



복습

[복습] MySQL 드라이버 로딩

signup_process.jsp

■ 사전 작업

- MySQL 드라이버 클래스 로딩

```
<%  
    String u_id = request.getParameter("userID");  
    String u_pw = request.getParameter("userPW");  
    String u_mail = request.getParameter("userMAIL");  
  
    String sql = "INSERT INTO members VALUES";  
    sql += "(" + u_id + "',' + u_pw + "',' + u_mail + "'";  
  
    Class.forName("com.mysql.jdbc.Driver");  
  
    String url = "jdbc:mysql://localhost:3306/jspdb";  
    String username = "root";  
    String password = "1234";  
  
    Connection conn = DriverManager.getConnection(url, username, password);  
    Statement sm = conn.createStatement();  
  
    int count = sm.executeUpdate(sql);  
    if(count == 1){  
        out.println("회원가입 성공!");  
    }else{  
        out.println("회원가입 실패!");  
    }  
    sm.close();  
    conn.close();  
%>
```

[복습] MySQL 데이터베이스 연동

■ 전체 과정 정리

데이터베이스 접속을 위해 DriverManager.getConnection()

① **Connection 객체 생성**

쿼리 실행 객체 생성을 위해 Connection 객체의 createStatement()
(Statement 객체 생성을 위해)

② **Statement 객체 생성**

③ Statement 객체를 이용한 쿼리 실행(**Statement 객체 활용**)

```
executeUpdate(String sql)
executeQuery(String sql)
```

[복습] MySQL 데이터베이스 연결

signup_process.jsp

- Connection 객체 생성

<%

```
String u_id = request.getParameter("userID");  
String u_pw = request.getParameter("userPW");  
String u_mail = request.getParameter("userMAIL");
```

```
String sql = "INSERT INTO members VALUES";  
sql += "(" + u_id + "', '" + u_pw + "', '" + u_mail + "'";
```

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

```
String url = "jdbc:mysql://localhost:3306/jspdb";  
String username = "root";  
String password = "1234";
```

```
Connection conn = DriverManager.getConnection(url, username, password);  
Statement sm = conn.createStatement();
```

:

☞ 다음과 같이 한 줄로 작성 가능

```
Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/jspdb", "root", "1234");
```

[복습] 쿼리 실행

signup_process.jsp

- Statement 객체 생성

- Statement 객체를 활용한

SQL문 실행 (`executeUpdate()`)

```
<%  
    String u_id = request.getParameter("userID");  
    String u_pw = request.getParameter("userPW");  
    String u_mail = request.getParameter("userMAIL");  
  
    String sql = "INSERT INTO members VALUES";  
    sql += "(" + u_id + "',' + u_pw + "',' + u_mail + "'";  
  
    Class.forName("com.mysql.jdbc.Driver");  
  
    String url = "jdbc:mysql://localhost:3306/jspdb";  
    String username = "root";  
    String password = "1234";  
  
    Connection conn = DriverManager.getConnection(url, username, password);  
    Statement sm = conn.createStatement();  
  
    int count = sm.executeUpdate(sql);  
    if(count == 1){  
        out.println("회원가입 성공!");  
    }else{  
        out.println("회원가입 실패!");  
    }  
    sm.close();  
    conn.close();  
%>
```

[복습] SQL문 실행

- Statement 객체를 이용한 쿼리 실행

- Connection 객체가 생성되면,

SQL문을 데이터베이스로 전송하기 위해 Connection 객체의 createStatement() 함수 실행

👉 Statement 객체 생성

👉 Statement 클래스가 제공하는 함수 (SQL문에 따라 실행함수가 달라짐)

메소드	반환 유형	설명
executeUpdate(String sql)	int	INSERT, DELETE, UPDATE 문을 실행하기 위한 함수이며, 실행 결과 변경된 레코드의 수를 반환
executeQuery(String sql)	java.sql.ResultSet	SELECT 문을 실행하기 위한 함수로, 실행 결과로 얻어진 테이블 형태(Java에서는 ResultSet으로 표현)의 데이터를 반환

```
int count = sm.executeUpdate(sql);
```

<%

memberList.jsp

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

```
Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/jspdb", "root", "1234");
```

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

```
ResultSet rs = sm.executeQuery("SELECT id, email FROM members");
```

```
out.print("Home > 회원가입 명단 <hr>");
```

Statement 객체를 활용한 SQL문 실행 (`executeQuery()`)

```
String str = "";
```

```
int count = 1;
```

```
while(rs.next()){  
    str = count + " : " + rs.getString("id") + " / " + rs.getString("email") + " / " + "<br>";  
    out.print(str);  
    count++;  
}
```

```
rs.close();  
sm.close();  
conn.close();
```

%>

	id	passwd	email
▶	hansung	1234	hansung@hansung.ac.kr
	hs	5678	hs@hansung.ac.kr
	jbj	1111	hansung@hansung.ac.kr

[복습] ResultSet 클래스

▪ ResultSet 클래스의 주요 함수

메소드	반환 유형	설명
next()	boolean	현재 레코드를 가리키는 커서 ^{Cursor} 를 다음 레코드로 이동시킵니다. 다음 레코드가 존재하여 이동이 성공할 경우에는 true를, 그렇지 않는 경우 False를 반환합니다.
getString(String column)	String	커서가 가리키는 레코드 내 파라미터로 주어진 column 값을 String 타입으로 반환합니다.
getInt(String column)	int	커서가 가리키는 레코드 내 파라미터로 주어진 column 값을 int 타입으로 반환합니다.
first()	boolean	커서를 첫번째 레코드로 이동합니다.
last()	boolean	커서를 마지막 레코드로 이동합니다.

[복습] 자원 반납

- 객체의 close() 함수 호출

- 마지막으로 데이터베이스 접근 과정에서 사용된 Connection, Statement, ResultSet 객체를 close() 함수를 이용해 모두 닫아주면 된다.
- 명시적으로 닫아주지 않을 경우 메모리에 객체들이 계속 남아있게 되어 불필요한 가비지 컬렉션 유발

```
rs.close();  
stmt.close();  
conn.close();
```

보다 편한 동적 쿼리문 작성

- 회원 탈퇴 프로그램 -

보다 편한 동적 쿼리문 작성

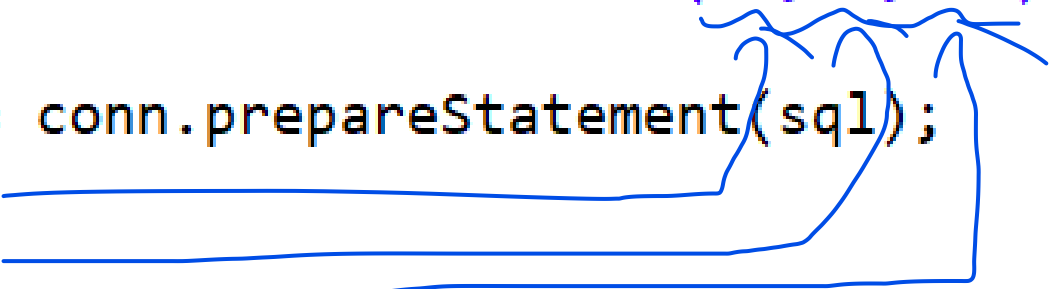
- 다음은 System.out.println()을 이용해 콘솔창에 출력해가면서 만들 수는 있으나
너무 복잡하고, 불편하며, 실수 가능성이 높음

~~String sql = "INSERT INTO members VALUES" + "(" + u_id + "," + u_pw + "," + u_mail + "," + u_gender + ")";~~

