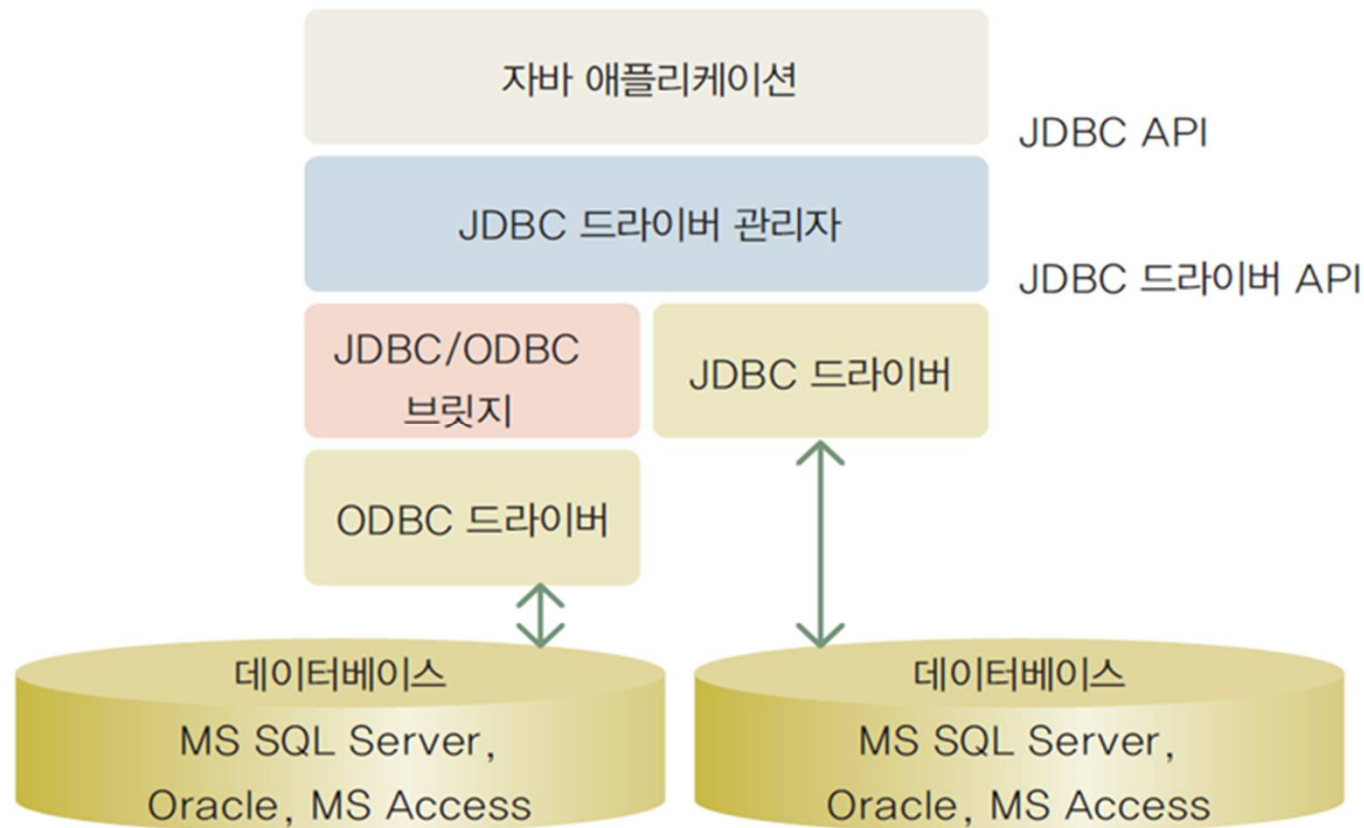
- 자바와 데이터베이스
- JDBC란
  - 자바에서 DB 프로그래밍을 위해 사용되는 라이브러리
- JDBC 프로그래밍 절차
  - JDBC 드라이버 로딩
  - 데이터베이스 커넥션 연결
  - PreparedStatement 객체 생성
  - 쿼리 실행 쿼리 실행 결과 사용
  - PreparedStatement 종료
  - 데이터베이스 커넥션 종료

자바를 통하여  
데이터베이스  
를 사용하는  
방법을  
학습합니다.



# 자바와 데이터베이스

- JDBC(Java Database Connectivity)는 자바 API의 하나로서 데이터베이스에 연결하여서 데이터베이스 안의 데이터에 대하여 검색하고 데이터를 변경할 수 있게 한다.







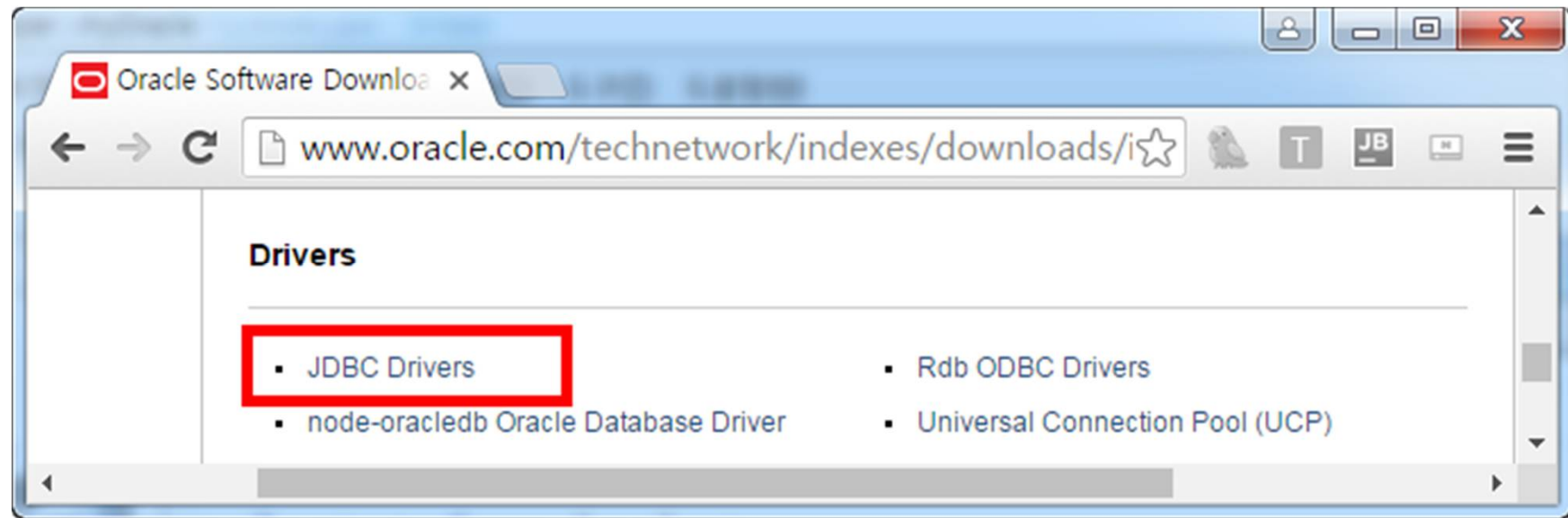
# 프로젝트에 JDBC 드라이버 추가

- 프로젝트에 JDBC 드라이버 추가
- jar 파일로 추가하는 방법
  - 비추
- build.gradle에 추가하는 방법
  - 추천



# Oracle JDBC 드라이버 다운로드

- Oracle JDBC 드라이버 jar 파일 다운로드
- 1. <https://www.oracle.com> 에서 드라이버 jar 파일을 다운로드 받는다. 단, **jar 파일 압축을 풀으면 안됩니다.**




## Oracle Database 12.2.0.1 JDBC Driver & UCP Downloads

Thank you for accepting the OTN License Agreement; you may now download this software.

### Oracle Database 12c Release 2 (12.2.0.1) JDBC Driver & UCP Downloads

JDBC Thin driver from the Oracle Database 12c Release 2

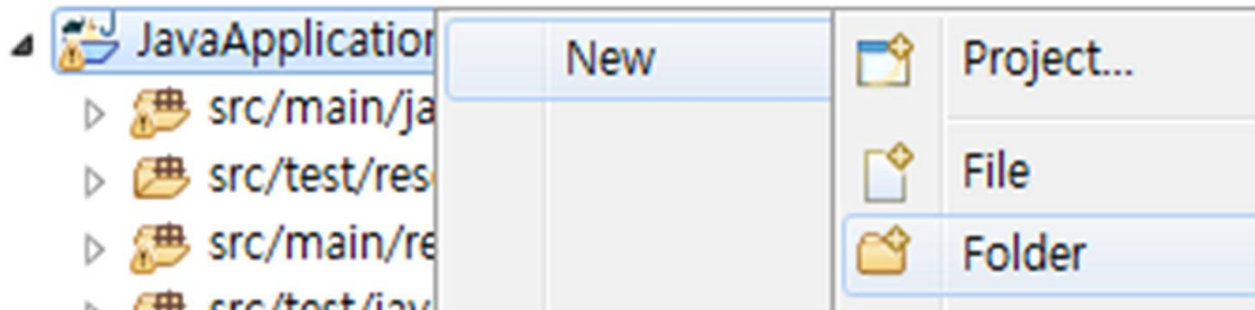
 [ojdbc8.jar](#) (4,036,257 bytes) - (SHA1 Checksum: 60f439fd01536508df32658d0a416c49ac6f07fb)

Certified with JDK 8; It contains the JDBC driver classes except classes for NLS support in Oracle Object and Collector types.

