



# JSP/Servlet

chap8 데이터베이스와 JDBC

# 데이터베이스와 JSP의 연동

---

## 학습 목표:

- 데이터베이스 연결 기술인 JDBC의 개념 이해
- JSP페이지에서 JDBC를 사용하여 데이터베이스를 연동

# 데이터베이스의 개요 및 설치

---

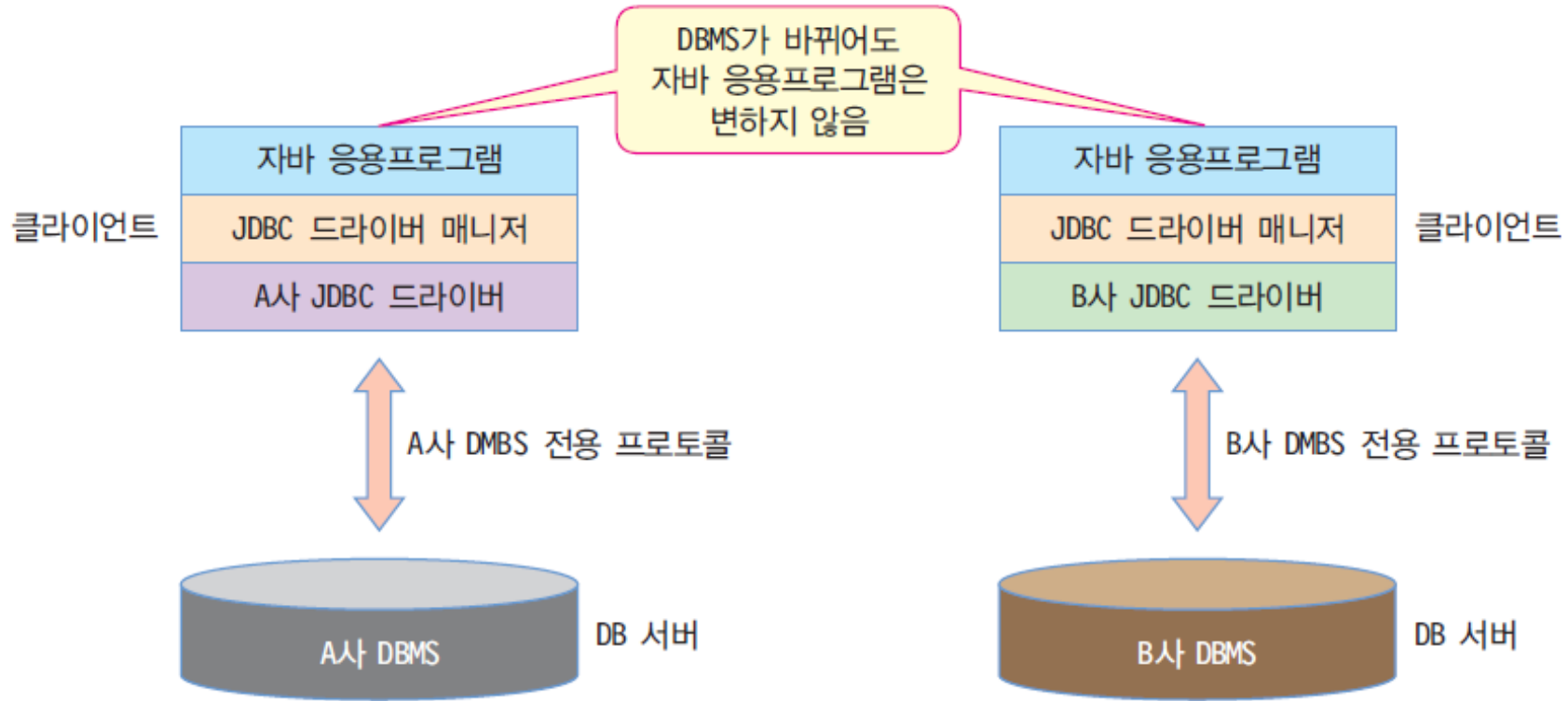
- 데이터베이스와 DBMS(Database Management System)
  - 데이터베이스
    - 데이터의 효율적인 관리를 목적으로 하는 데이터의 집합으로, 데이터를 지속적으로 관리하는 것이 목적
  - DBMS
    - 데이터를 안정적으로 보관할 수 있는 다양한 기능을 제공.
    - 주요한 기능: 데이터의 삽입/수정/삭제, 데이터의 무결성 유지, 트랜잭션관리, 데이터의 백업 및 복원, 데이터 보안기능.

