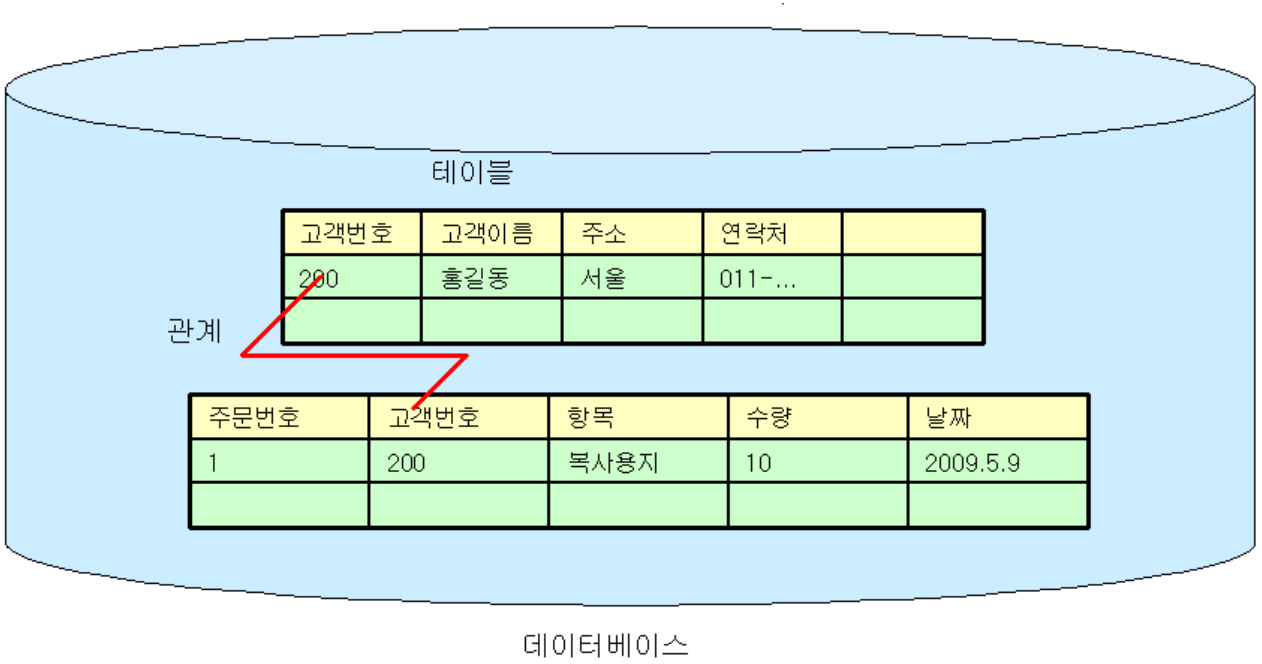
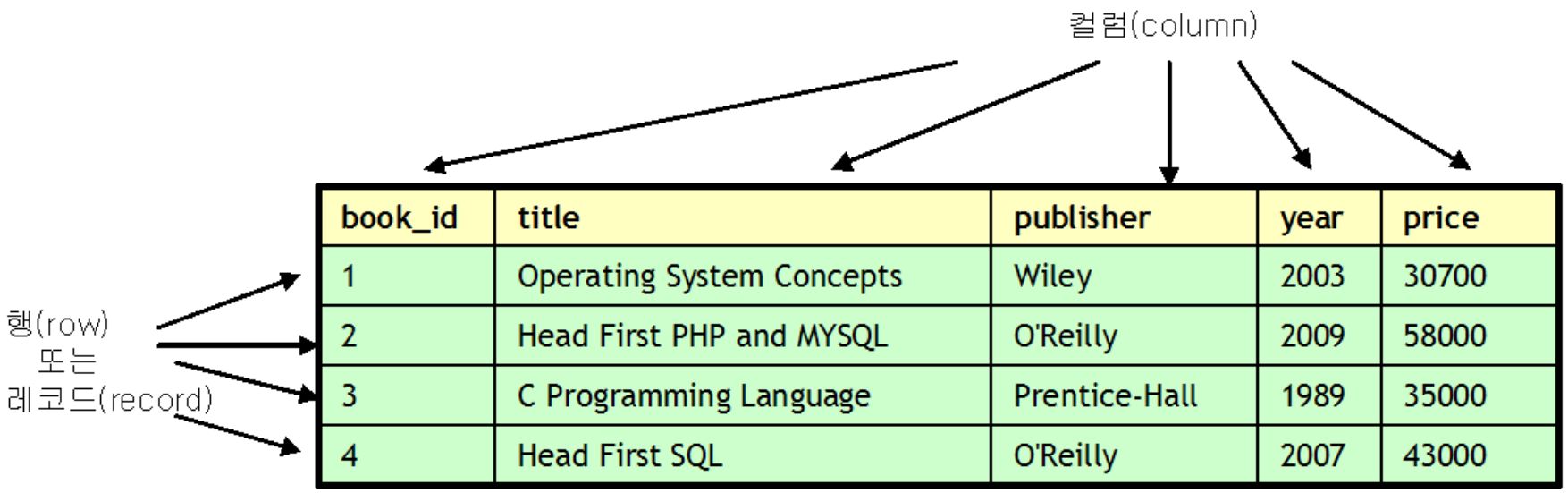
데이터베이스 프로그래밍