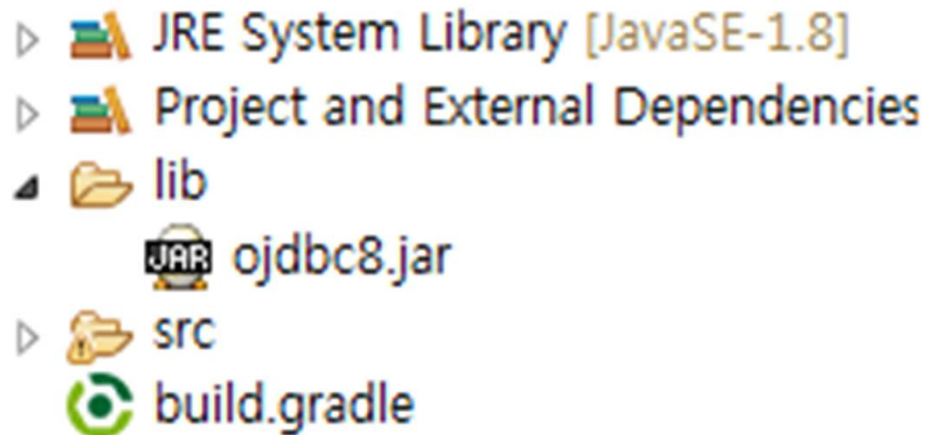


# 프로젝트에 Oracle JDBC 드라이버 추가-jar 파일로.

- 프로젝트에 lib 폴더 생성



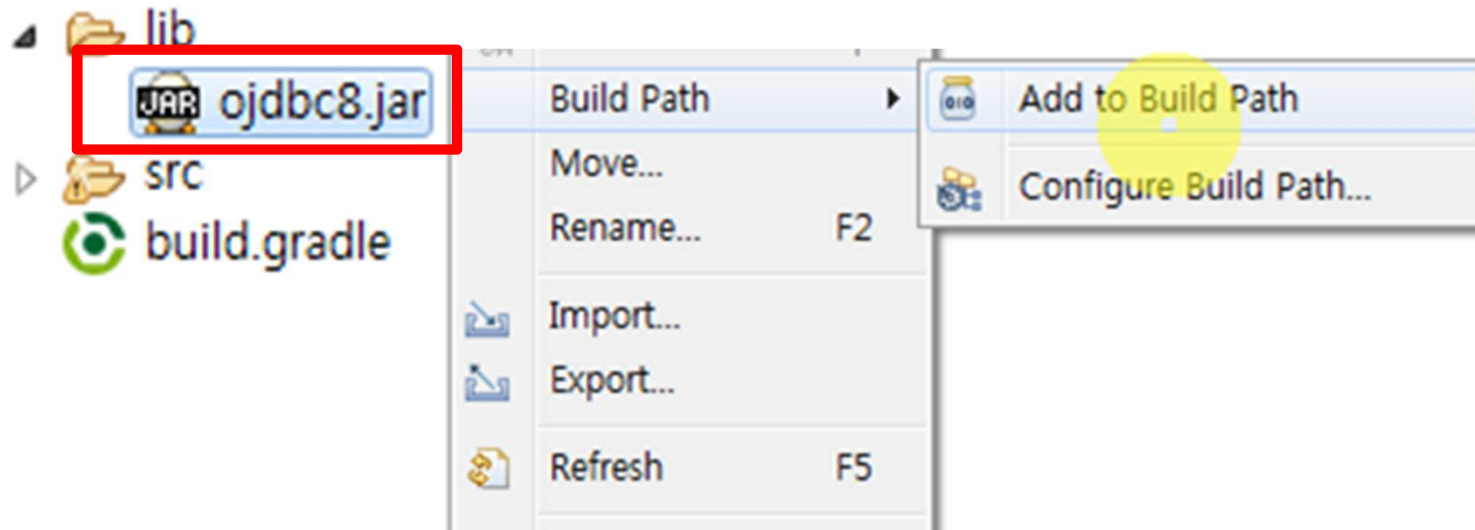
- lib 폴더에 다운로드 받은 Oracle 드라이버 jar 파일을 복사



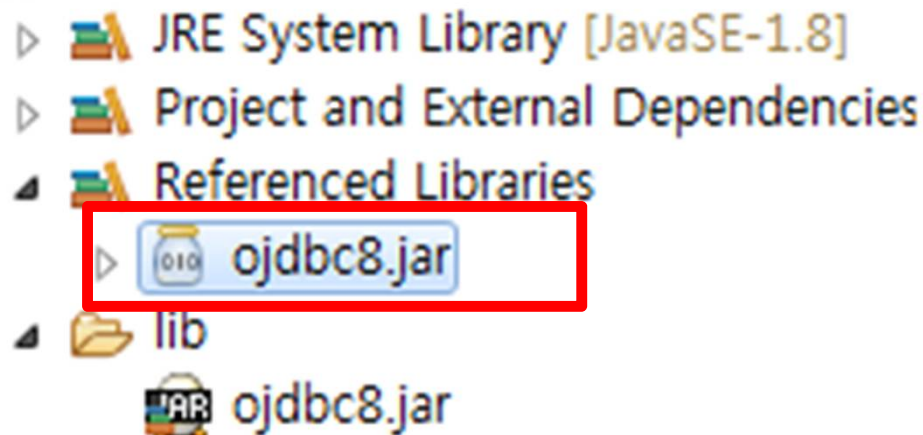


# 프로젝트에 Oracle JDBC 드라이버 추가-jar 파일로.

- jar 파일 Java Build Path에 추가



- jar 파일 Java Build Path에 추가 후





# 프로젝트에 MySQL 드라이버 추가

- <http://mvnrepository.com/>에서 mysql 드라이버를 검색한다.

The screenshot shows the Maven Repository website for the MySQL Connector/J 5.1.38 artifact. The page layout includes a top navigation bar with the Maven logo and a search bar. The left sidebar lists various categories such as Aspect Oriented, Actor Frameworks, and Application Metrics. The main content area displays the artifact details for MySQL Connector/J 5.1.38, including a note about a new version (8.0.8-dmr), the license (GPL 2.0), categories (MySQL Drivers), organization (Oracle Corporation), homepage, date, files, repositories, and the number of artifacts used by (2,124). A code snippet at the bottom shows how to add the driver to a project using Maven.

Indexed Artifacts (8.17M)

8163k  
4081k  
0

2004 2017

Popular Categories

- Aspect Oriented
- Actor Frameworks
- Application Metrics
- Build Tools
- Bytecode Libraries
- Command Line Parsers
- Cache Implementations
- Cloud Computing
- Code Analyzers
- Collections
- Configuration Libraries
- Core Utilities
- Date and Time Utilities

Home » mysql » mysql-connector-java » 5.1.38

Note: There is a new version for this artifact

New Version 8.0.8-dmr

MySQL Connector/J » 5.1.38

MySQL JDBC Type 4 driver

License	GPL 2.0
Categories	MySQL Drivers
Organization	Oracle Corporation
HomePage	<a href="http://dev.mysql.com/doc/connector-j/en/">http://dev.mysql.com/doc/connector-j/en/</a>
Date	(Dec 02, 2015)
Files	<a href="#">pom (1 KB)</a> <a href="#">jar (960 KB)</a> <a href="#">View All</a>
Repositories	<a href="#">Central</a> <a href="#">Aspose</a>
Used By	2,124 artifacts

Maven Gradle SBT Ivy Grape Leiningen Buildr

```
// https://mvnrepository.com/artifact/mysql/mysql-connector-java
compile group: 'mysql', name: 'mysql-connector-java', version: '5.1.38'
```

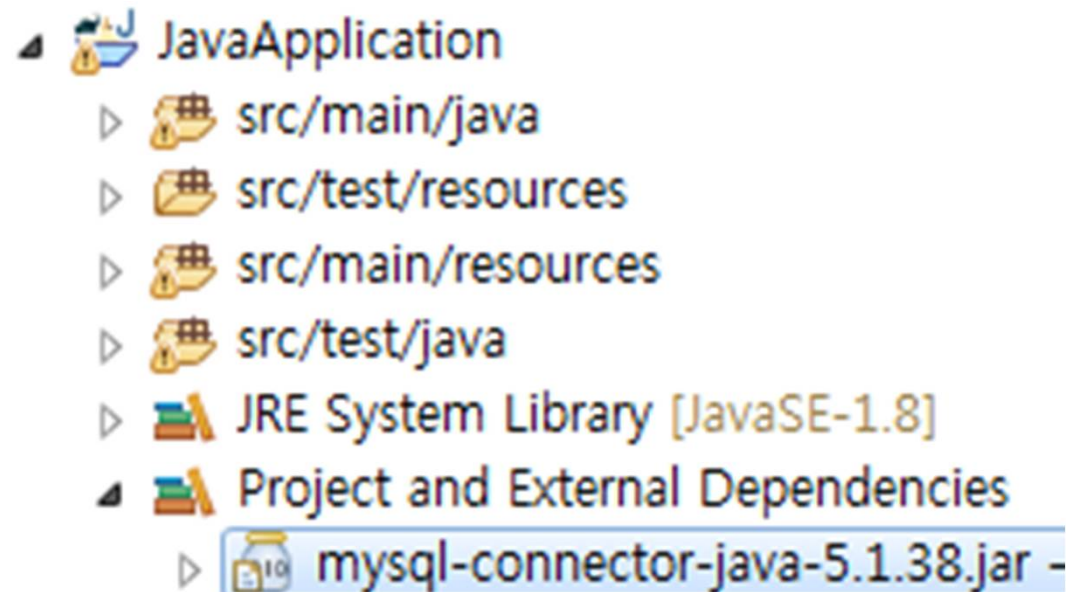
mvnrepository.com/artifact/mysql/mysql-connector-java/5.1.38#leini...



# 프로젝트에 MySQL 드라이버 추가

- build.gradle에 MySQL 드라이버 추가

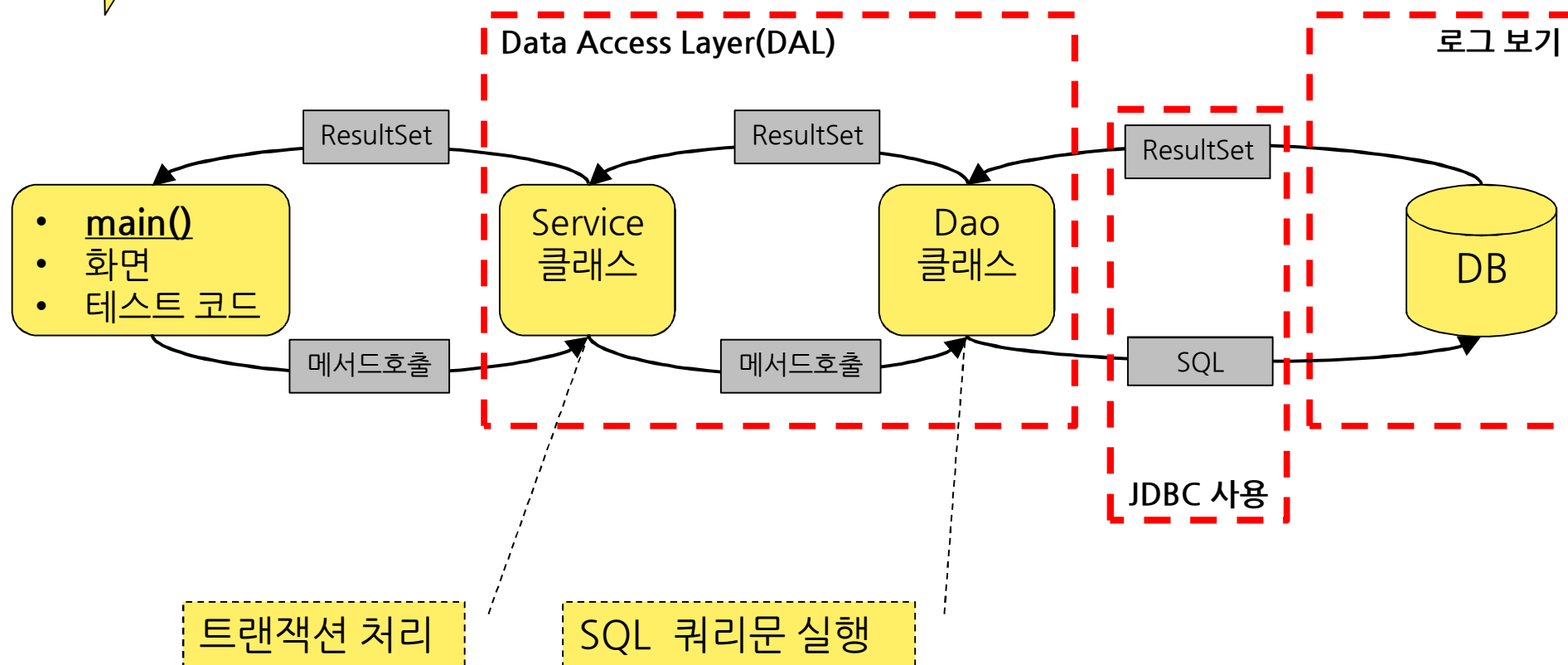
```
// 의존성 설정
dependencies {
    // mysql 드라이버 (라이브러리)
    compile "mysql:mysql-connector-java:5.1.38"
}
```