이제!! 쓰지마

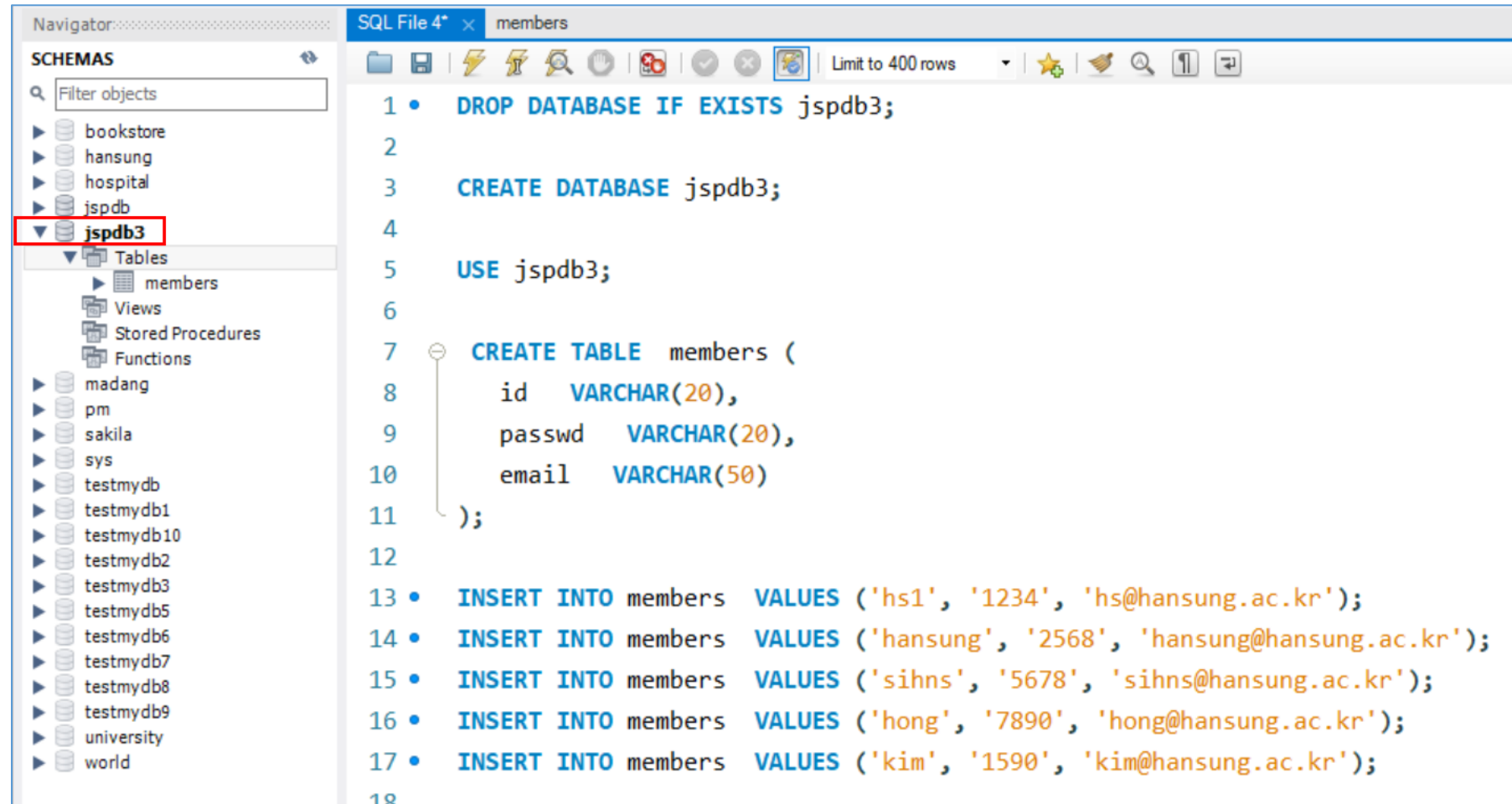
- 따라서 PreparedStatement 객체를 사용하면 보다 효율적으로 작성 가능

```
String sql = "INSERT INTO members VALUES ( ?, ?, ? )";  
  
PreparedStatement sm = conn.prepareStatement(sql);  
sm.setString(1, u_id);  
sm.setString(2, u_pw);  
sm.setString(3, u_mail);
```



DB 준비

- 실습 파일(올려드린 jspdb3.txt)



Navigator: SCHEMAS

Filter objects

- bookstore
- hansung
- hospital
- jspdb
- jspdb3**
- madang
- pm
- sakila
- sys
- testmydb
- testmydb1
- testmydb10
- testmydb2
- testmydb3
- testmydb5
- testmydb6
- testmydb7
- testmydb8
- testmydb9
- university
- world

Tables

- members
- Views
- Stored Procedures
- Functions

SQL File 4* x members

Limit to 400 rows

```
1 • DROP DATABASE IF EXISTS jspdb3;
2
3 CREATE DATABASE jspdb3;
4
5 USE jspdb3;
6
7 CREATE TABLE members (
8     id VARCHAR(20),
9     passwd VARCHAR(20),
10    email VARCHAR(50)
11 );
12
13 • INSERT INTO members VALUES ('hs1', '1234', 'hs@hansung.ac.kr');
14 • INSERT INTO members VALUES ('hansung', '2568', 'hansung@hansung.ac.kr');
15 • INSERT INTO members VALUES ('sihns', '5678', 'sihns@hansung.ac.kr');
16 • INSERT INTO members VALUES ('hong', '7890', 'hong@hansung.ac.kr');
17 • INSERT INTO members VALUES ('kim', '1590', 'kim@hansung.ac.kr');
18
```

	id	passwd	email
▶	hs1	1234	hs@hansung.ac.kr
	hansung	2568	hansung@hansung.ac.kr
	sihns	5678	sihns@hansung.ac.kr
	hong	7890	hong@hansung.ac.kr
	kim	1590	kim@hansung.ac.kr

회원 탈퇴 프로그램

- [ch07] 폴더에 withdraw.jsp 파일 생성 및 실행 결과 확인

회원 탈퇴

Home > Withdraw the Membership

회원 탈퇴

아이디 :

탈퇴하기

회원 탈퇴 프로그램

withdraw.jsp

- 소스코드 withdraw.jsp
(올려드린 withdraw.txt)

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title> 회원 탈퇴 </title>
</head>
<body>
    Home > Withdraw the Membership
    <hr>
    <form action="withdraw_process.jsp" name="user_info" method="post">
        <fieldset style="width:200px">
            <legend> 회원 탈퇴 </legend><p>

            아이디 : <br>
            <input type="text" name="userID"><br>

            <div align="center">
                <input type="submit" value=" 탈퇴하기 "> &nbsp;&nbsp;&nbsp;
            </div><br>
        </fieldset>
    </form>
</body>
</html>
```

회원 탈퇴 프로그램

- [ch07] 폴더에 *withdraw_process.jsp* 파일 생성 및 실행 결과 확인

Home > Withdraw the Membership

회원 탈퇴

아이디:

탈퇴하기

회원 탈퇴 성공!

유니스트

	id	passwd	email
▶	hs1	1234	hs@hansung.ac.kr
	hansung	2568	hansung@hansung.ac.kr
	sihns	5678	sihns@hansung.ac.kr
	hong	7890	hong@hansung.ac.kr
	kim	1590	kim@hansung.ac.kr

Result Grid | Filter Rows:

	id	passwd	email
▶	hansung	2568	hansung@hansung.ac.kr
	sihns	5678	sihns@hansung.ac.kr
	hong	7890	hong@hansung.ac.kr
	kim	1590	kim@hansung.ac.kr

withdraw.jsp

withdraw_process.jsp

회원 탈퇴 프로그램

■ 소스코드 withdraw_process.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ page import="java.sql.*" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title> 회원 탈퇴 </title>
</head>
<body>

<%
    Class.forName("com.mysql.jdbc.Driver");
    Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/jspdb3", "root", "1234");

    String u_id = request.getParameter("userID");
    String sql = "DELETE FROM members WHERE id = ?";

    PreparedStatement sm = conn.prepareStatement(sql);
    sm.setString(1, u_id);

    int count = sm.executeUpdate();

    if(count == 1){
        out.print("회원 탈퇴 성공!");
    }else{
        out.print("회원 탈퇴 실패!");
    }

    sm.close();
    conn.close();
%>

</body>
</html>
```


회원 탈퇴 프로그램

withdraw_process.jsp

- 스크립틀릿 <% ... %> 내부

<%

```
Class.forName("com.mysql.jdbc.Driver");  
Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/jspdb3", "root", "1234");
```

```
String u_id = request.getParameter("userID");  
String sql = "DELETE FROM members WHERE id = ?";
```

```
( PreparedStatement sm = conn.prepareStatement(sql);  
  sm.setString(1, u_id);
```

```
int count = sm.executeUpdate();
```

```
if(count == 1){  
    out.print("회원 탈퇴 성공!");  
}else{  
    out.print("회원 탈퇴 실패!");  
}
```

```
sm.close();  
conn.close();
```

%>

회원 탈퇴 프로그램

■ PreparedStatement 객체(클래스)

- 동적인 쿼리에 사용
- Prepared Statement 객체는 동일한 쿼리문을 특정 값만 바꾸어서 여러 번 실행해야 할 때, 매개변수가 많아서 쿼리문을 정리해야 할 때 유용
- PreparedStatement 클래스는 Statement 클래스를 상속받기 때문에 Statement 클래스 안의 모든 메소드 사용 가능

메소드	반환 유형	설명
executeUpdate(String sql)	int	INSERT, DELETE, UPDATE 문을 실행하기 위한 함수이며, 실행 결과 변경된 레코드의 수를 반환
executeQuery(String sql)	java.sql.ResultSet	SELECT 문을 실행하기 위한 함수로, 실행 결과로 얻어진 테이블 형태(Java에서는 ResultSet으로 표현)의 데이터를 반환