# SQL과 JDBC

---

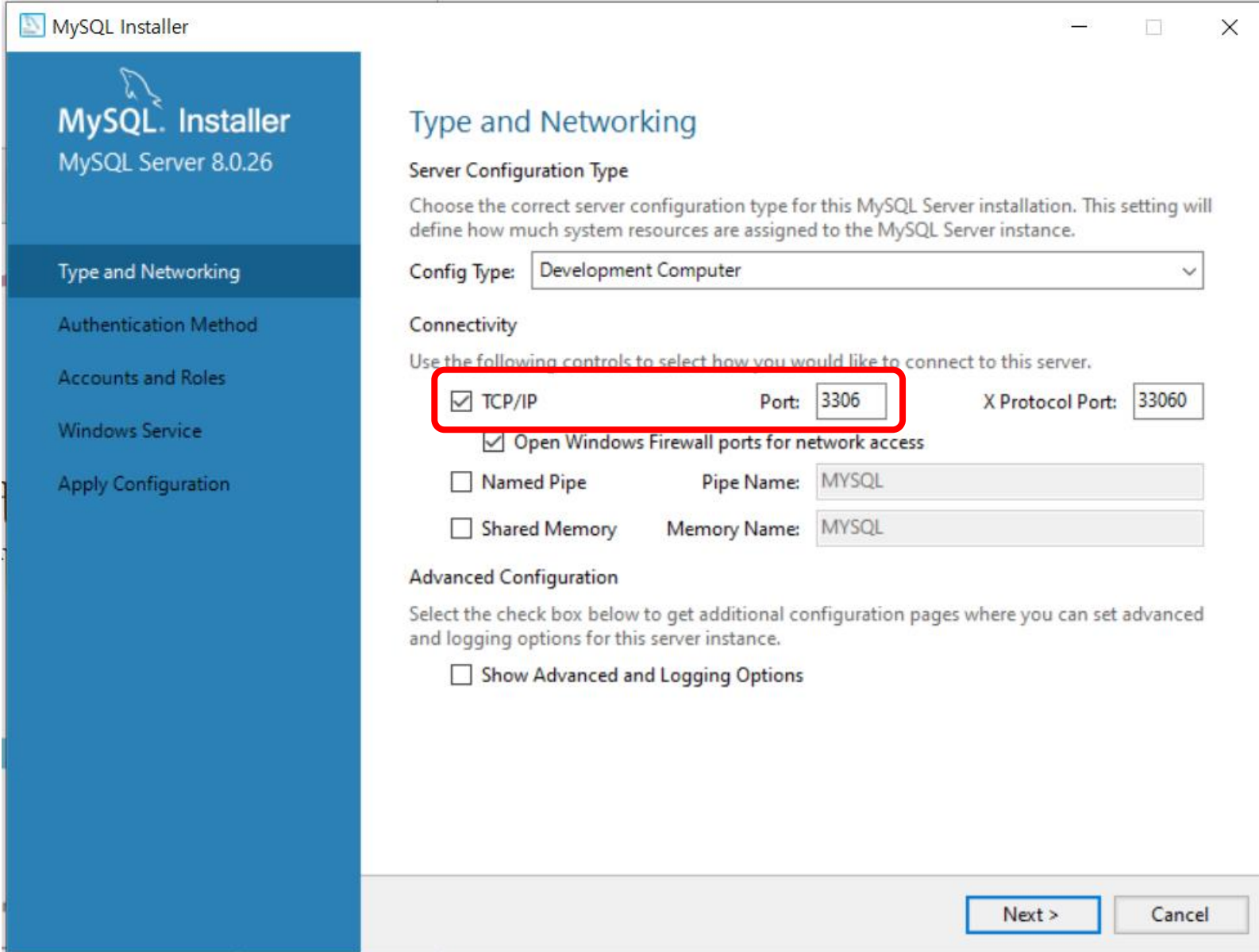
- SQL(Structured Query Language)
  - 관계형 데이터베이스 관리 시스템에서 사용
  - 데이터베이스 스키마 생성, 자료의 검색, 관리, 수정, 그리고 데이터베이스 객체 접근 관리를 위해 고안된 언어
  - 데이터베이스로부터 정보를 추출하거나 갱신하기 위한 표준 대화식 프로그래밍 언어
    - 다수의 데이터베이스 관련 프로그램들이 SQL을 표준으로 채택
- JDBC (Java DataBase Connectivity)
  - **관계형 데이터베이스에 저장된 데이터를 접근 및 조작할 수 있게 하는 API**
  - 다양한 DBMS에 대해 일관된 API로 데이터베이스 연결, 검색, 수정, 관리 등을 할 수 있게 함

# JDBC 구조



- JDBC 드라이버 매니저
  - 자바 API에서 지원하며 DBMS를 접근할 수 있는 JDBC 드라이버 로드
- JDBC 드라이버
  - DBMS마다 고유한 JDBC 드라이버 제공, JDBC 드라이버와 DBMS는 전용 프로토콜로 데이터베이스 처리
- DBMS
  - 데이터베이스 관리 시스템. 데이터베이스 생성·삭제, 데이터 생성·검색·삭제 등 전담 소프트웨어 시스템