# JDBC를 이용한 데이터 처리 과정

암기

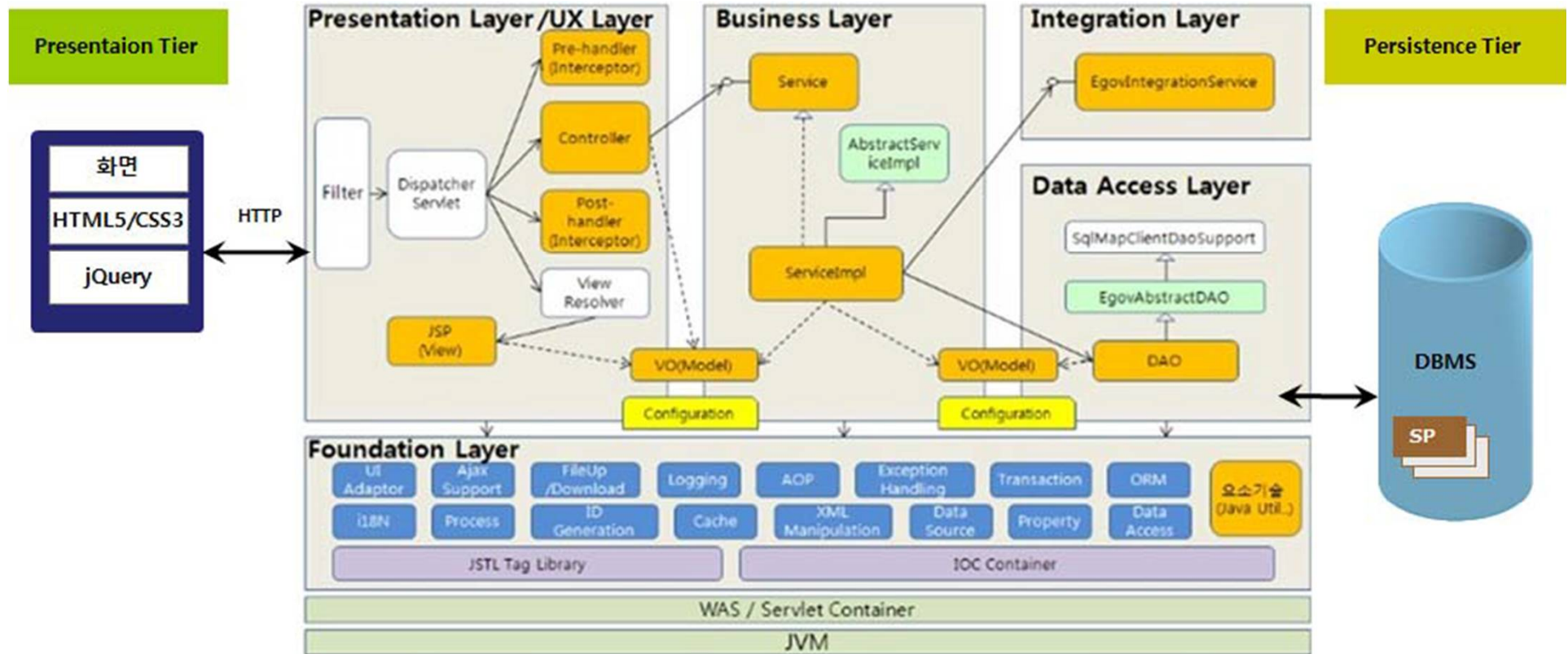






# 전자 정부 프레임워크

- 전자정부 모바일 공통 컴포넌트 적용 아키텍처 예시



전자정부 프레임워크 v3.0





# JDBC 프로그래밍 방식

- Statement 방식
  - *SQL 인젝션 공격에 취약*
  - *느리다*
- PreparedStatement 방식
  - *SQL 인젝션 공격을 무력화*
  - *빠르다*



# JDBC 프로그래밍 방식

Application	DB I/O		DBMS						
GUI화면	Service Layer	DAO(Data Access Object) Layer	TABLE						
		<table><tr><td>SELECT</td><td>INSERT DELETE UPDATE</td></tr><tr><td>executeQuery()</td><td>executeUpdate()</td></tr><tr><td>ResultSet</td><td>int</td></tr></table>	SELECT	INSERT DELETE UPDATE	executeQuery()	executeUpdate()	ResultSet	int	
SELECT	INSERT DELETE UPDATE								
executeQuery()	executeUpdate()								
ResultSet	int								



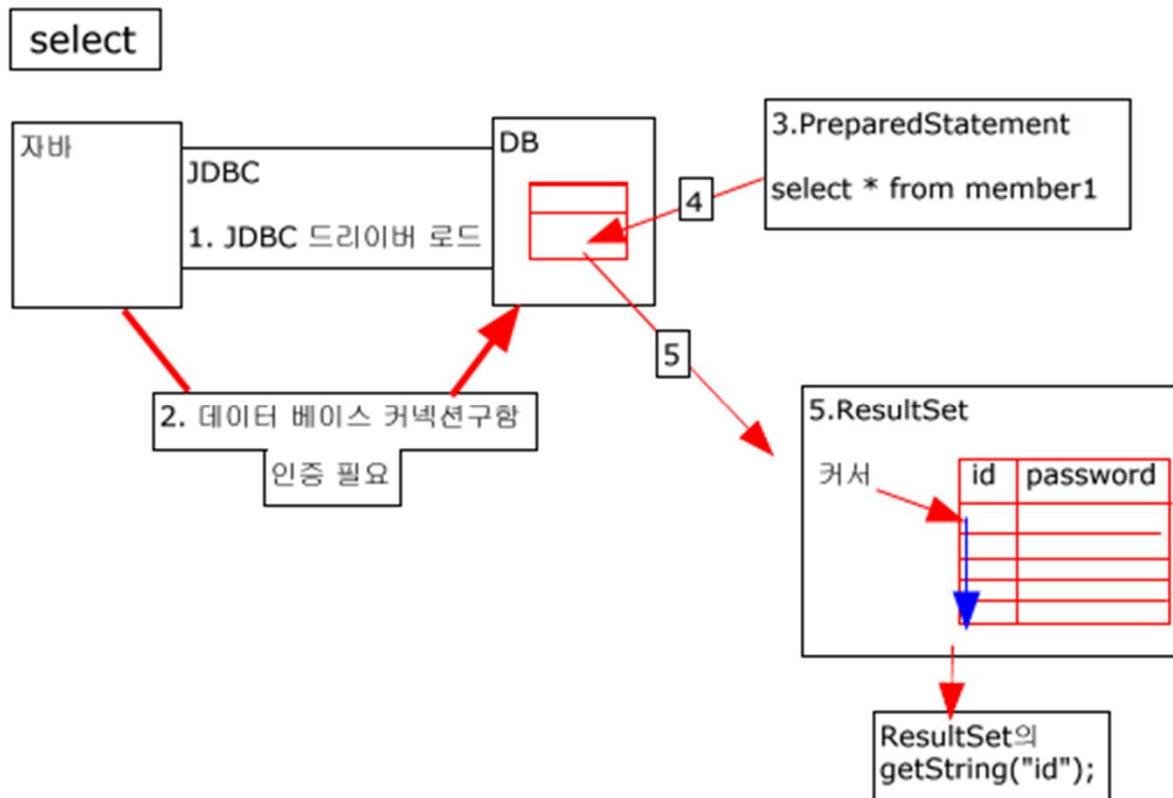
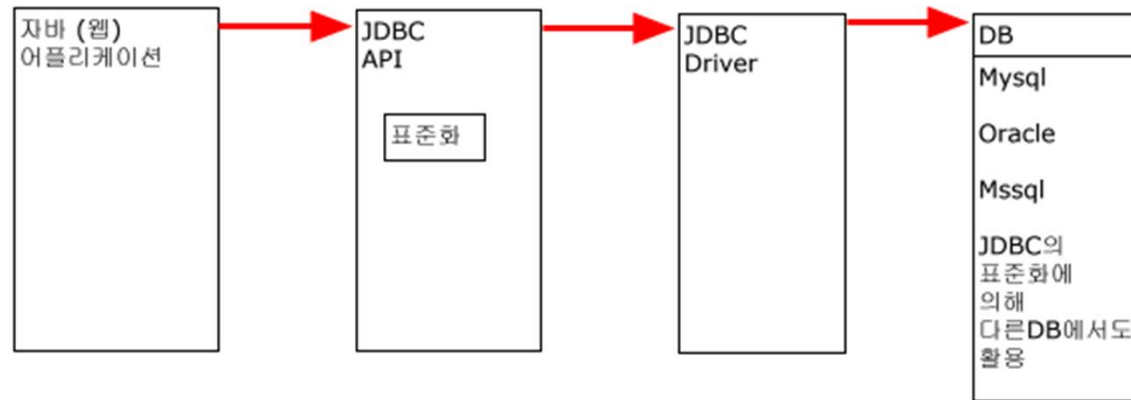
## JDBC프로그램 실행 순서(1/3)

암기

1. JDBC 드라이버 로딩
2. 데이터베이스 커넥션 연결(`java.sql.Connection`)
3. `PreparedStatement` 객체 생성
4. SQL 쿼리 실행
  - `PreparedStatement.executeQuery()` :  
Select
  - `PreparedStatement.executeUpdate()` :  
insert / update / delete
5. 쿼리 결과( `ResultSet` / int )
6. `PreparedStatement` 종료
7. 데이터베이스 커넥션 종료(`java.sql.Connection`)

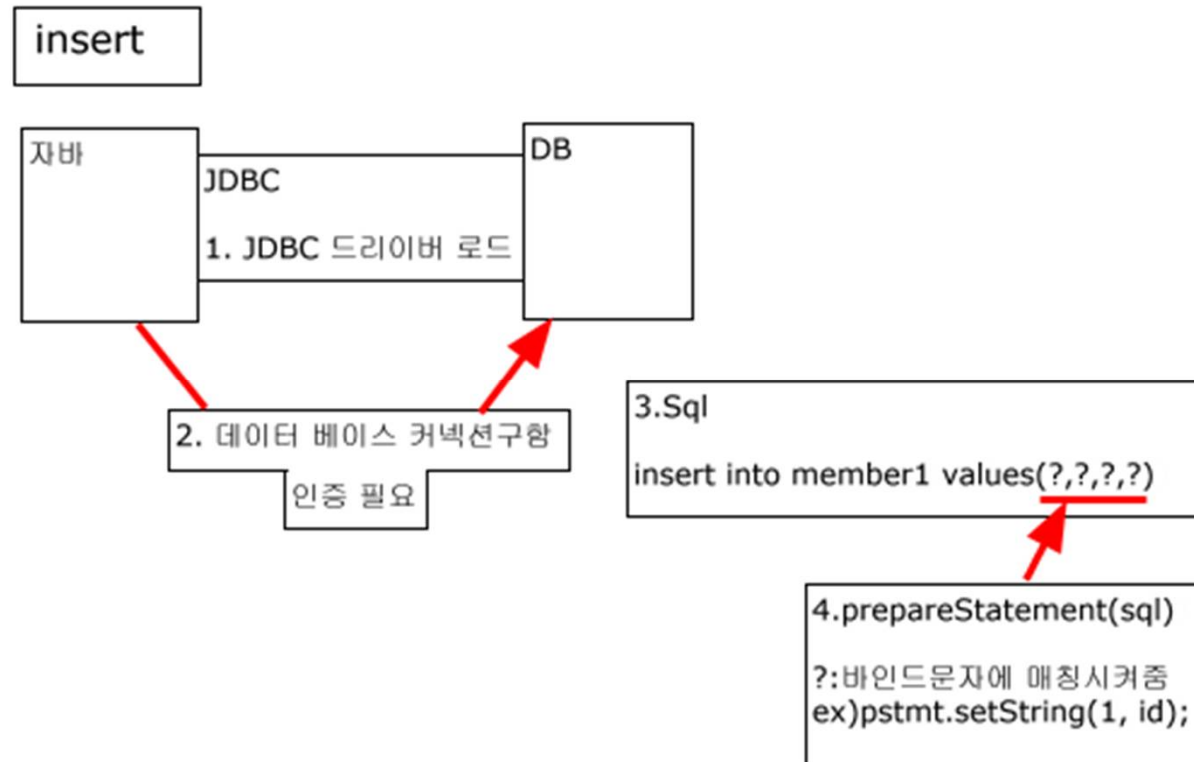
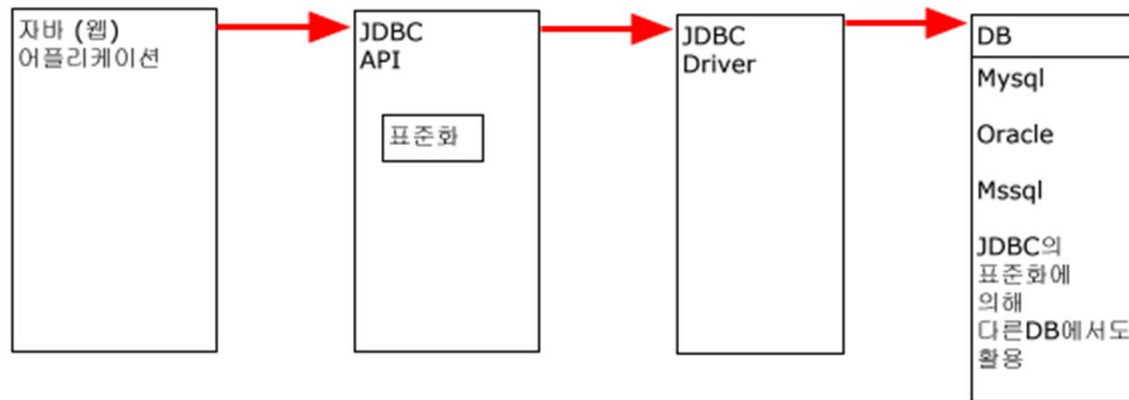