- 추가적으로 다음 메소드 등 제공 (setXxx() 형태의 메소스)

메소드	반환 유형	설명
setString(int parameterIndex, String x)	void	필드 유형이 문자열인 경우
setInt(int parameterIndex, int x)	void	필드 유형이 정수형인 경우
setDate(int parameterIndex, Date x)	void	필드 유형이 날짜형인 경우

```
String u_id = request.getParameter("userID");  
String u_pw = request.getParameter("userPW");  
String u_mail = request.getParameter("userMAIL");
```

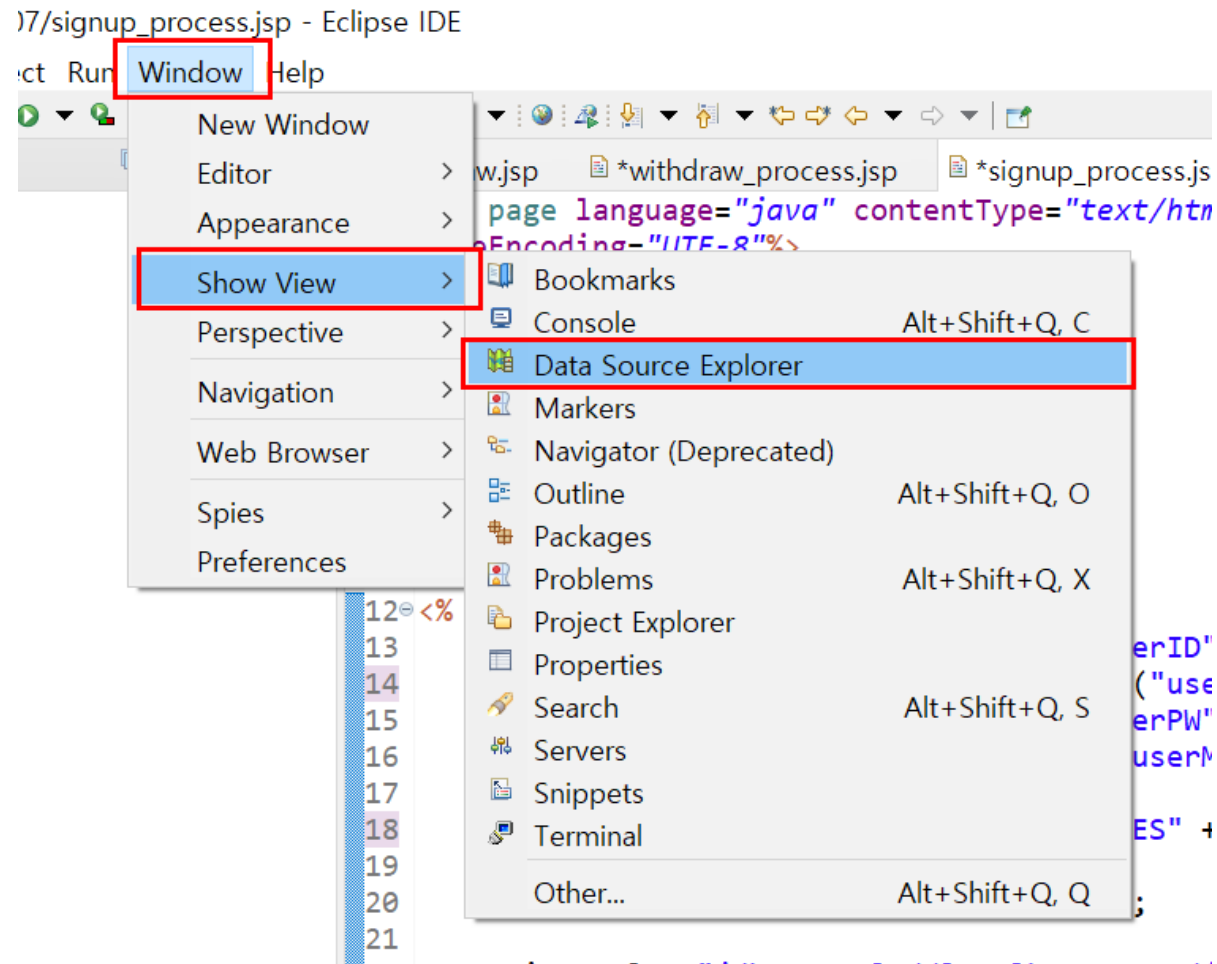
```
String sql = "INSERT INTO members VALUES ( ?, ?, ? )";
```

```
PreparedStatement sm = conn.prepareStatement(sql);  
sm.setString(1, u_id);  
sm.setString(2, u_pw);  
sm.setString(3, u_mail);
```

[참고] 커넥션 설정

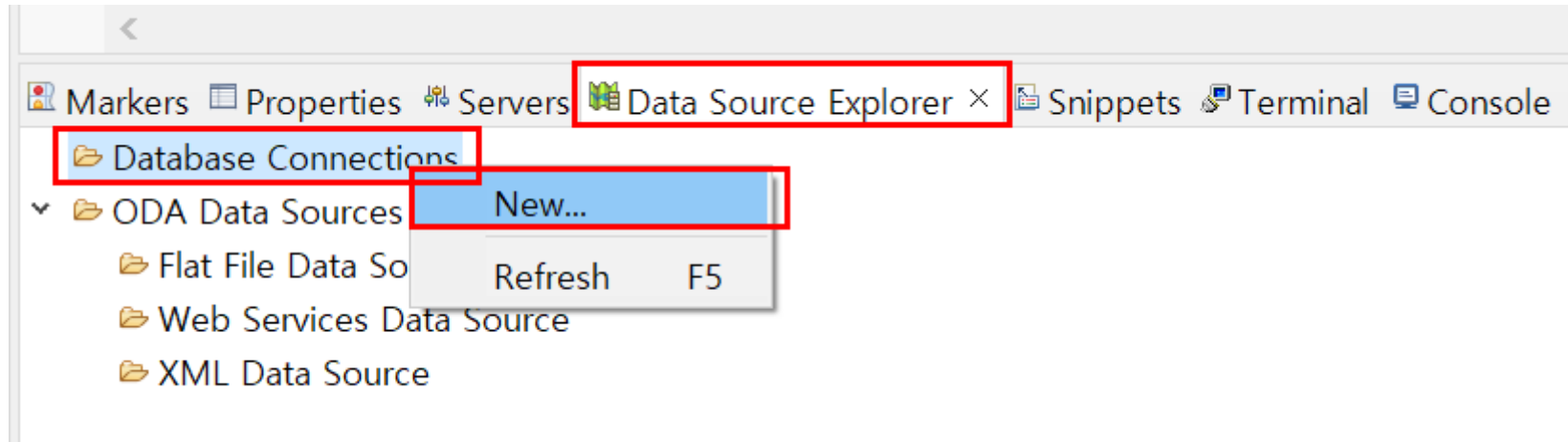
커넥션 설정

■ 이클립스에서 MySQL 제어



커넥션 설정

- 새로운 데이터베이스 커넥션 메뉴 선택



커넥션 설정

- 데이터베이스 커넥션 유형 선택

New Connection Profile

Connection Profile

Create a MySQL connection profile.

Connection Profile Types:

type filter text

- Informix
- Ingres
- MaxDB
- MySQL**
- Oracle
- PostgreSQL
- SQL Server
- SQLite
- Sybase ASA

Name:

New MySQL

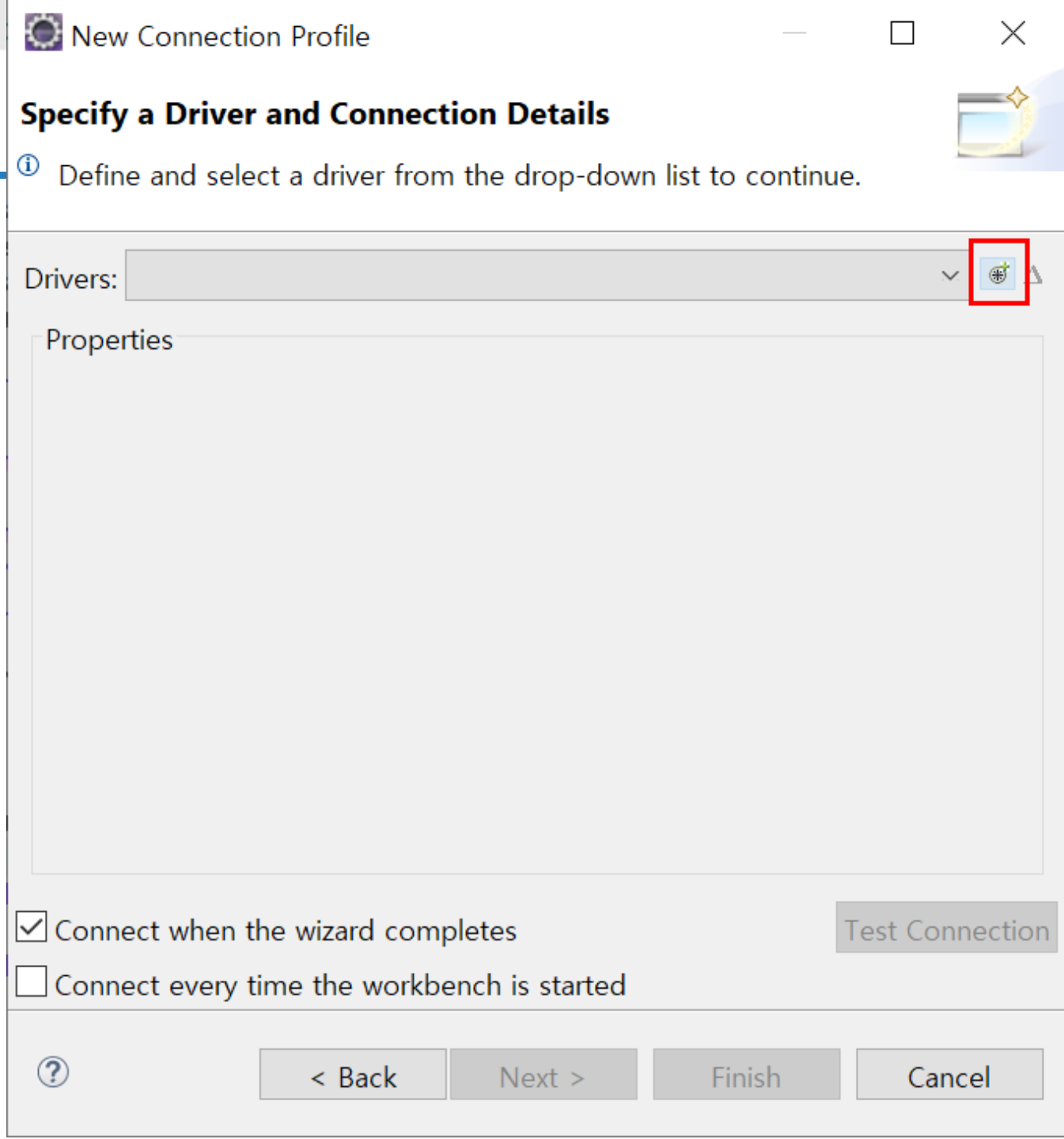
Description (optional):

?

< Back **Next >** Finish Cancel

커넥션 설정

- 새로운 드라이버 추가 버튼 클릭



커넥션 설정

New Driver Definition

Specify a Driver Template and Definition Name

✖ Unable to locate JAR/zip in file system as specified by the driver definition:
mysql-connector-java-5.1.0-bin.jar.

Name/Type JAR List Properties

Available driver templates:

Name ^	System Vendor	System V...
▼ Database		
MySQL JDBC Drive MySQL		4.0
MySQL JDBC Drive MySQL		4.1
MySQL JDBC Drive MySQL		5.0
MySQL JDBC Drive MySQL		5.1

Driver name:

MySQL JDBC Driver

Driver type:

MySQL JDBC Driver

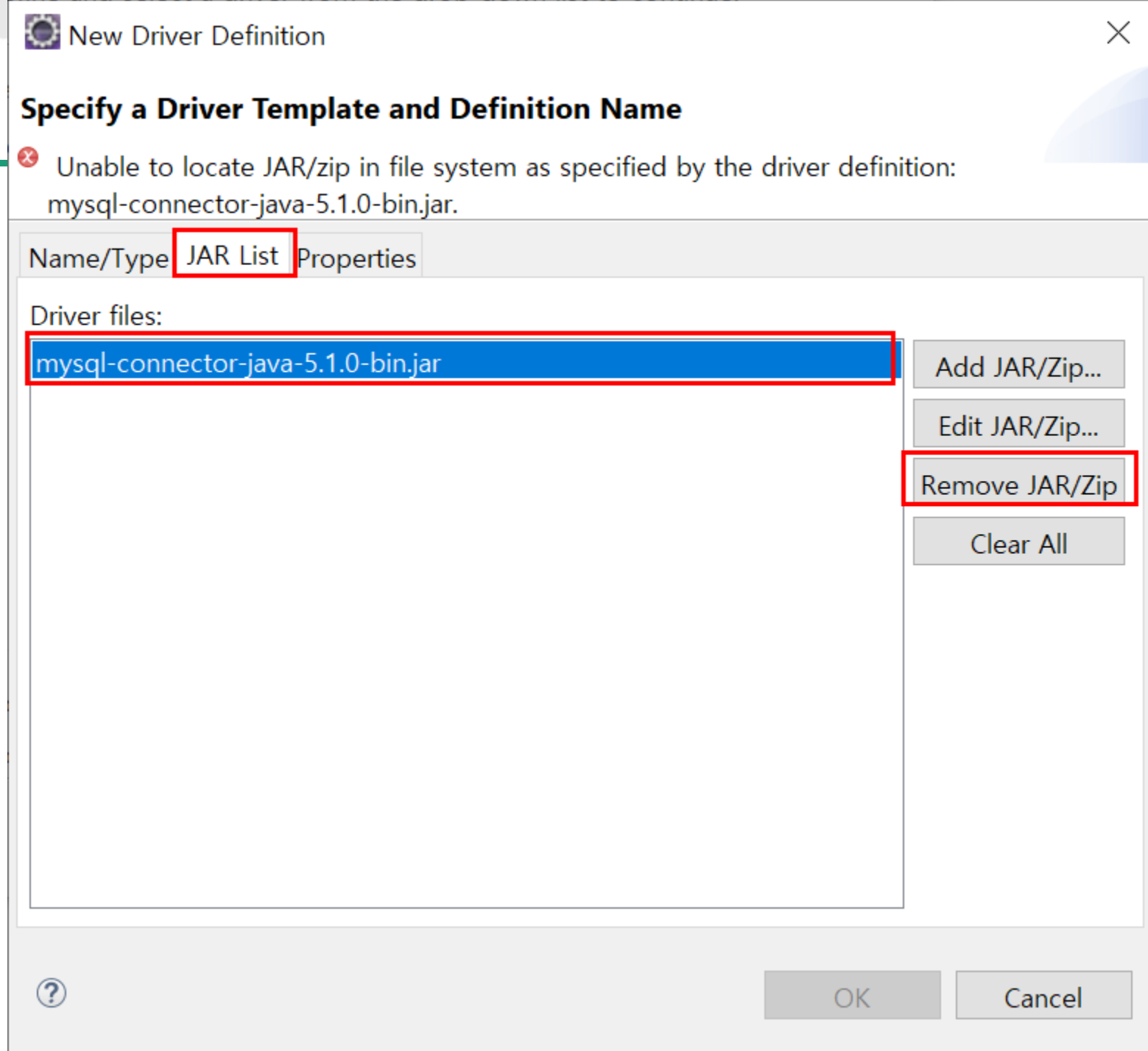


OK

Cancel

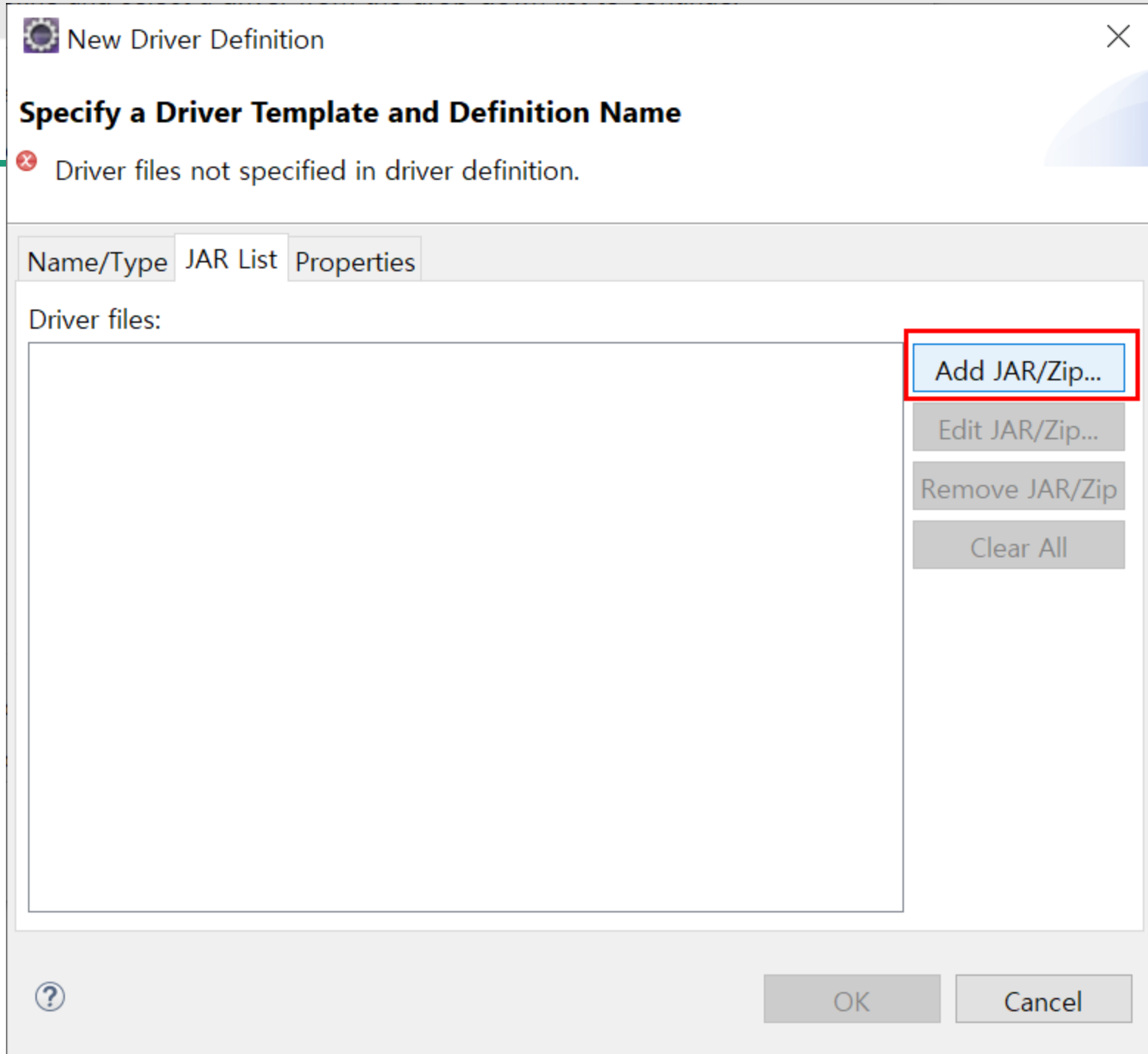
커넥션 설정

- 기존 예시 드라이버 삭제

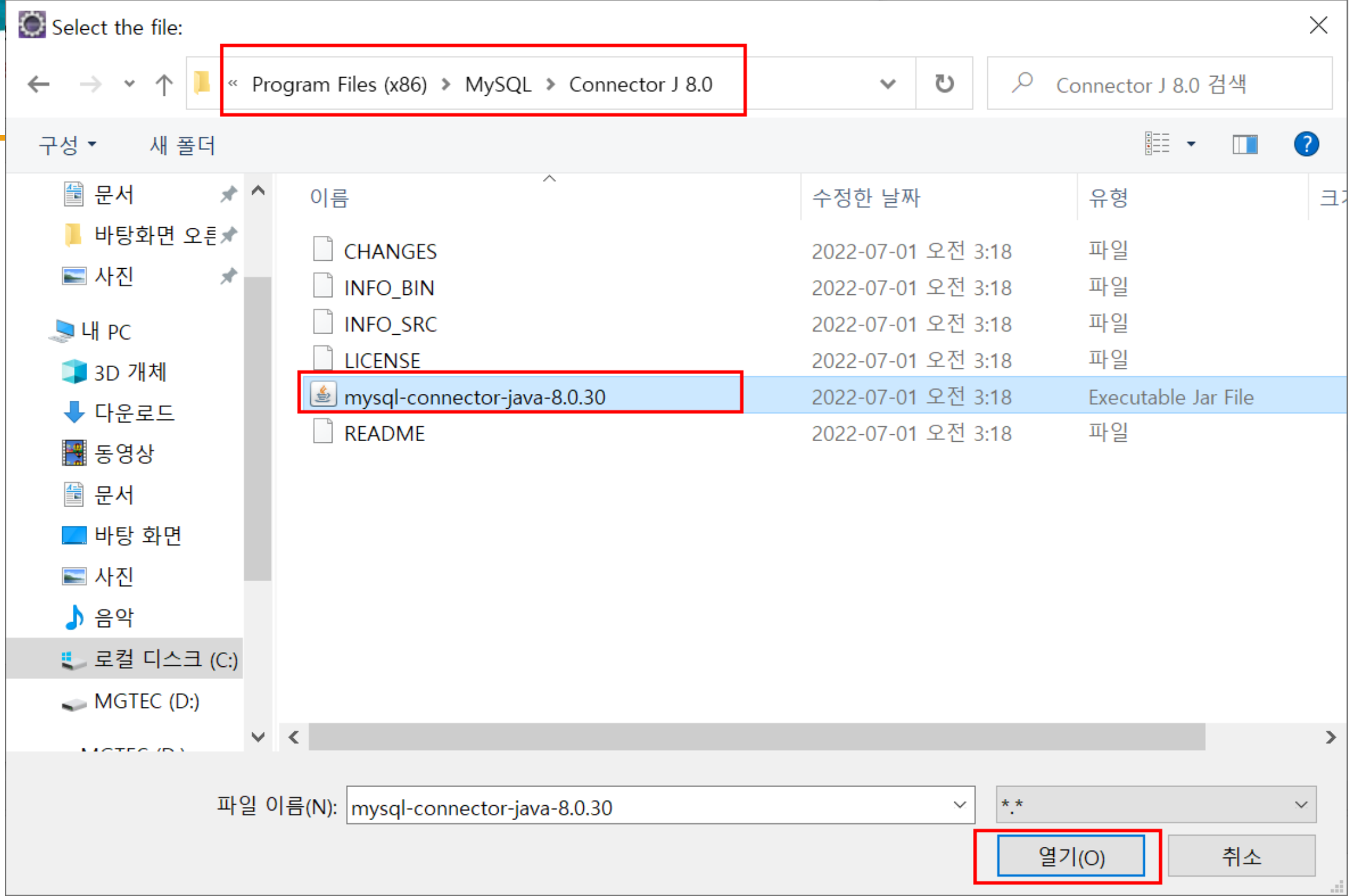


커넥션 설정

- 드라이버 추가



커넥션



커넥션 설정

Specify a Driver Template and Definition Name

Specify a driver template, then modify details in the fields below to provide a unique name, a list of required jars, and set any available and applicable property

Name/Type JAR List Properties

Driver files:

C:\Program Files (x86)\MySQL\Connector J 8.0\mysql-connector-java-

Add JAR/Zip...

Edit JAR/Zip...

Remove JAR/Zip

Clear All



OK

Cancel

커넥션 설정



New Driver Definition



Specify a Driver Template and Definition Name

Specify a driver template, then modify details in the fields below to provide a unique name, a list of required jars, and set any available and applicable property

Name/Type	JAR List	Properties
-----------	----------	------------

Properties:

Property	Value
▼ General	
Connection URL	jdbc:mysql://localhost:3306/jspdb3
Database Name	jspdb3
Driver Class	com.mysql.jdbc.Driver
Password	●●●●
User ID	root