# MySQL 서버

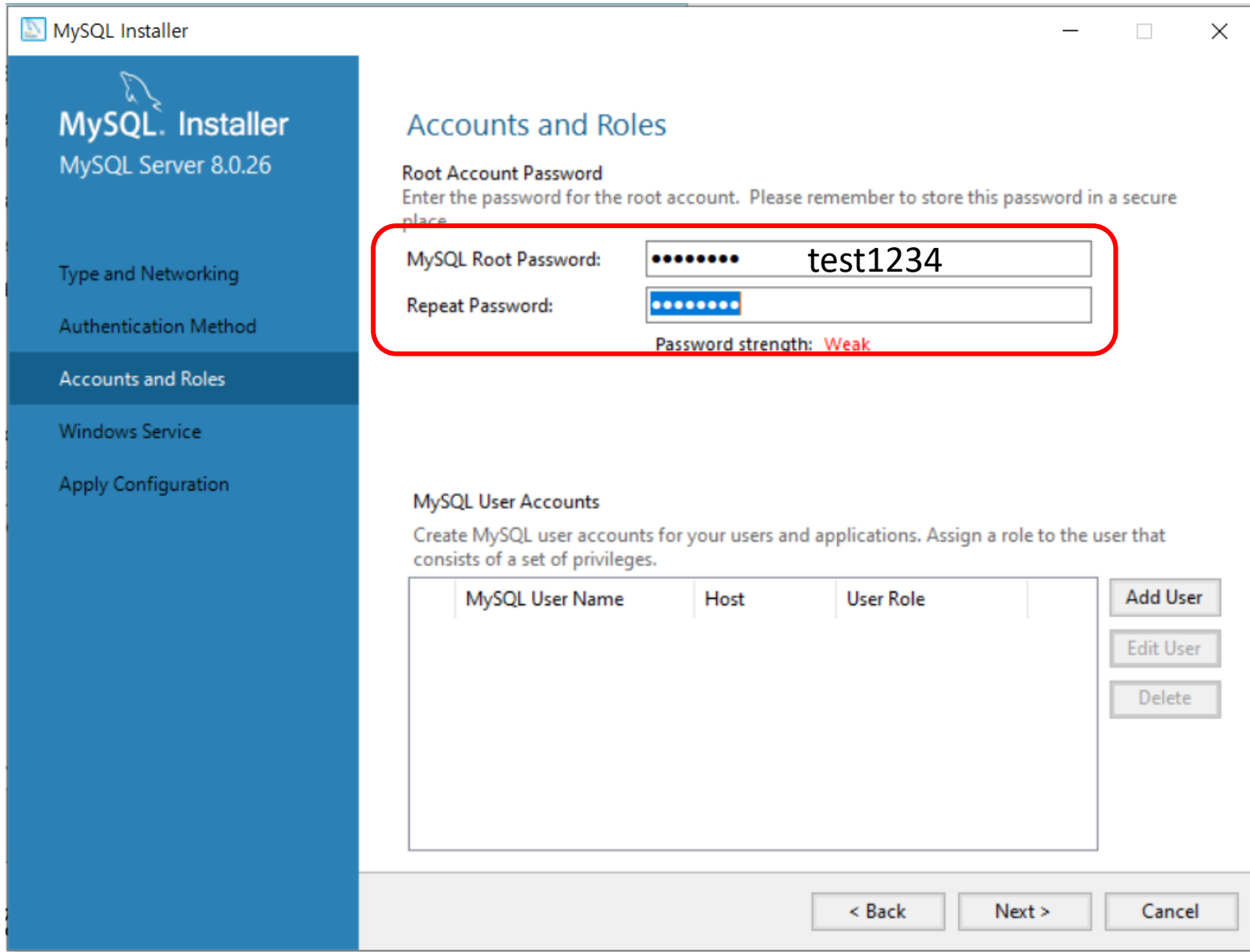


The image shows the MySQL Installer window for MySQL Server 8.0.26. The left sidebar contains the following navigation items: MySQL Installer, MySQL Server 8.0.26, Type and Networking (selected), Authentication Method, Accounts and Roles, Windows Service, and Apply Configuration. The main area is titled 'Type and Networking' and contains the following sections:

- Server Configuration Type**  
Choose the correct server configuration type for this MySQL Server installation. This setting will define how much system resources are assigned to the MySQL Server instance.  
Config Type: Development Computer
- Connectivity**  
Use the following controls to select how you would like to connect to this server.
  - ☒ TCP/IP Port: 3306 X Protocol Port: 33060
  - ☒ Open Windows Firewall ports for network access
  - ☐ Named Pipe Pipe Name: MYSQL
  - ☐ Shared Memory Memory Name: MYSQL
- Advanced Configuration**  
Select the check box below to get additional configuration pages where you can set advanced and logging options for this server instance.
  - ☐ Show Advanced and Logging Options

At the bottom right, there are 'Next >' and 'Cancel' buttons.

# MySQL 서버



The image shows the MySQL Installer window for MySQL Server 8.0.26. The left sidebar contains the following navigation items: MySQL Installer, MySQL Server 8.0.26, Type and Networking, Authentication Method, Accounts and Roles (selected), Windows Service, and Apply Configuration. The main area is titled 'Accounts and Roles' and contains two sections. The first section, 'Root Account Password', prompts the user to enter a password for the root account. It includes a text box for the password (displayed as 'test1234') and a 'Repeat Password' text box. A red box highlights these two text boxes. Below the text boxes, the password strength is indicated as 'Weak'. The second section, 'MySQL User Accounts', prompts the user to create MySQL user accounts. It includes a table with columns 'MySQL User Name', 'Host', and 'User Role'. To the right of the table are three buttons: 'Add User', 'Edit User', and 'Delete'. At the bottom of the window are three buttons: '< Back', 'Next >', and 'Cancel'.

MySQL Installer

MySQL Server 8.0.26

Type and Networking

Authentication Method

Accounts and Roles

Windows Service

Apply Configuration

### Accounts and Roles

Root Account Password

Enter the password for the root account. Please remember to store this password in a secure place.

MySQL Root Password: test1234

Repeat Password:

Password strength: Weak

### MySQL User Accounts

Create MySQL user accounts for your users and applications. Assign a role to the user that consists of a set of privileges.