## JDBC프로그램 실행 순서(2/3)





# JDBC프로그램 실행 순서(3/3)





# 데이터베이스 프로그래밍의 순서

## ① JDBC 드라이버를 적재

## ② 데이터베이스 연결

```
Class.forName("      ?      ");
```

```
Connection conn =  
DriverManager.getConnection(url, user, pw);
```



## ③ SQL 문장 작성 및 전송

```
String query = "SELECT * FROM book ";  
PreparedStatement stmt =  
conn.prepareStatement(query);  
ResultSet rs = stmt.executeQuery();
```



## ④ 결과 집합 사용 후 연결 해제

```
while( rs.next ) {  
    int bookid      = rs.getInt( "bookid" );  
    String bookname = rs.getString( "bookname" );  
}
```





# 타입 매핑

DB 타입(sqlite, mysql, oracle)	jdbc 함수	java
BIT(1)	getBoolean() setBoolean()	Boolean
BIT(1), TINYINT, SMALLINT, INT, BIGINT	getInt() setInt()	Integer
CHAR , VARCHAR TEXT	getString() setString()	String
DATE DATETIME TIME TIMESTAMP	getDate() setDate()	<a href="#">java.util.Date</a>
FLOAT, DOUBLE	getDouble() setDouble()	Double
NUMERIC, DECIMAL		<a href="#">java.math.BigDecimal</a> <a href="#">java.math.BigInteger</a>
BINARY, VARBINARY, BLOB		byte[]



# DBMS별 타입 매핑

MySQL DBMS	Oracle DBMS	SQLite DBMS
INT	NUMBER( 자리수, 0 )	INTEGER
INTEGER		
TINYINT		
SMALLINT		
MEDIUMINT		
BIGINT		
NUMERIC		
BOOLEAN		
CHAR	VARCHAR2	TEXT
VARCHAR		
NCHAR		
NVARCHAR2		
TEXT		
DATE	DATE	TEXT
DATETIME		
BLOB	BLOB	NONE
CLOB	CLOB	
REAL	NUMBER( 자리수, 소수점 )	REAL
DOUBLE		
FLOAT		
DECIMAL(10,5)		





# ResultSet 사용법

메서드명	설명
rs.next()	커서 위치를 현재 row에서 다음 row 로 이동.
rs.first()	커서 위치를 첫번째 row로 이동.
rs.last()	커서 위치를 마지막 row로 이동.
rs.getRow()	현재 커서의 위치 번호를 가져온다.
rs.getInt("컬럼명")	커서가 위치한 row에서 "컬럼명"에 해당되는 값을 가져온다.
rs.getDouble("컬럼명")	커서가 위치한 row에서 "컬럼명"에 해당되는 값을 가져온다.
rs.getString("컬럼명")	커서가 위치한 row에서 "컬럼명"에 해당되는 값을 가져온다.
rs.getBoolean("컬럼명")	커서가 위치한 row에서 "컬럼명"에 해당되는 값을 가져온다.
rs.getDate("컬럼명")	커서가 위치한 row에서 "컬럼명"에 해당되는 값을 가져온다.

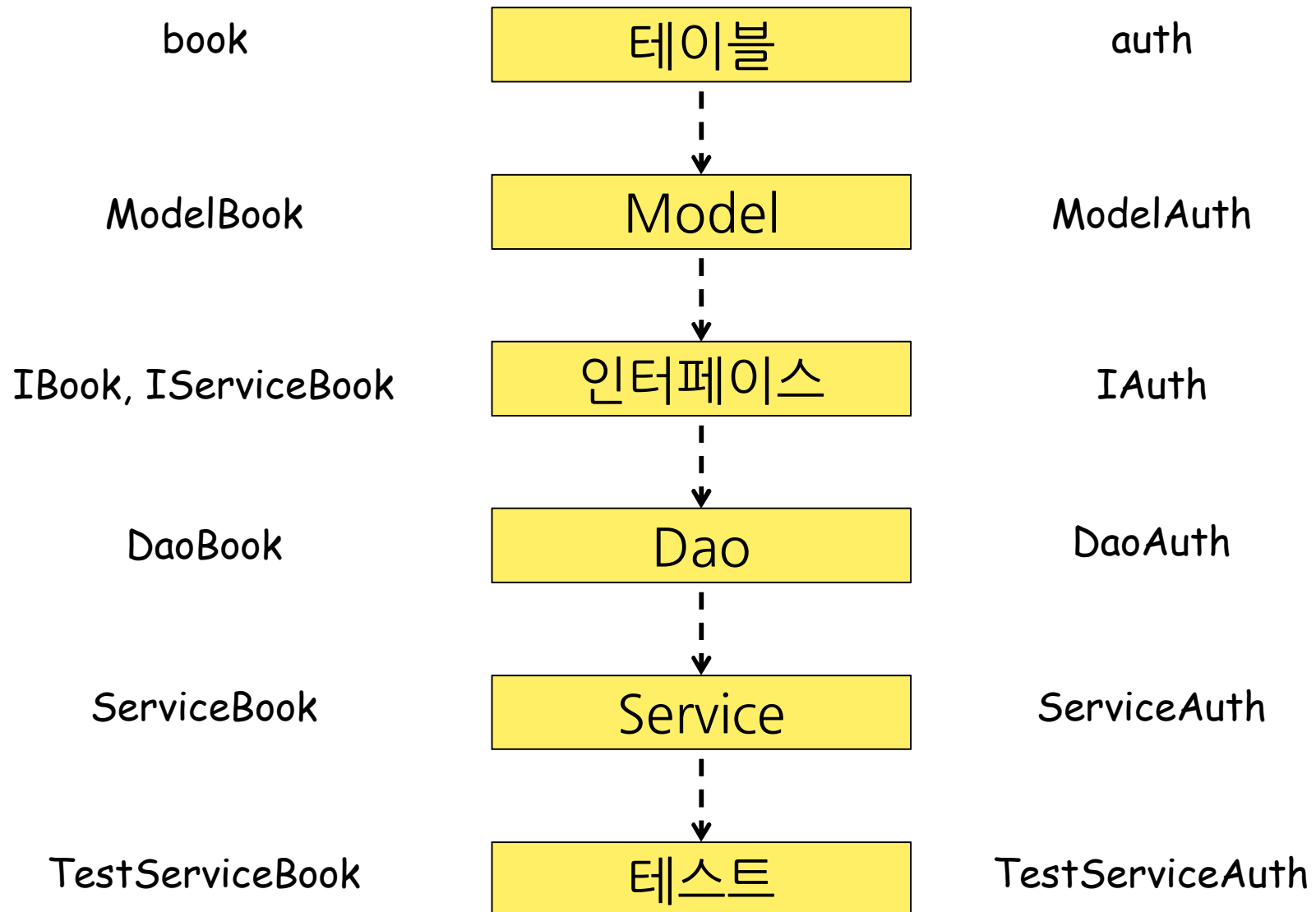
커서 위치



authid	name	birth
1	bob	1970.05.01
2	kim	1980.05.01
7	park	2000.05.01



# DB Layer 만들기





## Book 관련 작업

- DBConnect 클래스작성
  - `java.jdbc.test.DBConnectTestJUnit` 클래스 만들기
- ModelBook 클래스 만들기
- IBook 인터페이스 만들기
- IServiceBook 인터페이스 만들기
- DaoBook 클래스 만들기
  - `java.jdbc.test. TestDaoBook` 클래스 만들기
- ServiceBook 클래스 만들기
  - `java.jdbc.test.TestServiceBook` 클래스만들기



# DBConnect 만들기 - MySQL용

```
public class DBConnect {  
    public static java.sql.Connection connectionMySQL(){  
        String url      = "jdbc:mysql://localhost:3306/book_db";  
        String user      = "root";  
        String password = "1234";  
        java.sql.Connection conn = null;  
        try {  
            Class.forName("com.mysql.jdbc.Driver"); // mysql driver 로딩  
            conn = java.sql.DriverManager.getConnection(url, user, password); //DB연결  
        } catch (ClassNotFoundException e) {  
            e.printStackTrace();  
        } catch (java.sql.SQLException e) {  
            e.printStackTrace();  
        }  
        return conn;  
    }  
    public static java.sql.Connection makeConnection(){  
        return connectionMySQL(); // MySQL과 연결할 때  
    }  
}
```

localhost: mysql 서버 주소  
3306: MySQL 포트 번호  
book\_db: DB 명



# TestDBConnect 만들기

DBConnect 클래스의 메서드를  
테스트하기 위한 JUnit 클래스

```
public class TestDBConnect {
    @Test
    public void test_connectionMySQL() throws Exception {
        java.sql.Connection conn = DBConnect.connectionMySQL();

        if( conn != null) {
            assertTrue("db connect success", true);
        }
        else {
            assertTrue("db connect fail", false);
        }
    }

    @Test
    public void test_makeConnection() throws Exception {
        java.sql.Connection conn = DBConnect.makeConnection();

        if( conn != null) {
            assertTrue("db connect success", true);
        }
        else {
            assertTrue("db connect fail", false);
        }
    }
}
```



# ModelBook 만들기

book Table에 매칭되는  
자바 클래스 만들기

```
public class ModelBook {  
    private Integer bookid    = null;  
    private String  bookname  = "" ;  
    private String  publisher = "" ;  
    private String  year      = "" ;  
    private Integer price     = null;  
    private Date    dtm       = null;  
    private Boolean use_yn     = null;  
    private Integer authid    = null;  
  
    // getter & setter 만들기  
  
    // 디폴트 생성자(constructor) 만들기  
  
    // toString() 만들기  
  
}
```