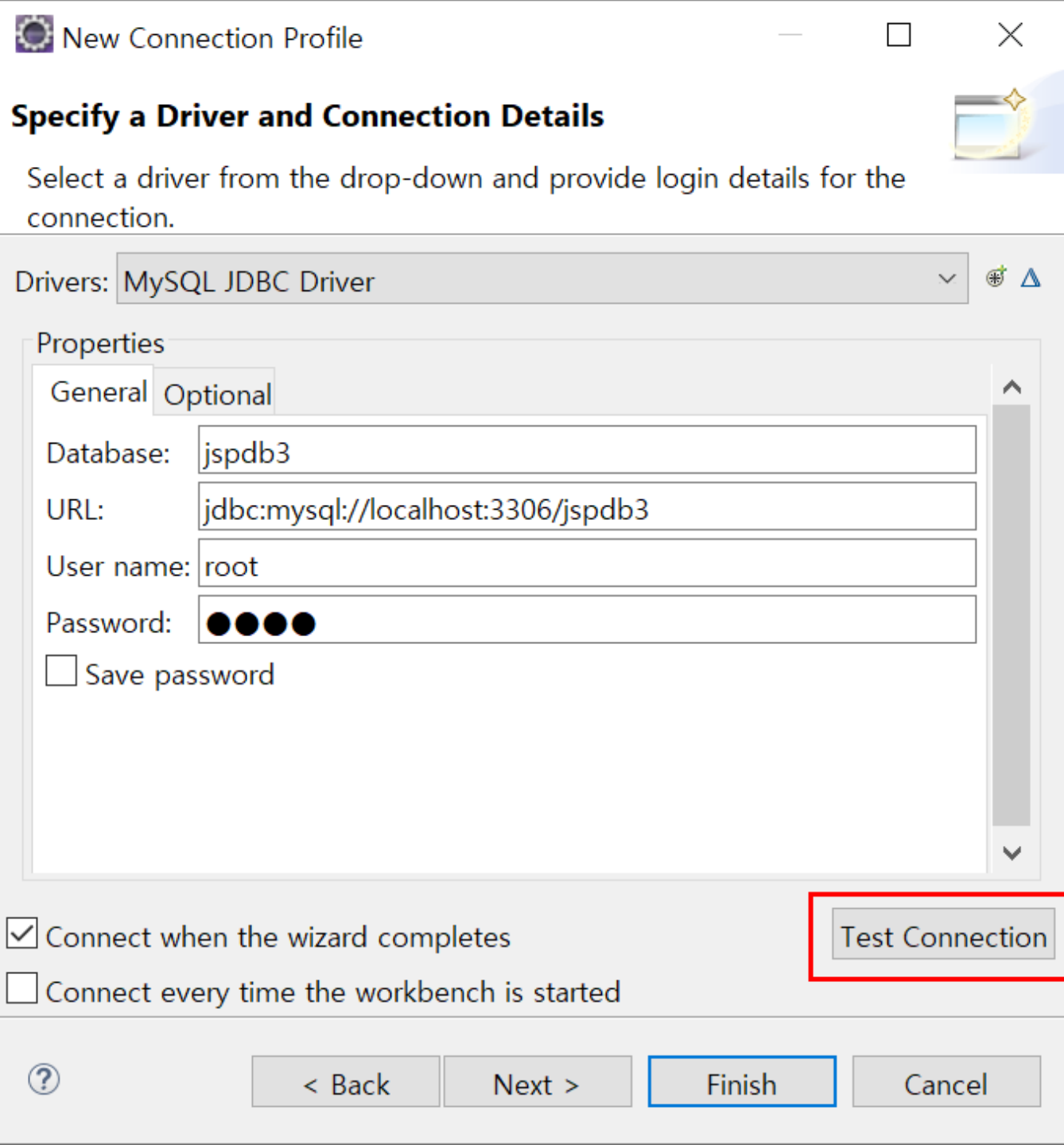
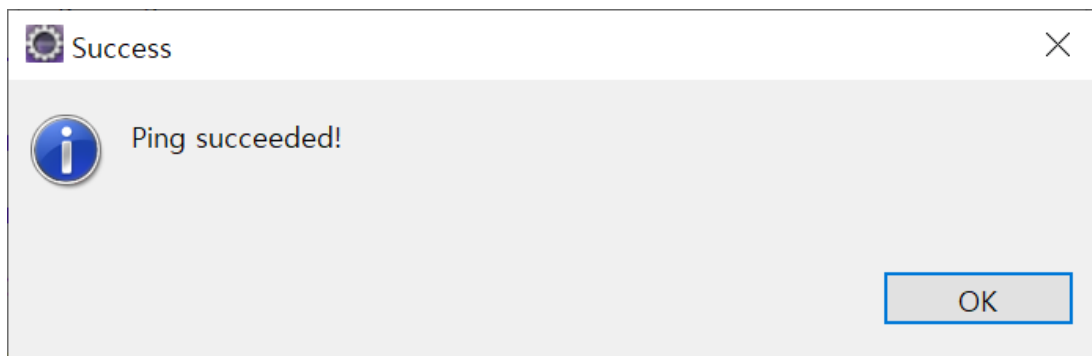


OK

Cancel

커넥션 설정

■ 연결 테스트



커넥션 설정

New Connection Profile

Specify a Driver and Connection Details

Select a driver from the drop-down and provide login details for the connection.

Drivers: MySQL JDBC Driver

Properties

General Optional

Database: jspdb3

URL: jdbc:mysql://localhost:3306/jspdb3

User name: root

Password: ●●●●

☐ Save password

☒ Connect when the wizard completes

☐ Connect every time the workbench is started

Test Connection



< Back

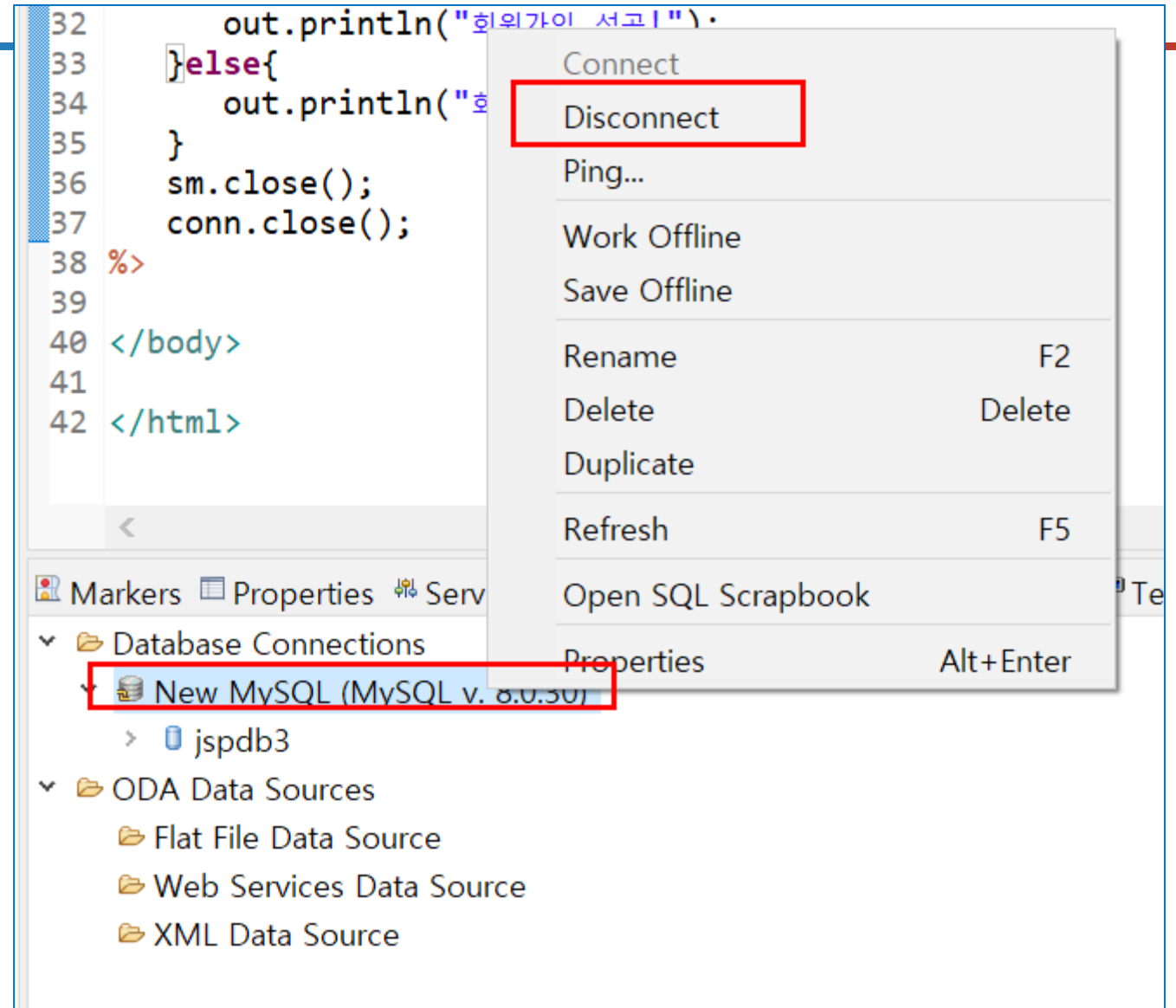
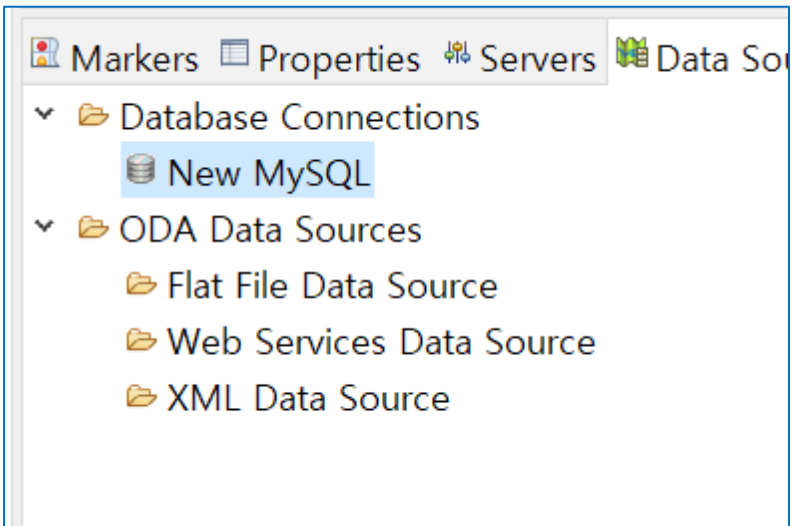
Next >