MySQL User Name	Host	User Role
-----------------	------	-----------

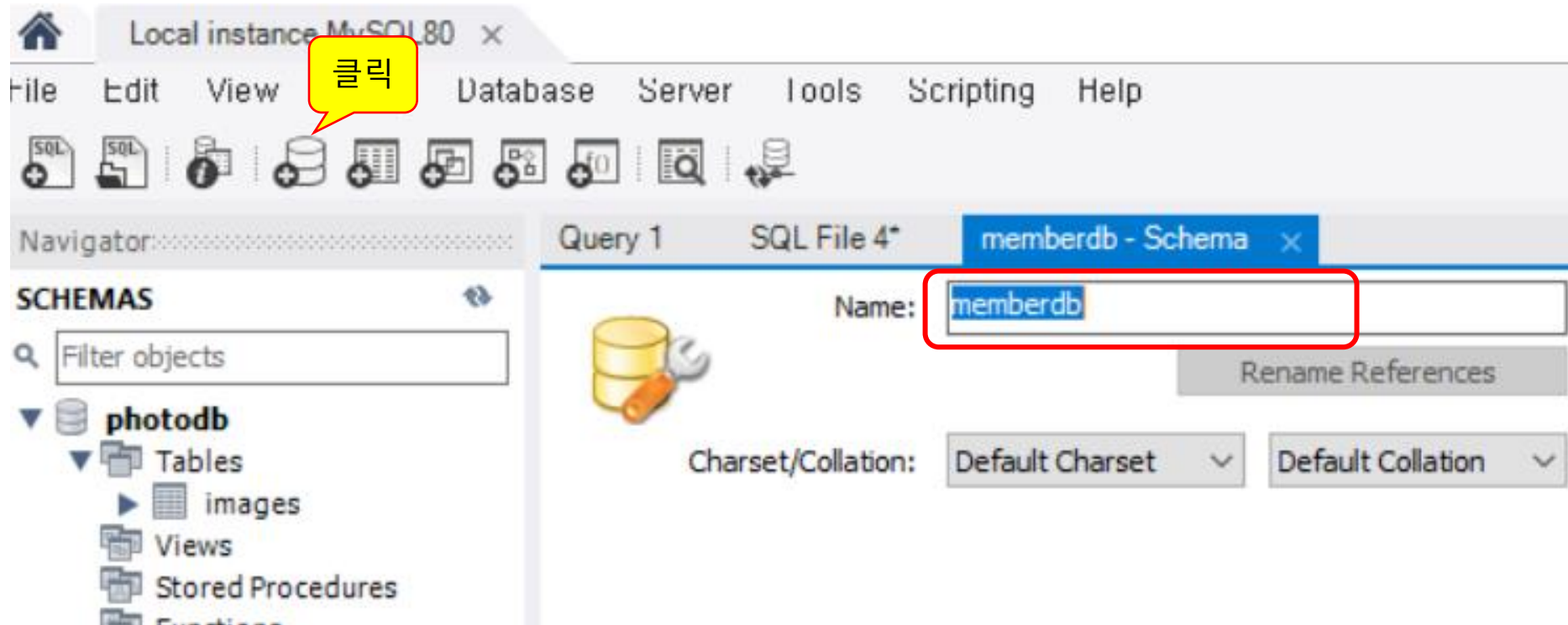
Add User

Edit User

Delete

< Back Next > Cancel


# 스키마 생성 & 테이블 생성










# 스키마 생성 & 테이블 생성

Query 1   SQL File 4\*   memberdb - Schema   member - Table x

 Table Name:  Schema: **memberdb**

Charset/Collation:   Engine:

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
 name	VARCHAR(10)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 userid	VARCHAR(10)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 pwd	VARCHAR(10)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 email	VARCHAR(20)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 phone	CHAR(13)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

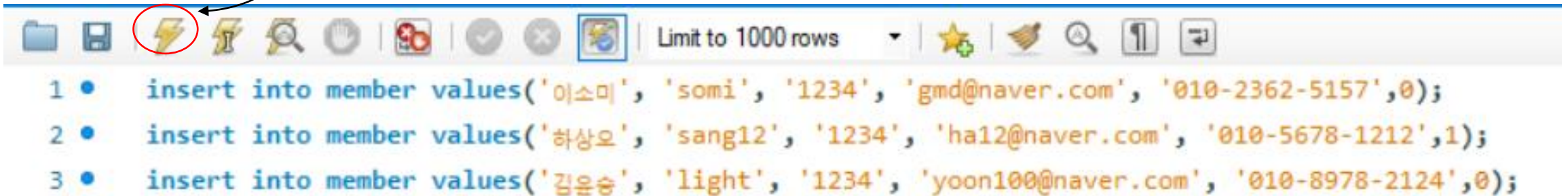
Apply

```
CREATE TABLE `memberdb`.`member` (  
  `name` VARCHAR(10) NOT NULL,  
  `userid` VARCHAR(10) NOT NULL,  
  `pwd` VARCHAR(10) NOT NULL,  
  `email` VARCHAR(20) NOT NULL,  
  `phone` CHAR(13) NOT NULL,  
  `admin` INT NOT NULL DEFAULT 0,  
  PRIMARY KEY (`userid`));
```

# 레코드 추가

- 테이블에는 레코드 단위로 데이터 추가
  - insert into 테이블명 (필드명1, 필드명2, ...) values (값1, 값2, ... );

클릭하여 쿼리문 실행



The screenshot shows a toolbar with various icons. A red circle highlights the 'Execute' icon (a lightning bolt), with an arrow pointing to it from the text '클릭하여 쿼리문 실행'. Below the toolbar, three SQL insert statements are listed, each preceded by a blue bullet point and a line number.

```
1 • insert into member values('이소미', 'somi', '1234', 'gmd@naver.com', '010-2362-5157',0);  
2 • insert into member values('하상오', 'sang12', '1234', 'ha12@naver.com', '010-5678-1212',1);  
3 • insert into member values('김은승', 'light', '1234', 'yoon100@naver.com', '010-8978-2124',0);
```

# 데이터 검색

- select문으로 테이블 내의 데이터 검색
  - select name, phone from member where userid='somi';
    - select 다음에는 데이터를 추출할 열 이름을 콤마로 분리하여 나열
    - 모든 열에 대해 데이터를 추출할 때는 \*를 열 이름 대신 사용
    - from 다음에 테이블 이름을 지정
    - where 다음에 검색 조건 지정. 위의 예에서 id 값이 'somi'인 레코드 검색
    - where는 생략 가능

```
select * from member;
```

name	userid	pwd	email	phone	admin
김윤슬	light	1234	yoony100@naver.com	010-8978-2124	0
하상오	sang12	1234	ha12@naver.com	010-5678-1212	1
이소미	somi	1234	gmd@naver.com	010-2362-5157	0
NULL	NULL	NULL	NULL	NULL	NULL

```
select name, phone from member where userid='somi';
```

name	phone
이소미	010-2362-5157

# 데이터 수정

- update문을 이용하여 데이터 수정
- 명령
  - update member set phone='010-2123-4351' where userid='somi';
    - update 다음에는 테이블 이름 지정
    - set 다음에 수정할 열의 이름과 값을 콤마로 분리하여 나열
    - where 다음에는 검색 조건을 지정.
    - where는 생략 가능

```
5 • select name, phone from member where userid='somi';  
6 • update member set phone='010-2123-4351' where userid='somi';
```

<		
Result Grid     Filter Rows: <input type="text"/>   Export:    Wrap Cell Content: 		
	name	phone
▶	이소미	010-2123-4351

Update 실행후, select 문으로 확인결과

# 레코드 삭제

- delete문을 이용하여 데이터 삭제
- 명령
  - delete from member where name='이소미';
    - delete from 다음에는 테이블 이름 지정
    - where 다음에는 검색 조건을 지정. 위의 예에서는 name 값이 '이소미'인 레코드 삭제
    - where는 생략 가능