실습

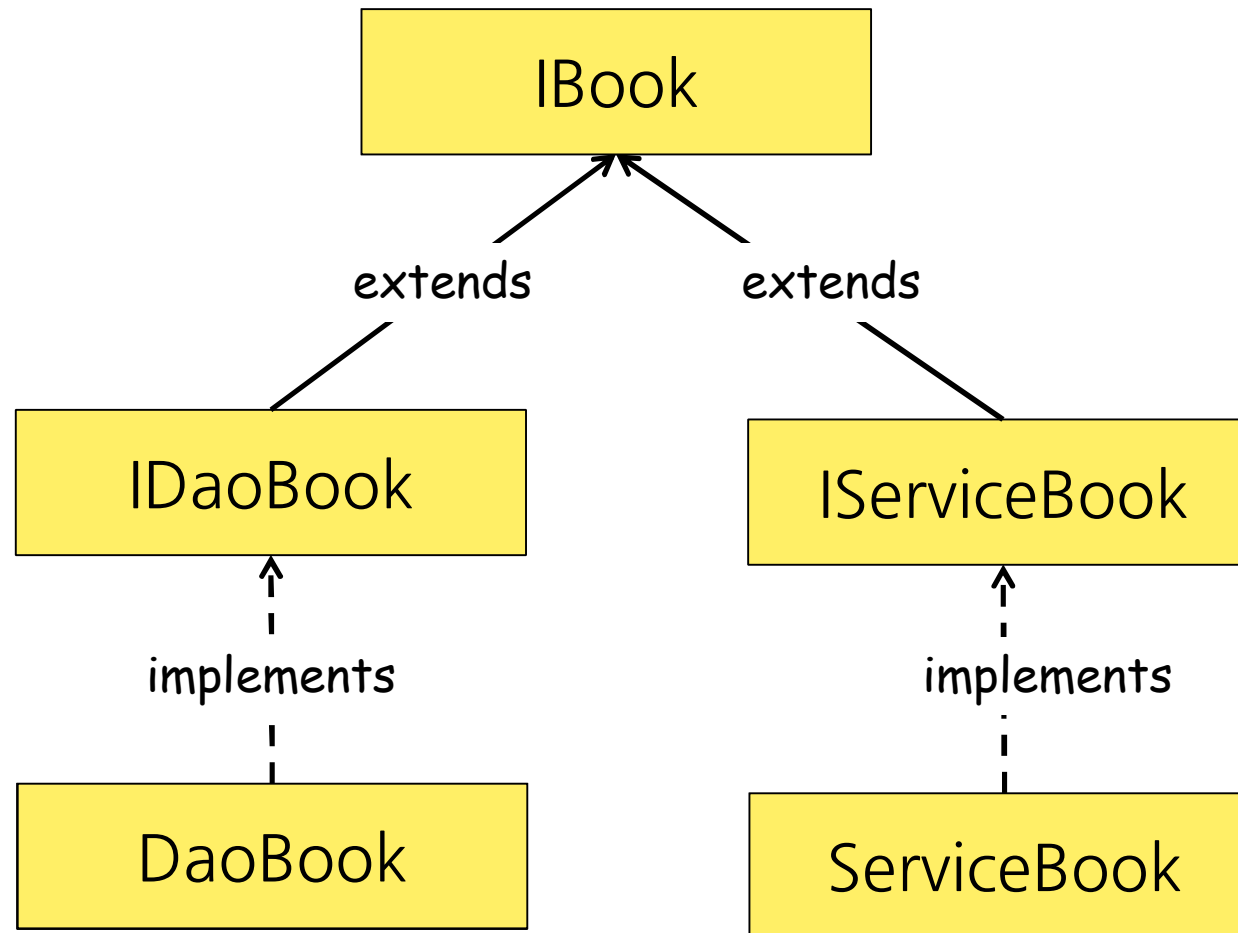
## ModelAuth 만들기

auth Table에 매칭되는  
자바 클래스 만들기

```
public class ModelAuth {  
  
    // 필드 선언  
  
    // getter & setter 만들기  
  
    // toString() 만들기  
  
    // 디폴트 생성자(constructor) 만들기  
  
}
```



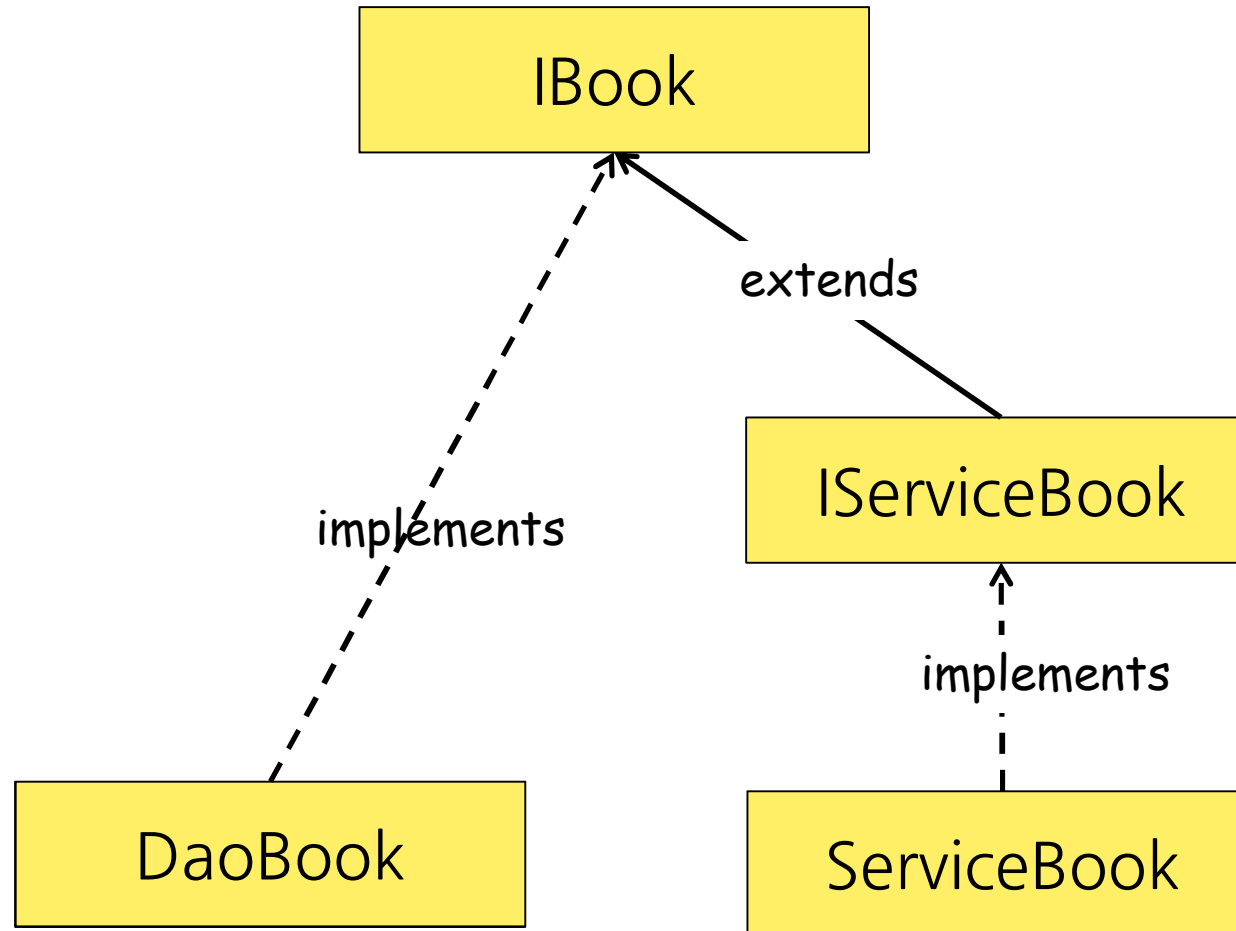
# Book 인터페이스 구조







# Book 인터페이스 구조





# IBook 만들기

```
import java.sql.*;

public interface IBook {

    int getCount(ModelBook book) throws SQLException;

    int getMaxBookid() throws SQLException;

    ResultSet selectAll() throws SQLException;

    ResultSet selectLike(ModelBook book) throws SQLException;

    ResultSet selectEqual(ModelBook book) throws SQLException;

    ResultSet selectDynamic(ModelBook book) throws SQLException;

    int insertBook(ModelBook book) throws SQLException;

    int updateBook(ModelBook wherebook, ModelBook setbook) throws SQLException;

    int deleteBook(ModelBook book) throws SQLException;
}
```



# IServiceBook 만들기

```
import java.sql.*;

public interface IServiceBook extends IBook {

    // 추가 메서드. Service 에서만 사용되는 메서드.
    public int transCommit(ModelBook b1, ModelBook b2) ;
    public int transRollback(ModelBook b1, ModelBook b2);

}
```



# DaoBook 만들기

```
public class DaoBook implements IBook {  
    private java.sql.Connection conn = null;  
    public DaoBook(Connection conn) { this.conn = conn; } // 생성자  
  
    @Override  
    public int getCount(ModelBook book) { return -1; }  
    @Override  
    public int getMaxBookid() { return -1; }  
    @Override  
    public ResultSet selectAll() { return null; }  
    @Override  
    public ResultSet selectLike(ModelBook bookname) { return null; }  
    @Override  
    public ResultSet selectEqual(ModelBook book) { return null; }  
    @Override  
    public int insertBook(ModelBook book) { return 0; }  
    @Override  
    public int updateBook(ModelBook wherebook, ModelBook setbook) { return 0; }  
    @Override  
    public int deleteBook(ModelBook book) { return 0; }  
    @Override  
    public ResultSet selectDynamic(ModelBook book) { return null; }  
}
```



## DaoBook의 getCount(...) 메서드 작성하기

```
public int getCount(ModelBook book) throws java.sql.SQLException {  
    int result = -1;  
  
    // SQL 문장  
    String query = "SELECT count(*) as total from book where 1 = 1 ";  
  
    try {  
        // 문장 객체 생성  
        java.sql.PreparedStatement stmt = conn.prepareStatement(query);  
  
        // 문장 객체 실행  
        java.sql.ResultSet rs = stmt.executeQuery();  
  
        rs.next(); // 커서 이동. ResultSet 에서 첫번째로 row로 이동.  
        result = rs.getInt("total"); // total 컬럼의 값을 가져온다. rs.getInt( 0 );  
    } catch (java.sql.SQLException e) {  
        e.printStackTrace();  
    }  
    return result;  
}
```

SQL문장을 실행하고  
결과로 **ResultSet**을 반환한다.