Finish

Cancel

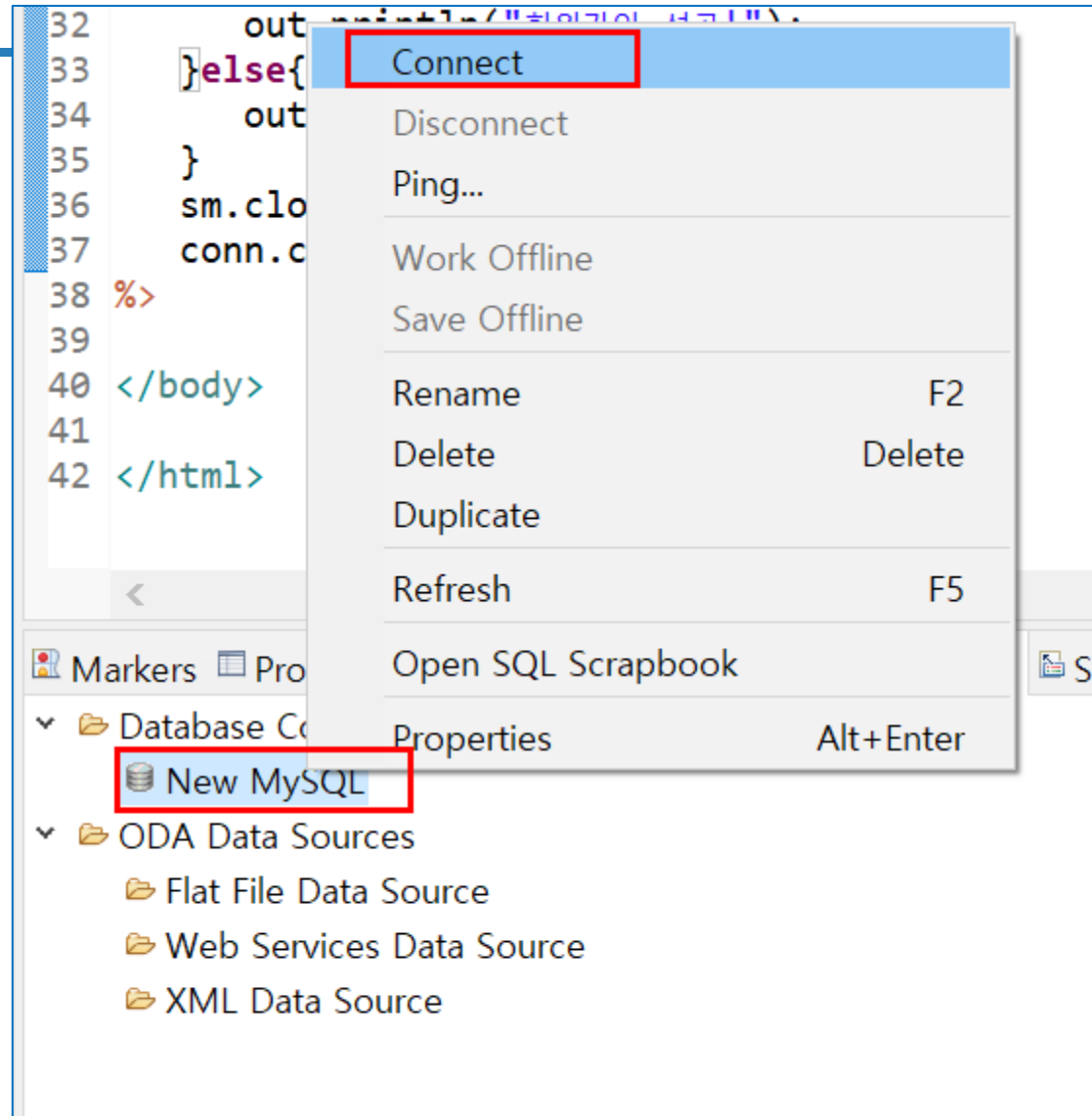
커넥션 설정

■ 연결 해제

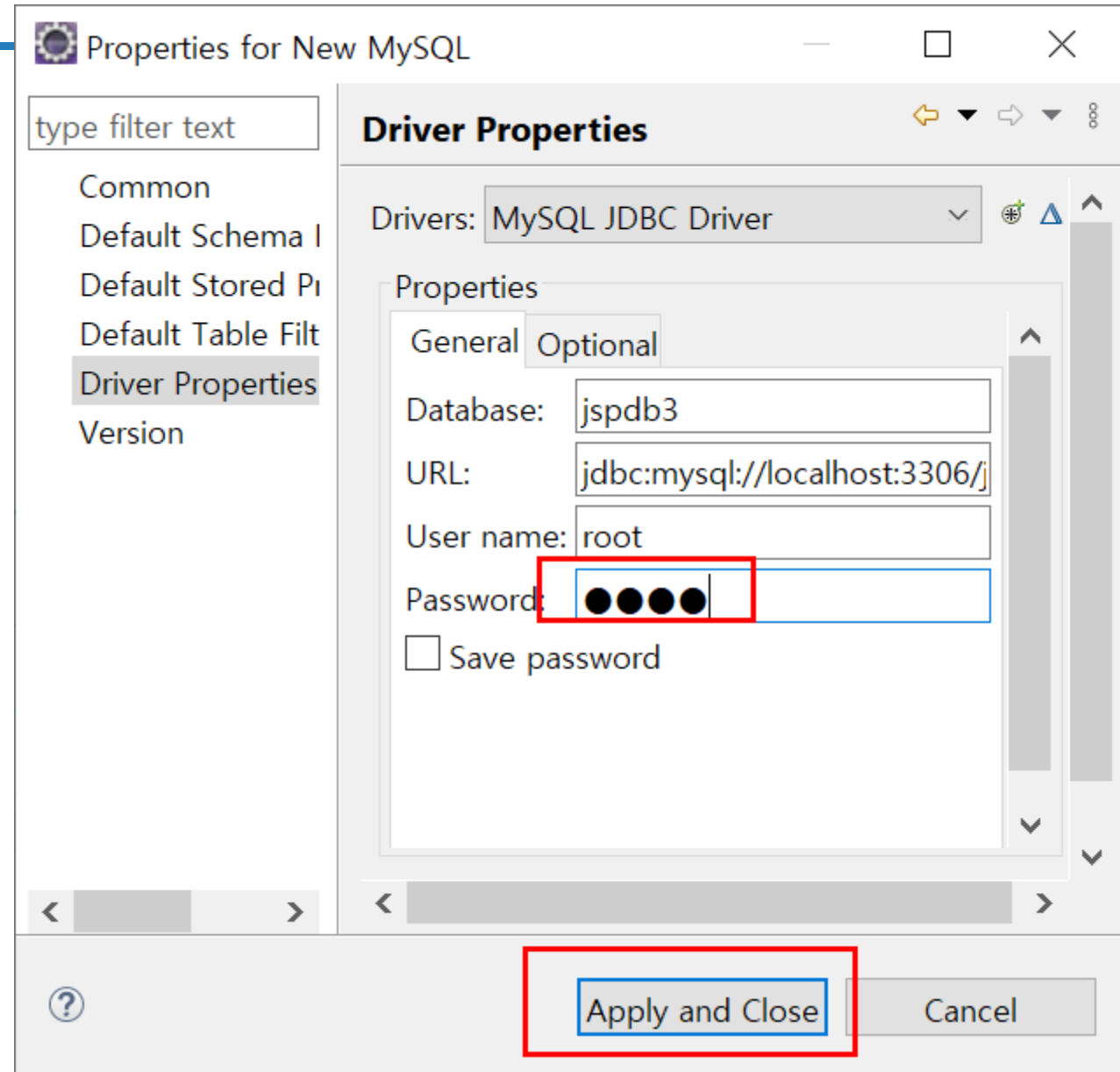


커넥션 설정

■ 연결

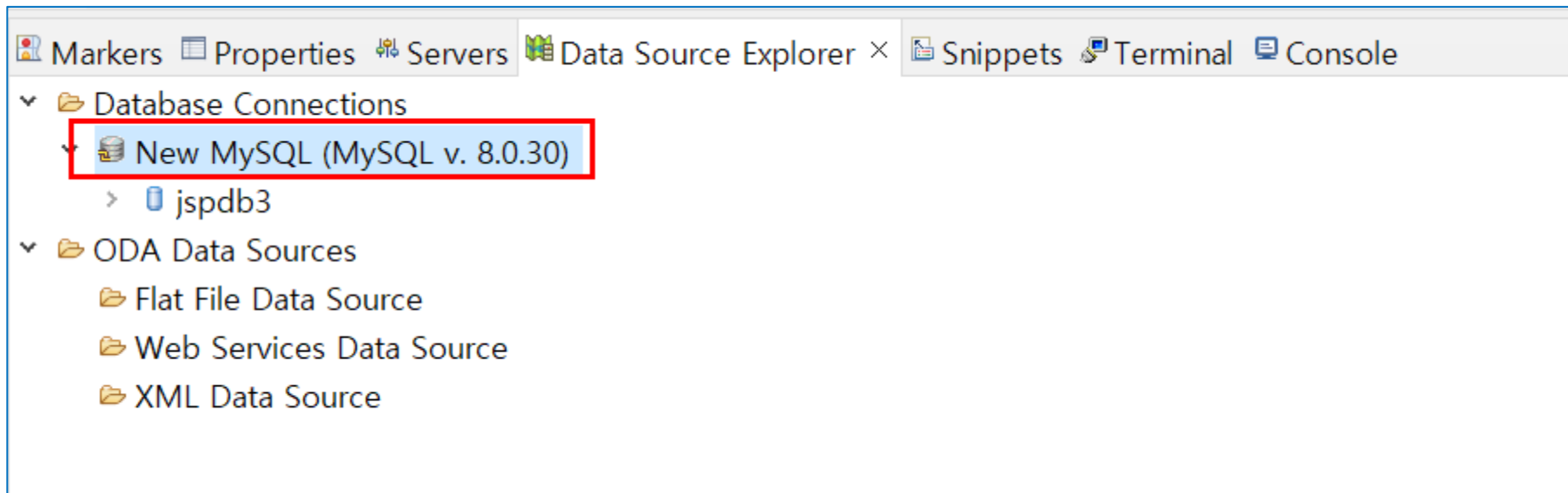


커넥션 설정



커넥션 설정

■ SQL 스크랩북 선택



커넥션 설정

■ 커넥션 속성 선택

Connection profile

Type: MySQL_5.1 Name: New MySQL Database: jspdb3 Status: Connected, Auto Commit

1

커넥션 설정

- 드래그해서 블록 지정 후
실행

Connection profile

Type: MySql_5.1

Name: New MySQL

```
1 DROP DATABASE IF EXISTS jspdb3;
2
3 CREATE DATABASE jspdb3;
4
5 USE jspdb3;
6
7 CREATE TABLE members (
8     id    VARCHAR(20),
9     passwd VARCHAR(20),
10    email  VARCHAR(50)
11 );
12
13 INSERT INTO members VALUES ('h
14 INSERT INTO members VALUES ('h
15 INSERT INTO members VALUES ('s
16 INSERT INTO members VALUES ('h
17 INSERT INTO members VALUES ('k
18
19
20
```

Undo Typing	Ctrl+Z
Cut	Ctrl+X
Copy	Ctrl+C
Paste	Ctrl+V
Toggle Comment	
Ctrl+/	
Execute All	Ctrl+Alt+X
Execute Selected Text	Alt+X
Execute Selected Text As One Statement	Alt+C
Execute Current Text	Alt+S
Save as Template...	
Edit in SQL Query Builder...	Alt+Q
Preferences...	
Set Connection Info	

Execute Selected SQL Statements

커넥션 설정

■ 실행 결과

Markers Properties Servers Data Source Explorer Snippets Terminal Console SQL Results ×

Type query expression here

Status	Operation	Date	Connection Profile
> ✓ Succeeded	Group Execution	2023. 5. 11. 오후 ...	New MySQL

Status

```
id VARCHAR(20),  
passwd VARCHAR(20),  
email VARCHAR(50)  