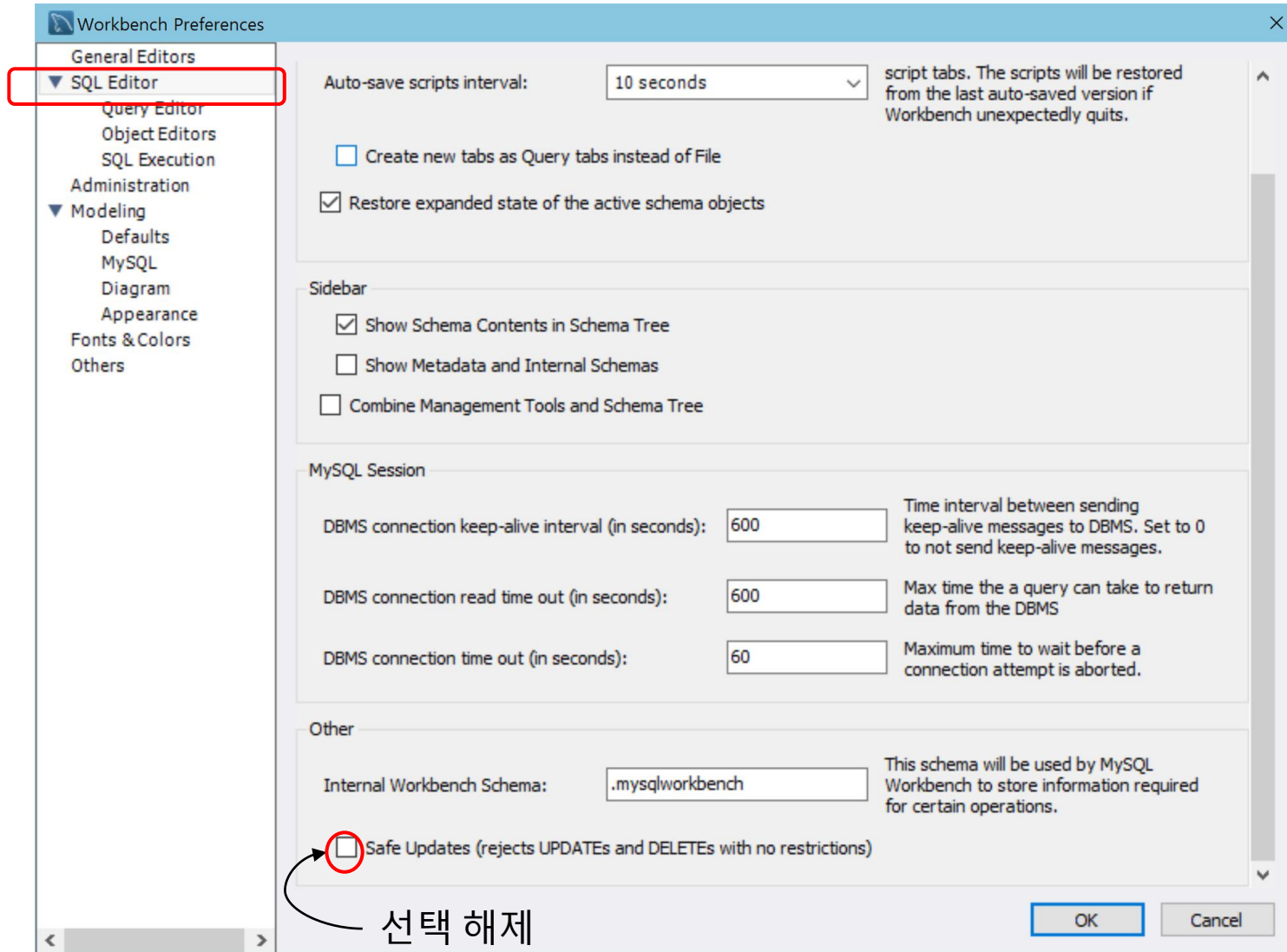
```
4 • select * from member;
5 • select name, phone from member where userid='somi';
6 • update member set phone='010-2123-4351' where userid='somi';
7 • delete from member where name='이소미';
```

8

Result Grid						
		Filter Rows:		Edit:		Export/Import:
name	userid	pwd	email	phone	admin	
김윤슬	light	1234	yon100@naver.com	010-8978-2124	0	
하상오	sang12	1234	ha12@naver.com	010-5678-1212	1	

# 데이터 수정, 삭제시(Edit>Preferences>SQL Editor)

Workbench에서 데이터 수정 또는 삭제가 안될 경우 삭제 시 옵션 확인 필요



# JSP를 위한 MySQL 드라이버 설치


- 이클립스 프로젝트에 MySQL Connector/J 드라이버 연결

## 1) 드라이버 다운로드

<https://downloads.mysql.com/archives/c-j/>

### MySQL Product Archives

MySQL Connector/J (Archived Versions)

 Please note that these are old versions. New releases will have recent bug fixes and features!  
To download the latest release of MySQL Connector/J, please visit [MySQL Downloads](#).

Product Version:

Operating System:

Platform Independent (Architecture Independent),  
Compressed TAR Archive  
(mysql-connector-java-8.0.29.tar.gz) MD5: 25fea8c2e6b6ec630db7438bfbff6d29 | [Signature](#)

Platform Independent (Architecture Independent),  
ZIP Archive

mysql-connector-java-8.0.29 > mysql-connector-java-8.0.29

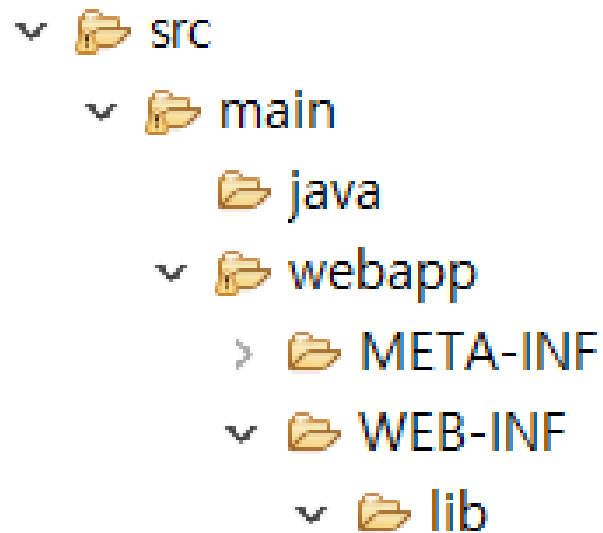
이름	수정된 날짜
src	2022-03-
build.xml	2022-03-
CHANGES	2022-03-
INFO_BIN	2022-03-
INFO_SRC	2022-03-
LICENSE	2022-03-
mysql-connector-java-8.0.29.jar	2022-08-
README	2022-03-