1 SELECT count(\*) as total from book

결과 #1 (1×1)

total

4



## DaoBook의 getMaxBookid(...) 메서드 작성하기

```
public int getMaxBookid() throws java.sql.SQLException {  
    int result = -1;  
  
    // SQL 문장  
    String query = " select max(bookid) maxid from book ";  
  
    try {  
        // 문장 객체 생성  
        java.sql.PreparedStatement stmt = conn.prepareStatement(query);  
  
        // 문장 객체 실행  
        java.sql.ResultSet rs = stmt.executeQuery();  
  
        rs.next(); // 커서 이동. ResultSet 에서 첫번째로 row로 이동.  
        result = rs.getInt("maxid"); // maxid 컬럼의 값을 가져온다.  
    } catch (java.sql.SQLException e) {  
        e.printStackTrace();  
    }  
  
    return result;  
}
```

SQL문장을 실행하고  
결과로 **ResultSet**을 반환한다.

```
1 select max(bookid) maxid from book
```

결과 #1 (1×1)

maxid

4



## DaoBook의 selectAll(...) 메서드 작성하기

```
public java.sql.ResultSet selectAll() {
    java.sql.ResultSet rs = null;
    // SQL 문장
    String query = "SELECT * FROM book ORDER BY bookid ASC ";

    try {
        // 문장 객체 생성
        java.sql.PreparedStatement stmt = conn.prepareStatement(query);

        // 문장 객체 실행
        rs = stmt.executeQuery();
    } catch (java.sql.SQLException e) {
        e.printStackTrace();
    }

    return rs;
}
```

SQL문장을 실행하고  
결과로 **ResultSet**을 반환한다.

bookid	bookname	publisher	year	price	dtm	use_yn	authid
1	operating system	wiley	2003	30,700	2004-01-01	0	1
2	mysql	oreilly	2009	58,700	2010-01-01	1	2
3	java	hall	2013	40,000	2014-01-01	1	3
4	first sql	wiley	2015	57,700	2016-01-01	1	4



## DaoBook의 selectLike(...) 메서드 작성하기

```
public java.sql.ResultSet selectLike(ModelBook book) {  
    java.sql.ResultSet rs = null;  
  
    try {  
        // SQL 문장 생성  
        String query = "SELECT * FROM book where bookname like ? ";  
  
        // 문장 객체 생성  
        java.sql.PreparedStatement stmt = conn.prepareStatement(query);  
        stmt.setString(1, "%" + book.getBookkname() + "%");  
  
        // 문장 객체 실행  
        rs = stmt.executeQuery();  
    } catch (java.sql.SQLException e) {  
        e.printStackTrace();  
    }  
    return rs;  
}
```

query의 첫번째 ? 에 `"%" + book.getBookkname() + "%"` 값을 대입 한다

SQL문장을 실행하고  
결과 row를 반환한다.

1 select * from book where bookname like '%ja%'								
book (8x1)								
bookid	bookname	publisher	year	price	dtm	use_yn	authid	
3	java	hall	2013	40,000	2014-01-01	1	3	





## DaoBook의 selectEqual(..) 메서드 작성하기

```
public java.sql.ResultSet selectEqual(ModelBook book) {  
  
    java.sql.ResultSet rs = null;  
  
    try {  
        // SQL 문장 생성  
        String query = "SELECT * FROM book where bookname = ? ";  
  
        // 문장 객체 생성  
        java.sql.PreparedStatement stmt = conn.prepareStatement(query);  
        stmt.setString(1, book.getBookname());  
  
        // 문장 객체 실행  
        rs = stmt.executeQuery();  
    } catch (java.sql.SQLException e) {  
        e.printStackTrace();  
    }  
  
    return rs;  
}
```

query의 첫번째 ? 에 book.getBookname() 값을 대입 한다

SQL문장을 실행하고  
결과 row를 반환한다.

1 SELECT \* FROM book where bookname='mysql'

book (8x1)

bookid	bookname	publisher	year	price	dtm	use_yn	authid
2	mysql	oreilly	2009	58,700	2010-01-01	1	2



## DaoBook의 insertBook(...) 메서드 작성하기

```
public int insertBook(ModelBook book) throws java.sql.SQLException {  
    int rs = -1;  
    try {  
        String query = "";  
        query += " INSERT INTO                                \n";  
        query += " BOOK( BOOKNAME, PUBLISHER, YEAR, PRICE, DTM, USE_YN, AUTHID) \n";  
        query += " VALUES( ?, ?, ?, ?, ?, ?, ? )                \n";  
        java.sql.PreparedStatement stmt = conn.prepareStatement(query);  
        stmt.setString (1, book.getBookname()                );  
        stmt.setString (2, book.getPublisher()                );  
        stmt.setString (3, book.getYear()                    );  
        stmt.setInt    (4, book.getPrice()                   );  
        stmt.setDate   (5, (java.sql.Date) book.getDtm());  
        stmt.setBoolean(6, book.getUse_yn()                  );  
        stmt.setInt    (7, book.getAuthid()                  );  
  
        rs = stmt.executeUpdate();  
    } catch (java.sql.SQLException e) {  
        e.printStackTrace();  
    }  
    return rs;  
}
```

SQL문장을 실행하고  
영향 받은 **row**의 갯수를 반환한다.



## DaoBook의 updateBook(...) 메서드 작성하기

```
public int updateBook(ModelBook wherebook, ModelBook setbook) {  
    int rs = -1;  
    try {  
        // SQL 문장 생성  
        String query = "UPDATE book\n";  
        query      += "    SET year      = ? , price      = ?\n";  
        query      += " WHERE bookname = ?\n";  
  
        // 문장 객체 생성  
        java.sql.PreparedStatement stmt = conn.prepareStatement(query);  
        stmt.setString ( 1, setbook.getYear()      );  
        stmt.setInt    ( 2, setbook.getPrice()     );  
        stmt.setString ( 3, wherebook.getBookname() );  
  
        // 문장 객체 실행  
        rs = stmt.executeUpdate();  
    } catch (java.sql.SQLException e) {  
        e.printStackTrace();  
    }  
    return rs;  
}
```

SQL문장을 실행하고  
영향 받은 **row**의 갯수를 반환한다.



## DaoBook의 deleteBook(...) 메서드 작성하기

```
public int deleteBook(ModelBook book) {  
  
    int rs = -1;  
  
    try {  
        // SQL 문장 생성  
        String query = "delete from book where bookname = ? ";  
  
        // 문장 객체 생성  
        java.sql.PreparedStatement stmt = conn.prepareStatement(query);  
        stmt.setString ( 1, book.getBookname() );  
  
        // 문장 객체 실행  
        rs = stmt.executeUpdate();  
    } catch (java.sql.SQLException e) {  
        e.printStackTrace();  
    }  
  
    return rs;  
}
```

SQL문장을 실행하고  
영향 받은 **row**의 갯수를 반환한다.



## DaoBook의 selectDynamic(..) 메서드 작성하기

```
public java.sql.ResultSet selectDynamic(ModelBook book) throws java.sql.SQLException{
    java.sql.ResultSet result = null;

    try {
        // query 작성
        String query = " select * from book  \n";
        query += "   where 1 = 1           \n";
        if( book.getBookid() != null ) query += "   and bookid   = ? \n";
        if( !book.getBookname().isEmpty() ) query += "   and bookname = ? \n";

        // 문장 객체 생성
        java.sql.PreparedStatement stmt = conn.prepareStatement(query);

        int c = 1;
        if( book.getBookid() != null ) stmt.setInt( c++, book.getBookid() );
        if( !book.getBookname().isEmpty() ) stmt.setString( c++, book.getBookname());

        // 문장 객체 실행
        result = stmt.executeQuery();
    } catch (java.sql.SQLException e) {
        e.printStackTrace();
    }

    return result;
}
```



# ServiceBook 만들기

```
public class ServiceBook implements IServiceBook {  
    private Connection conn = null;  
    public ServiceBook() { this.conn = DBConnect.makeConnection(); }  
  
    @Override public int getCount(ModelBook mb) { return -1; }  
    @Override public int getMaxBookid() { return -1; }  
    @Override public java.sql.ResultSet selectAll() { return null; }  
    @Override public java.sql.ResultSet selectLike(ModelBook mb) { return null; }  
    @Override public java.sql.ResultSet selectEqual(ModelBook mb) { return null; }  
    @Override public int insertBook(ModelBook mb) { return -1; }  
    @Override public int updateBook(ModelBook whereb, ModelBook setb) { return -1; }  
    @Override public int deleteBook(ModelBook mb) { return -1; }  
    @Override public java.sql.ResultSet selectDynamic(ModelBook mb) { return null; }  
  
    @Override public int transCommit(ModelBook b1, ModelBook b2) { return null; }  
    @Override public int transRollback(ModelBook b1, ModelBook b2) { return null; }  
}
```



## ServiceBook의 getCount(...) 메서드 작성하기

```
@Override
public int getCount(ModelBook book) throws SQLException {

    int rs = -1;

    try {
        // 트랜잭션 시작
        conn.setAutoCommit( false );

        DaoBook dao = new DaoBook( this.conn );
        rs= dao.getCount(book);

        conn.commit(); // 트랜잭션 종료
    } catch (SQLException e) {
        e.printStackTrace();
        conn.rollback(); // 트랜잭션 롤백
    }

    return rs;
}
```



## ServiceBook의 getMaxBookid(...) 메서드 작성하기

```
@Override
public int getMaxBookid() throws SQLException {

    int rs = -1;

    try {
        // 트랜잭션 시작
        conn.setAutoCommit( false );

        DaoBook dao = new DaoBook( conn );
        rs = dao.getMaxBookid();

        // 트랜잭션 커밋
        conn.commit();
    } catch (SQLException e) {
        e.printStackTrace();
        conn.rollback();
    }

    return rs;
}
```





## ServiceBook의 selectAll(...) 메서드 작성하기

```
@Override
public ResultSet selectAll() throws SQLException {

    ResultSet rs = null;

    try {
        // 트랜잭션 시작
        conn.setAutoCommit( false );

        DaoBook dao = new DaoBook( conn );
        rs = dao.selectAll();

        // 트랜잭션 커밋
        conn.commit();
    } catch (SQLException e) {
        e.printStackTrace();
        conn.rollback();
    }

    return rs;
}
```



## ServiceBook의 selectLike(...) 메서드 작성하기

```
@Override
public ResultSet selectLike(ModelBook book) throws SQLException {

    ResultSet rs = null;

    try {
        // 트랜잭션 시작
        conn.setAutoCommit( false );

        DaoBook dao = new DaoBook(conn);
        rs = dao.selectLike(book);

        // 트랜잭션 커밋
        conn.commit();
    } catch (SQLException e) {
        e.printStackTrace();
        conn.rollback();
    }
    return rs;
}
```



## ServiceBook의 selectEqual(...) 메서드 작성하기

```
@Override
public ResultSet selectEqual(ModelBook book) throws SQLException {

    ResultSet rs = null;

    try {
        // 트랜잭션 시작
        conn.setAutoCommit( false );

        DaoBook dao = new DaoBook(conn);
        rs = dao.selectEqual(book);

        // 트랜잭션 커밋
        conn.commit();
    } catch (SQLException e) {
        e.printStackTrace();
        conn.rollback();
    }
    return rs;
}
```



## ServiceBook의 insertBook(...) 메서드 작성하기

```
@Override
public int insertBook(ModelBook book) throws SQLException {

    int result = -1;

    try {
        // 트랜잭션 시작
        conn.setAutoCommit(false);

        DaoBook dao = new DaoBook(conn);
        result = dao.insertBook(book);

        // 트랜잭션 커밋
        conn.commit();
    } catch (SQLException e) {
        e.printStackTrace();
        conn.rollback();
    }

    return result;
}
```



## ServiceBook의 updateBook(...) 메서드 작성하기

```
@Override
public int updateBook(ModelBook wherebook, ModelBook setbook) throws SQLException {

    int result = -1;

    try {
        // 트랜잭션 시작
        conn.setAutoCommit(false);

        DaoBook dao = new DaoBook(conn);
        result = dao.updateBook(wherebook, setbook);

        // 트랜잭션 커밋
        conn.commit();
    } catch (SQLException e) {
        e.printStackTrace();
        conn.rollback();
    }

    return result;
}
```



## ServiceBook의 deleteBook(...) 메서드 작성하기

```
@Override
public int deleteBook(ModelBook book) throws SQLException {

    int result = -1;

    try {
        // 트랜잭션 시작
        conn.setAutoCommit(false);

        DaoBook dao = new DaoBook(conn);
        result = dao.deleteBook(book);

        // 트랜잭션 커밋
        conn.commit();
    } catch (SQLException e) {
        e.printStackTrace();
        // 트랜잭션 롤백
        conn.rollback();
    }

    return result;
}
```