)  
INSERT INTO members VALUES ('hs1', '1234', 'hs@hansung.ac.kr')  
INSERT INTO members VALUES ('hansung', '2568', 'hansung@hansung.ac.kr')  
INSERT INTO members VALUES ('sihns', '5678', 'sihns@hansung.ac.kr')  
INSERT INTO members VALUES ('hong', '7890', 'hong@hansung.ac.kr')  
INSERT INTO members VALUES ('kim', '1590', 'kim@hansung.ac.kr')
```

커넥션 설정

- 실행 결과 확인

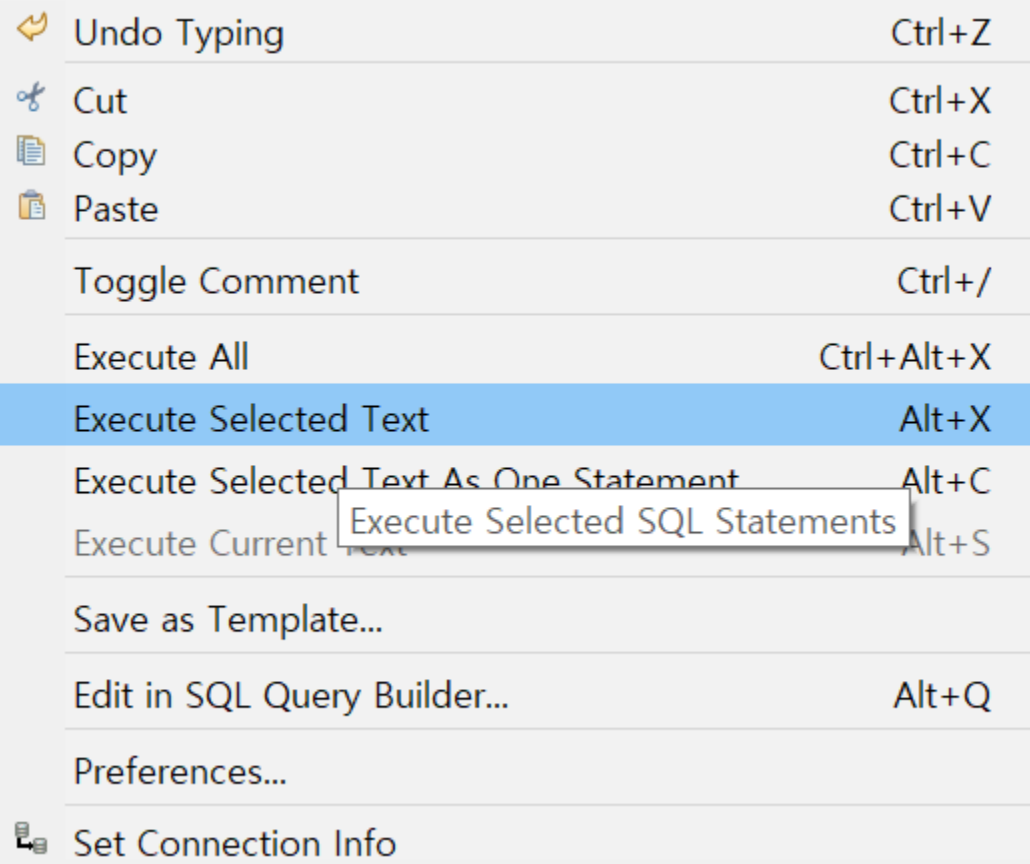
The screenshot displays a database management interface. The Navigator pane on the left shows a tree structure of databases and tables. The 'jspdb3' database is expanded, and the 'members' table is selected. The SQL Editor pane on the right shows a query: `SELECT * FROM jspdb3.members;`. The Result Grid pane at the bottom shows the query results in a table format.

	id	passwd	email
▶	hs1	1234	hs@hansung.ac.kr
	hansung	2568	hansung@hansung.ac.kr
	sihns	5678	sihns@hansung.ac.kr
	hong	7890	hong@hansung.ac.kr
	kim	1590	kim@hansung.ac.kr

커넥션 설정

■ 테이블 목록 확인

show tables;



Status	Result1
	Tables_in_jspdb3
1	members

커넥션 설정

■ 테이블 구조 확인

```
desc members;
```

Status	Result1					
	Field	Type	Null	Key	Default	Extra
1	id	varchar(20)	YES		NULL	
2	passwd	varchar(20)	YES		NULL	
3	email	varchar(50)	YES		NULL	



 **T h a n k y o u**

TECHNOLOGY

>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Velit ex
plicabo ipsum, labore sed tempora ratione asperiores des
cenderat bore sed tempora rati jgert one bore sed tem!