다운로드 후 압축 풀기

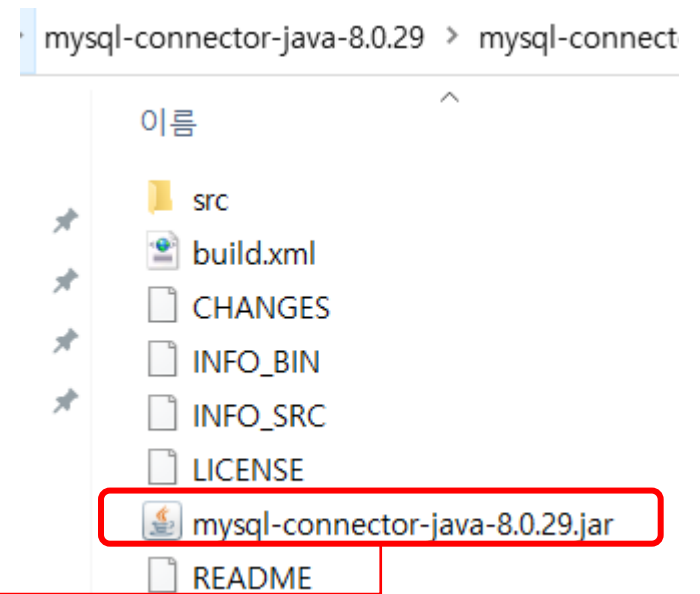
# JSP를 위한 MySQL 드라이버 설치

## 2) JDBC 사용할 프로젝트에 드라이브 설정하기

- 해당드라이버 설치 위치 : [webapp]-[WEB-INF]-[lib]폴더



mysql-connector-java-8.0.29.jar



드라이브 파일  
복사&붙이기



# MySQL 드라이버 연결 테스트

```
<%
    Connection conn;
    //DB에 접근하게 해주는 객체

    PreparedStatement pstmt;
    ResultSet rs;
    //정보를 담을 객체
    try
    {
        String dbURL = "jdbc:mysql://localhost:3306/memberdb?useUnicode=true&characterEncoding=utf-8";
        //localhost의 MySQL 3306포트 memberdb라는 DB 접속 경로를 dbURL에 저장.

        String dbID = "root";
        String dbPassword = "test1234";
        Class.forName("com.mysql.jdbc.Driver");
        //MySQL에 접속할 수 있도록 매개체 역할을 해주는 하나의 라이브러리, JDBC 드라이버 로드

        conn = DriverManager.getConnection(dbURL, dbID, dbPassword);
        //DB 접속되면 conn 객체에 접속정보가 저장됨.
        System.out.print("커넥션 성공: ");
    }catch(Exception e)
    {
        e.printStackTrace();
    }
%>
```

# 회원 정보 보기(allMember.jsp)

코드 수정: 선언부

```
<%!//선언부는 첫 방문자에 의해서 단 한번 수행합니다.  
    Connection conn = null;  
    Statement stmt = null;  
    ResultSet rs = null;  
    String url = "jdbc:mysql://localhost:3306/memberdb?"  
                +"useUnicode=true&characterEncoding=utf-8";  
    String uid = "root";  
    String pass = "test1234";  
    String sql = "select * from member";  
%>
```

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

allMember.jsp 실행 결과

이름	아이디	암호	이메일	전화번호	권한(1:관리자, 2:일반회원)
김윤승	light	1234	yoons100@naver.com	010-8978-2124	0
하상오	sang12	1234	ha12@naver.com	010-5678-1212	1

# 회원정보 추가하기

addMemberForm.jsp 실행 결과

## 회원의 정보 입력 폼

이름	<input type="text" value="홍길동"/>
아이디	<input type="text" value="hkdong"/>
비밀번호	<input type="password" value="...."/>
이메일	<input type="text" value="kdhong@naver.com"/>
전화번호	<input type="text" value="010-2344-1234"/>
등급	<input type="radio"/> 관리자 <input checked="" type="radio"/> 일반회원
<input type="button" value="전송"/>	<input type="button" value="취소"/>

전송

addMember.jsp

회원 가입 성공

[회원 전체 목록 보기](#)

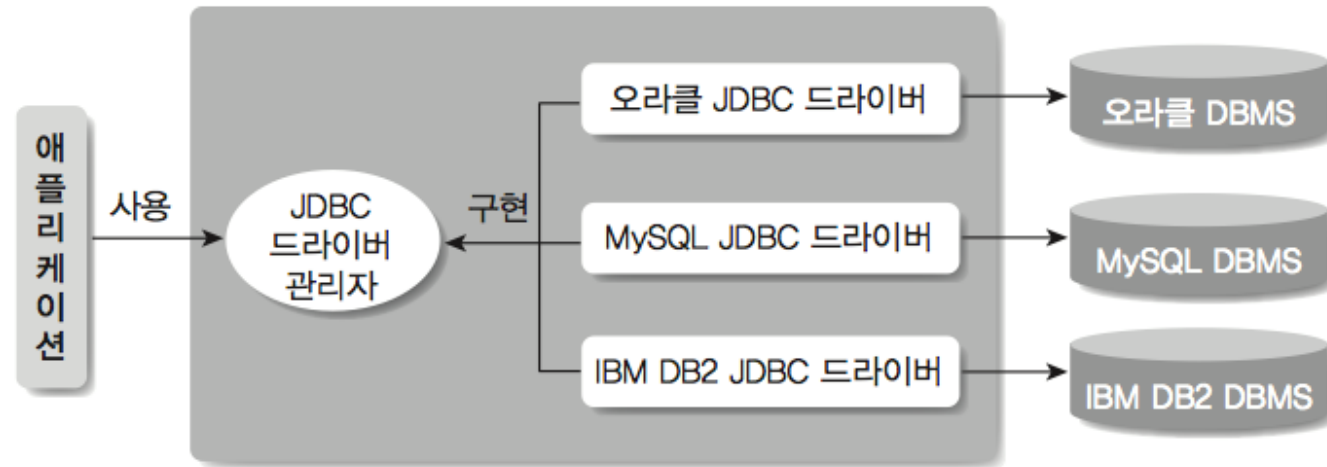
정보 추가 후 목록 보기

이름	아이디	암호	이메일	전화번호	권한(1:관리자, 2:일반회원)
홍길동	hkdong	1234	kdhong@naver.com	010-2344-1234	0
김윤승	light	1234	yoony100@naver.com	010-8978-2124	0
하상오	sang12	1234	ha12@naver.com	010-5678-1212	1