## ServiceBook의 selectDynamic(...) 메서드 작성하기

```
@Override
public ResultSet selectDynamic(ModelBook book) throws SQLException {

    ResultSet rs = null;

    try {
        // 트랜잭션 시작
        conn.setAutoCommit( false );

        DaoBook dao = new DaoBook(conn);
        rs = dao.selectDynamic(book);

        // 트랜잭션 커밋
        conn.commit();
    } catch (SQLException e) {
        e.printStackTrace();
        conn.rollback();
    }
    return rs;
}
```



## ServiceBook의 transRollback(...) 메서드 작성하기

```
public ResultSet transRollback(ModelBook b1, ModelBook b2) throws SQLException {  
  
    ResultSet rs = null;  
  
    try {  
        // 트랜잭션 시작  
        conn.setAutoCommit(false);  
        DaoBook dao = new DaoBook(conn);  
        dao.deleteBook(b1);  
        dao.insertBook(b2);  
        rs = dao.selectAll();  
        throw new SQLException();  
    } catch (SQLException e) {  
        e.printStackTrace();  
        conn.rollback();  
    }  
    return rs;  
}
```





## ServiceBook의 transCommit(...) 메서드 작성하기

```
public int transCommit(ModelAuth auth, ModelBook book) throws SQLException {  
    int result = -1;  
  
    try {  
        conn.setAutoCommit(false);  
  
        DaoAuth daoAuth = new DaoAuth(conn);  
        result = daoAuth.insert(auth);  
  
        // 추가. Auth 테이블에 inserted 된 authid 를  
        // book 테이블에 insert 할 때 사용하도록 수정.  
        book.setAuthid( result );  
  
        DaoBook daoBook = new DaoBook(conn);  
        result = daoBook.insert(book );  
  
        conn.commit();  
    } catch (SQLException e) {  
        e.printStackTrace();  
        conn.rollback();  
    }  
  
    return result;  
}
```



## ServiceBook에 printResultSet(...) 만들기 작성하기

1. ResultSet 을 매개변수로 받아 출력하는 메서드, printResultSet 메서드를 만드시오. (출력 예시)

1	operating system	wiley	2003	30700	2004-01-01	false
2	mysql	oreilly	2009	58700	2010-01-01	true
3	java	hall	2013	40000	2014-01-01	true
4	first sql	wiley	2015	57700	2016-01-01	true

2. selectAll(), selectLike(), selectEqual()에서 ResultSet 출력 부분을 삭제하시오.
3. main 메서드에서 selectAll() 호출 시 결과를 반환 받아 결과를 출력하도록 하시오.  

```
rs = dao.selectAll(conn);  
dao.printResultSet(rs);
```
4. main 메서드에서 selectLike() 호출 시 결과를 반환 받아 결과를 출력하도록 하시오.
5. main 메서드에서 selectEqual() 호출 시 결과를 반환 받아 결과를 출력하도록 하시오.



## ServiceBook의 printResultSet(...) 메서드 작성하기

```
public void printResultSet( ResultSet rs){
    rs.first();
    try {
        for ( ; rs.next(); ) {
            int id          = rs.getInt    ("bookid"    );
            String bookname = rs.getString ("bookname" );
            String publisher = rs.getString ("publisher" );
            String year      = rs.getString ("year"      );
            int price        = rs.getInt    ("price"     );
            Date dtm         = rs.getDate   ("dtm"        );
            boolean use_yn   = rs.getBoolean("use_yn"    );

            String out = String.format(" %d \t %15s \t %10s \t %4s \t %7d \t %10s \t %b", id, bookname, publisher, year, price, dtm, use_yn);
            System.out.println(out);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```



# TestServiceBook 만들기

```
public class TestServiceBook {  
  
    private static ServiceBook svr = null;  
  
    @BeforeClass  
    public static void setClass() throws Exception {  
        svr = new ServiceBook();  
    }  
  
    @Test public void selectAll() throws Exception { }  
    @Test public void selectLike() throws Exception { }  
    @Test public void selectEqual() throws Exception { }  
    @Test public void insertBook() throws Exception { }  
    @Test public void updateBook() throws Exception { }  
    @Test public void deleteBook() throws Exception { }  
  
    @Test public void selectDynamic() throws Exception { }  
    @Test public void transCommit() throws Exception { }  
    @Test public void transRollback() throws Exception { }  
}
```



## TestServiceBook의 selectAll() 테스트 코드 작성

```
public class TestServiceBook {
    static ServiceBook svr = null;

    @BeforeClass
    public static void setClass() throws Exception {
        svr = new ServiceBook();
    }

    @Test
    public void selectAll() throws Exception {
        ModelBook book = new ModelBook();

        ResultSet rs = svr.selectAll();
        assertNotNull(rs);
        rs.last();
        assertEquals(rs.getRow(), svr.getCount() );
    }
}
```



## TestServiceBook의 selectLike() 테스트 코드 작성

```
@Test
public void selectLike() throws Exception {

    ModelBook book = new ModelBook();
    book.setBookname("java");

    ResultSet rs = svr.selectLike(book);
    assertNotNull( rs );
    rs.last();

    ResultSet rs2 = svr.selectAll(book);
    assertNotNull(rs2);
    int count = 0;
    for( ; rs2.next() ; ) {
        if( rs2.getString("bookname").contains( book.getBookname() )){
            count = count +1;
        }
    }
    assertEquals( rs.getRow(), count );
}
```



## TestServiceBook의 selectEqual() 테스트 코드 작성

```
@Test
public void selectEqual() throws Exception {
    ModelBook book = new ModelBook();
    book.setBookname("java");

    ResultSet rs = svr.selectEqual(book);
    assertNotNull( rs );
    rs.last();

    ResultSet rs2 = svr.selectAll(book);
    assertNotNull(rs2);
    int count = 0;
    for( ; rs2.next() ; ) {
        if( rs2.getString("bookname").equalsIgnoreCase( book.getBookname() ) ){
            count = count +1;
        }
    }
    assertEquals( rs.getRow(), count );
}
```



## TestServiceBook의 insertBook() 테스트 코드 작성

```
@Test
public void insert() throws Exception {
    ModelBook book = new ModelBook();
    book.setBookname("");
    book.setAuthid( 1 );
    book.setDtm(Date.valueOf("2017-01-01"));
    book.setPrice(2000);
    //book.setBookid(1000);
    book.setYear("2017");
    book.setUse_yn(true);

    int rs = svr.insert(book);

    if( rs >=1 )
        assertTrue(true);
    else
        assertTrue(false);
}
```





## TestServiceBook의 updateBook() 테스트 코드 작성

```
@Test
public void update() throws Exception {

    ResultSet rs = svr.selectAll(null);
    assertNotNull(rs);
    rs.next();

    ModelBook where = new ModelBook();
    where.setBookname(rs.getString("bookname"));

    ModelBook set = new ModelBook();
    set.setYear("2017");
    set.setPrice(100);

    int rlt = svr.update(where, set);

    if (rlt >= 0)
        assertTrue(true);
    else
        assertTrue(false);
}
```



## TestServiceBook의 deleteBook() 테스트 코드 작성

```
@Test
public void delete() throws Exception {
    ModelBook book = new ModelBook();
    book.setBookname("aaa");

    int rlt = svr.delete(book);

    if (rlt >= 0) {
        assertTrue(true);
    }
    else {
        assertTrue(false);
    }
}
```



## TestServiceBook의 selectDynamic\_1() 테스트 코드 작성

```
@Test
public void selectDynamic_1() throws Exception {
    ModelBook book = new ModelBook();
    // select * from book where 1 = 1
    ResultSet rs = svr.selectDynamic(book);
    assertNotNull(rs);
    rs.last();

    assertEquals( rs.getRow(), svr.getCount(null) );
}
```



## TestServiceBook의 selectDynamic\_2() 테스트 코드 작성

```
@Test
public void selectDynamic_2() throws Exception {

    // select * from book where 1 = 1 and bookid = 1;
    ResultSet rs2 = svr.selectAll(null);
    assertNotNull(rs2);
    rs2.next();

    ModelBook book = new ModelBook();
    book.setBookid( rs2.getInt("bookid") );

    ResultSet rs = svr.selectDynamic(book);
    assertNotNull(rs);
    rs.next();

    assertEquals( rs.getString("bookname"), rs2.getString("bookname") );
}
```



## TestServiceBook의 selectDynamic\_3() 테스트 코드 작성

```
@Test
public void selectDynamic_3() throws Exception {

    // select * from book where 1 = 1 and bookname= 'aaa';
    ResultSet rs2 = svr.selectAll(null);
    assertNotNull(rs2);
    rs2.next();

    ModelBook book = new ModelBook();
    book.setBookname( rs2.getString("bookname") );

    ResultSet rs = svr.selectDynamic(book);
    assertNotNull(rs);
    rs.next();

    ResultSet rs3 = svr.selectEqual(book);
    assertNotNull(rs3);
    rs3.next();

    assertEquals( rs.getString("bookname"), rs3.getString("bookname") );
}
```



## TestServiceBook의 selectDynamic\_4() 테스트 코드 작성

```
@Test
public void selectDynamic_4() throws Exception {

    // select * from book where 1 = 1 and bookid = 1 and bookname= 'aaa';
    ResultSet rs2 = svr.selectAll(null);
    assertNotNull(rs2);
    rs2.next();

    ModelBook book = new ModelBook();
    book.setBookid ( rs2.getInt("bookid") );
    book.setBookname( rs2.getString("bookname") );

    ResultSet rs = svr.selectDynamic(book);
    assertNotNull(rs);
    rs.next();

    ResultSet rs3 = svr.selectEqual(book);
    assertNotNull(rs3);
    rs3.next();

    assertEquals( rs.getString("bookname"), rs3.getString("bookname") );
}
```



## TestServiceBook의 transCommit() 테스트 코드 작성

```
@Test
public void transCommit() throws Exception {
    ModelAuth auth = new ModelAuth();
    auth.setName( " test auth");
    auth.setBirth("1910-01-01");

    ModelBook book = new ModelBook();
    book.setBookname("test book");
    book.setUse_yn(true);
    book.setPublisher("test pub");
    book.setPrice(20000);
    book.setYear("2011");
    book.setDtm(java.sql.Date.valueOf("2017-05-15"));

    int result = svr.transCommit(auth, book);

    if(result>=0 )
        assertTrue(true);
    else
        assertTrue(false);
}
```



## TestServiceBook의 transRollback() 테스트 코드 작성

```
@Test  
public void transRollback() throws Exception {  
  
}
```



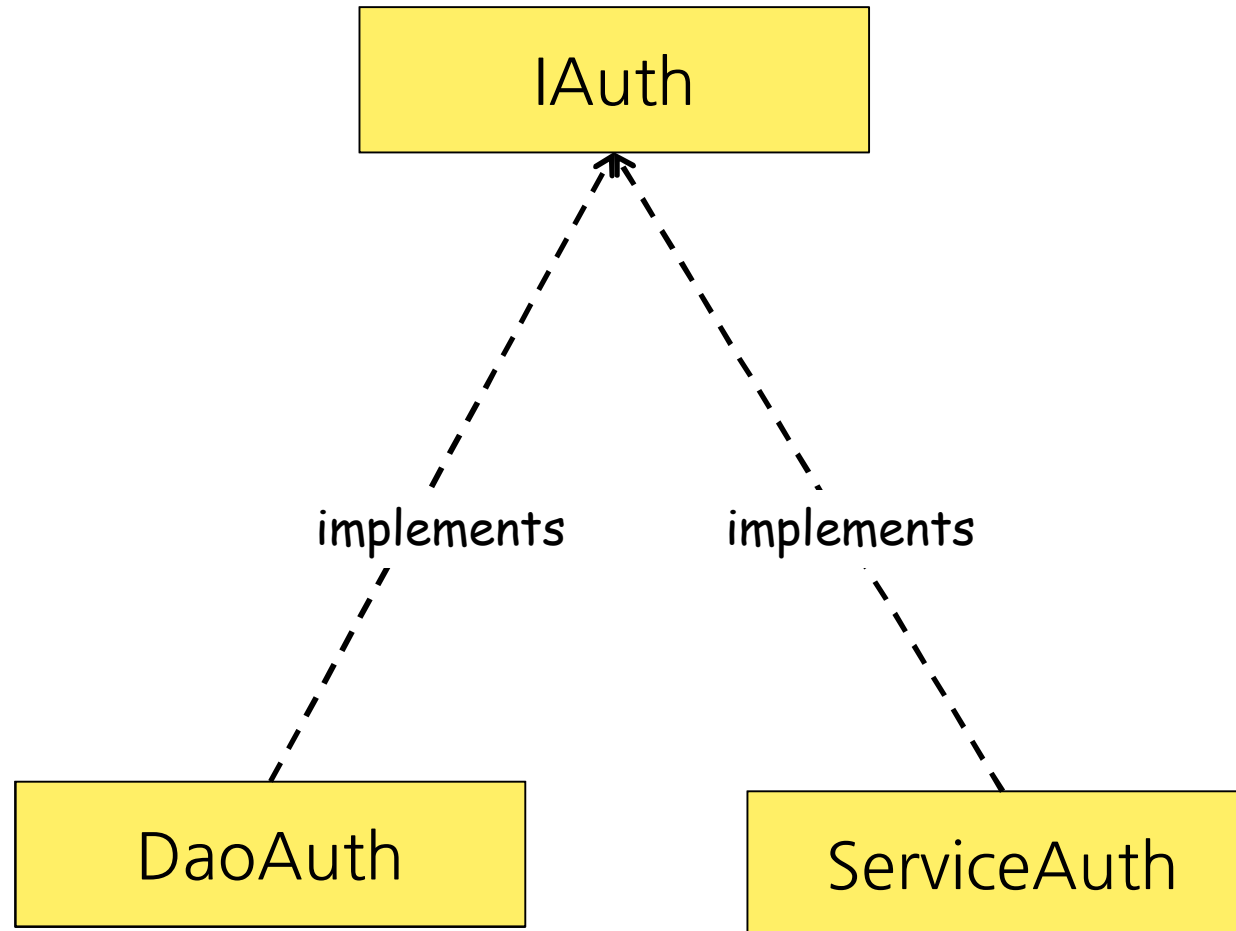


# Auth 관련 작업

- ModelAuth 클래스 만들기
- IAuth 인터페이스 만들기
- DaoAuth 클래스 만들기
  - java22.jdbc.test.TestDaoAuth 클래스 만들기
- ServiceAuth 클래스 만들기
  - java22.jdbc.test.TestServiceAuth 클래스만들기

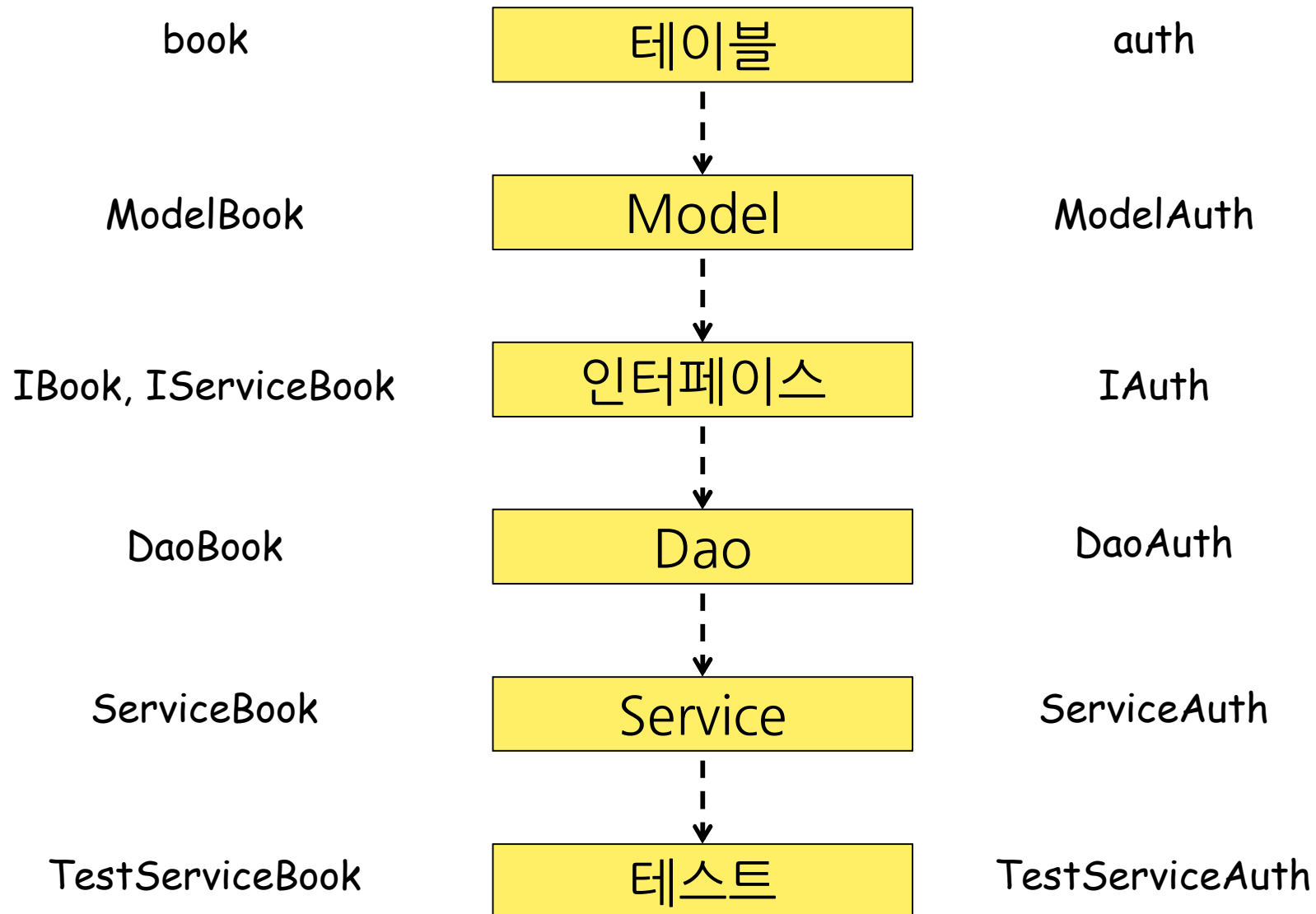


# Auth 인터페이스 구조





# DB Layer 만들기





# IAuth 만들기

```
import java.sql.*;

public interface IAuth {

    int getCount(ModelAuth auth) throws SQLException;

    int getMaxAuthid() throws SQLException;

    ResultSet selectAll() throws SQLException;

    ResultSet selectLike(ModelAuth auth) throws SQLException;

    ResultSet selectEqual(ModelAuth auth) throws SQLException;

    ResultSet selectDynamic(ModelAuth auth) throws SQLException;

    int insertAuth(ModelAuth auth) throws SQLException;

    int updateAuth(ModelAuth whereauth, ModelAuth setauth) throws SQLException;

    int deleteAuth(ModelAuth auth) throws SQLException;
}
```



# IServiceAuth 만들기

```
import java.sql.*;

public interface IServiceAuth extends IAuth {

    // 추가 메서드. ServiceAuth 에서만 사용되는 메서드.
    public int transCommit(ModelAuth a1, ModelAuth a2) ;
    public int transRollback(ModelAuth a1, ModelAuth a2);
}
```



# DaoAuth 만들기

```
public class DaoAuth implements IAuth {  
    private java.sql.Connection conn = null;  
    public DaoAuth(Connection conn) { this.conn = conn; } // 생성자  
  
    @Override  
    public int getCount(ModelAuth auth) { return -1; }  
    @Override  
    public int getMaxAuthid() { return -1; }  
    @Override  
    public ResultSet selectAll() { return null; }  
    @Override  
    public ResultSet selectLike(ModelAuth auth) { return null; }  
    @Override  
    public ResultSet selectEqual(ModelAuth auth) { return null; }  
    @Override  
    public int insertAuth(ModelAuth auth) { return 0; }  
    @Override  
    public int updateAuth(ModelAuth whereauth, ModelAuth setauth) { return 0; }  
    @Override  
    public int deleteAuth(ModelAuth auth) { return 0; }  
    @Override  
    public ResultSet selectDynamic(ModelAuth Auth) { return null; }  
}
```



# ServiceAuth 만들기

```
public class ServiceAuth implements IServiceAuth {  
    private java.sql.Connection conn = null;  
    public ServiceAuth() { this.conn = DBConnect.makeConnection(); }  
  
    @Override public int getCount(ModelAuth auth) { return -1; }  
    @Override public int getMaxAuthid() { return -1; }  
  
    @Override public java.sql.ResultSet selectAll() { return null; }  
    @Override public java.sql.ResultSet selectLike(ModelAuth auth) { return null; }  
    @Override public java.sql.ResultSet selectEqual(ModelAuth auth) { return null; }  
  
    @Override public int insertAuth(ModelAuth auth) { return -1; }  
    @Override public int updateAuth(ModelAuth whereb, ModelAuth setb) { return -1; }  
    @Override public int deleteAuth(ModelAuth auth) { return -1; }  
  
    @Override public java.sql.ResultSet selectDynamic(ModelAuth auth) { return null; }  
  
    @Override public int transCommit(ModelBook b1, ModelBook b2) { return null; }  
    @Override public int transRollback(ModelBook b1, ModelBook b2) { return null; }  
}
```



# TestServiceAuth 만들기

```
public class TestServiceAuth {  
  
    private static ServiceAuth svr = null;  
  
    @BeforeClass  
    public static void setClass() throws Exception {  
        svr = new ServiceAuth();  
    }  
  
    @Test public void selectAll() throws Exception { }  
    @Test public void selectLike() throws Exception { }  
    @Test public void selectEqual() throws Exception { }  
  
    @Test public void insertAuth() throws Exception { }  
    @Test public void updateAuth() throws Exception { }  
    @Test public void deleteAuth() throws Exception { }  
  
    @Test public void selectDynamic() throws Exception { }  
    @Test public void transCommit() throws Exception { }  
    @Test public void transRollback() throws Exception { }  
}
```





## DaoAuth의 selectDynamic(..) 메서드 작성하기

DaoAuth 클래스에 selectDynamic(..) 메서드를 작성한다.

```
String query = " select * from auth \n";
               query += " where 1 = 1      \n";
if( auth.getAuthid() != null ) query += " and authid = ? \n";
if( !auth.getName().isEmpty() ) query += " and name = ? \n";
if( !auth.getBirth().isEmpty() ) query += " and birth = ? \n";
```

ServiceAuth 클래스에 selectDynamic(..) 메서드를 작성한다.

TestServiceAuth에 테스트 메서드 testSelectDynamic(...) 를 작성하여 본다.

- 작성할 테스트 케이스

1. select \* from auth where 1 = 1
2. select \* from auth where 1 = 1 and authid=1
3. select \* from auth where 1 = 1 and name='bob'
4. select \* from auth where 1 = 1 and birth='2000.05.01'
5. select \* from auth where 1 = 1 and authid=1 and name='bob'
6. select \* from auth where 1 = 1 and authid=1 and birth='2000.05.01'
7. select \* from auth where 1 = 1 and authid=1 and name='bob' and birth='2000.05.01'