# JDBC 기본 구조와 API 이해

## ■ JDBC(Java Database Connectivity)란 ?

- JDBC 는 자바 프로그램에서 서로 다른 데이터베이스를 (표준화된 방법으로) 접속 할 수 있도록 만든 (API 규격)
- JDBC 라이브러리는 'java.sql' 패키지에의 구현

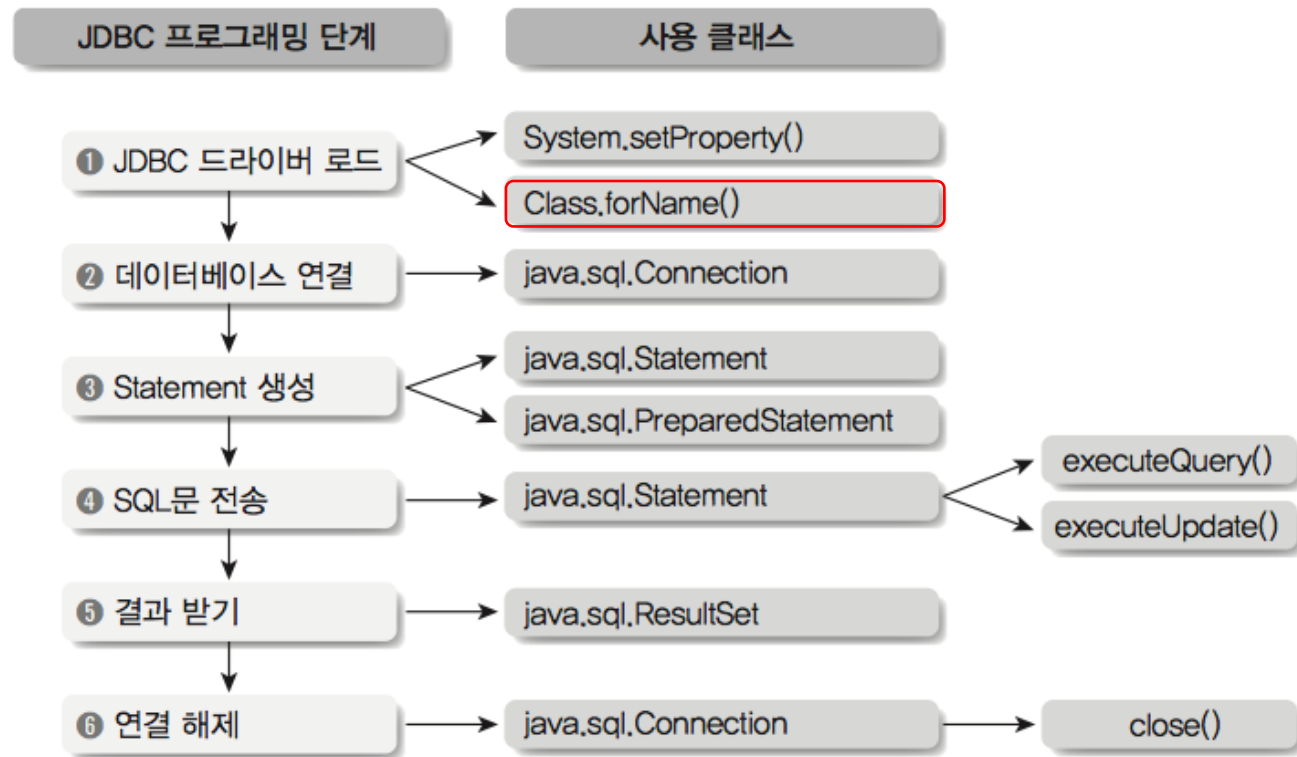


[JDBC 구조]

# JDBC 기본 구조와 API 이해

## ■ JDBC 프로그래밍 개요

- JDBC API를 이용해 정해진 절차와 규격에 따라 프로그램을 작성
- JSP 에서 JDBC 사용 : 스크립트릿으로 작성 또는 자바 빈즈로 연결



JDBC 프로그래밍 단계

# JDBC를 사용한 JSP와 데이터베이스의 연동

---

- JDBC를 사용한 JSP와 데이터베이스의 연동
- JDBC프로그램의 작성 단계
  - 1단계(JDBC 드라이버 Load)
    - Class 클래스의 forName() 메소드를 사용해서 드라이버 로드.
    - MySQL 드라이버를 로딩

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



MySQL의 JDBC 메인 클래스  
이름으로 오타나면 안됨

# JDBC를 사용한 JSP와 데이터베이스의 연동

- 2단계(**Connection 객체 생성**) : Connection 객체 연결 단계
  - DriverManager에 등록된 각 드라이버들을 **getConnection(String url)** 메소드를 사용해서 식별하고, Connection 객체 생성

getConnection() 메소드 매개변수 구성

"jdbc:하위 프로토콜:IP주소:PORT/스키마", "아이디", "비밀번호"

**Connection conn= DriverManager.getConnection(url, uid, pass);**

```
String url = "jdbc:mysql://localhost:3306/memberdb?"  
            + "useUnicode=true&characterEncoding=utf-8";  
String uid = "root";  
String pass = "test1234";
```

# JDBC를 사용한 JSP와 데이터베이스의 연동

- 3단계(**Statement/PreparedStatement 객체 생성**) : 데이터베이스 연결로부터 SQL 문을 수행할 수 있도록 지원하는 클래스
  - 3단계부터는 JDBC드라이버에 구매 받지 않음

Statement 객체 생성

```
Statement stmt = conn.createStatement();
```

PreparedStatement 객체 생성

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



# JDBC를 사용한 JSP와 데이터베이스의 연동

---

- Statement/PreparedStatement 객체의 대표적 메소드
  - executeQuery()
    - SELECT문을 수행할 때 사용
    - 반환 값은 ResultSet 클래스의 인스턴스
  - executeUpdate()
    - UPDATE, DELETE와 같은 문을 수행할 때 사용
    - 반환 값은 int 값으로, 처리된 데이터의 수를 반환

# JDBC를 사용한 JSP와 데이터베이스의 연동

- Statement 객체는 쿼리를 문자열로 연결해야 하므로 소스가 복잡하고 오류가 발생하기 쉽다.

```
Statement stmt = conn.createStatement();  
  
Stmt.executeUpdate("insert into test values  
( ' "+request.getParameter("username")+"  
' '"+request.getParameter  
("email")+" ' '");
```

# JDBC를 사용한 JSP와데이터베이스의 연동

---

- PreparedStatement는 SQL 에 필요한 변수 데이터를 “?”로 표시하고 메서드를 통해 설정하는 방식으로 Statement 보다 구조적이고 편리해 권장되는 방법

```
PreparedStatement pstmt = conn.prepareStatement("insert into test values(?,?)");  
  
pstmt.setString(1,request.getParameter("username");  
  
pstmt.setString(2,request.getParameter("email");  
  
pstmt.executeUpdate();
```

# JDBC를 사용한 JSP와데이터베이스의 연동

- 4단계(**Query 수행**) : 데이터 조합과 함께 만들어진 SQL 문은 명시적인 처리 명령에 의해 DB에 전달되어 실행
- executeQuery(), executeUpdate() 사용
  - stmt.executeQuery() : recordSet 반환 => Select 문에서 사용

```
ResultSet rs = stmt.executeQuery ("select * from 소속기관");
```

- stmt.executeUpdate(): 성공한 row 수 반환 => Insert 문, Update 문, Delete 문에서 사용

```
String sql="update member set passwd='3579' where id='abc'";  
stmt.executeUpdate(sql);
```

# JDBC를 사용한 JSP와 데이터베이스의 연동

- 5단계(**ResultSet 처리**) : executeQuery()메소드 처리 결과를 ResultSet 객체로 받아 처리
  - ResultSet객체는 '**커서(cursor)**'라 불리는 것을 가지고 있는데, 이것을 사용해 ResultSet 객체에서 특정 레코드를 참조.
  - 커서는 초기에 첫 번째 레코드(행)의 직전 즉, 필드명이 위치한 곳을 가리킴
  - ResultSet의 첫 번째 필드는 **1**부터 시작.
  - rs.getString("name")과 같이 필드명을 사용하는 것이 권장 형태.

# JDBC를 사용한 JSP와 데이터베이스의 연동



A diagram illustrating the cursor position in a JDBC result set. A red box labeled "커서의 위치" (Cursor Position) has a red arrow pointing to the first row of a table. The table has three columns, all labeled "필드명3" (Field Name 3), and two rows of data, each containing "XXX".

필드명3	필드명3	필드명3
XXX	XXX	XXX
XXX	XXX	XXX

```
ResultSet rs = pstmt.executeQuery();

while(rs.next()) {

    name = rs.getString(1); // or rs.getString("name");

    age = rs.getInt(2); // or rs.getInt("email");

}

rs.close();
```

# RecordSet 객체

name      userid      pwd      email      phone      admin

↓      ↓      ↓      ↓      ↓      ↓

Begin Of File:  
Before the First Row

이소미	somi	1234	gmd@naver.com	010-2362-5157	0
하상오	sang12	1234	ha12@naver.com	010-5629-8888	1
김윤승	light	1234	youn1004@naver.com	010-9999-8282	0

End Of File(EOF)  
After the Last Row

Begin Of File((BOF) : Before the First Row					
이소미	somi	1234	gmd@naver.com	010-2362-5157	0
하상오	sang12	1234	ha12@naver.com	010-5629-8888	1
김윤승	light	1234	youn1004@naver.com	010-9999-8282	0
End Of File(EOF) : After the Last Row					

코드 셋 객체가  
언어지자 마자 Cursor  
위치

rs.next( )

# RecordSet 객체

The diagram illustrates how RecordSet methods are used to access data from a table. Red arrows point from method calls to specific columns or rows in the table.

name	userid	pwd	email	phone	admin
이소미	somi	1234	gmd@naver.com	010-2362-5157	0
하상오	sang12	1234	ha12@naver.com	010-5629-8888	1
김윤승	light	1234	youn1004@naver.com	010-9999-8282	0

Methods and their corresponding data access:

- `rs.getString("name")` points to the **name** column.
- `rs.getString("userid")` points to the **userid** column.
- `rs.getString("pwd")` points to the **pwd** column.
- `rs.getString("email")` points to the **email** column.
- `rs.getString("phone")` points to the **phone** column.
- `rs.getInt("admin")` points to the **admin** column.
- `rs.getString(1)` points to the first row (이소미).
- `rs.getString(2)` points to the second row (하상오).
- `rs.getString(3)` points to the third row (김윤승).
- `rs.getString(4)` points to the first row (이소미).
- `rs.getString(5)` points to the second row (하상오).
- `rs.getInt(b)` points to the **admin** column.



# JDBC를 사용한 JSP와데이터베이스의 연동

---

- 6단계: 연결 해제

- 사용이 끝난 데이터베이스 연결은 `conn.close()` 메서드를 이용해 닫아 주도록 한다.
- DB 연결은 중요한 자원이고 제한적이므로 사용이 끝난 연결은 반드시 해제해 주어야 하므로 주의 하도록 한다.