데이터베이스란?

- 관계형 데이터베이스(database)는 데이터를 여러 개의 테이블에 나누어서 저장한다.
- 가장 많이 사용되는 DBMS
  - 오라클, 마이크로소프트의 SQL Server, 사이베이스, MySQL



테이블



book_id	title	publisher	year	price
1	Operating System Concepts	Wiley	2003	30700
2	Head First PHP and MYSQL	O'Reilly	2009	58000
3	C Programming Language	Prentice-Hall	1989	35000
4	Head First SQL	O'Reilly	2007	43000

MariaDB 설치

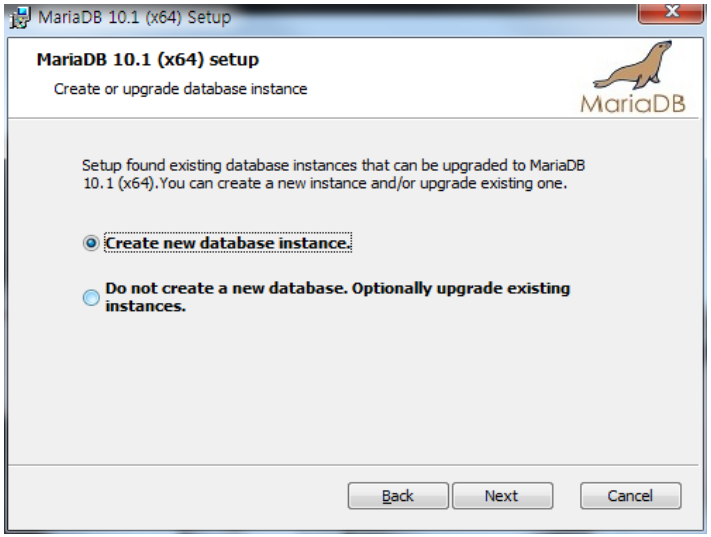
- https://downloads.mariadb.org/

MariaDB 설치하기

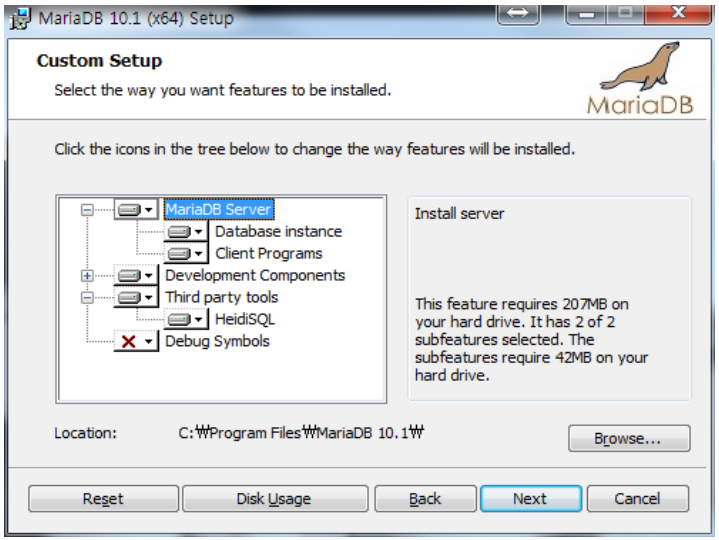
(1)

(2)

(3)

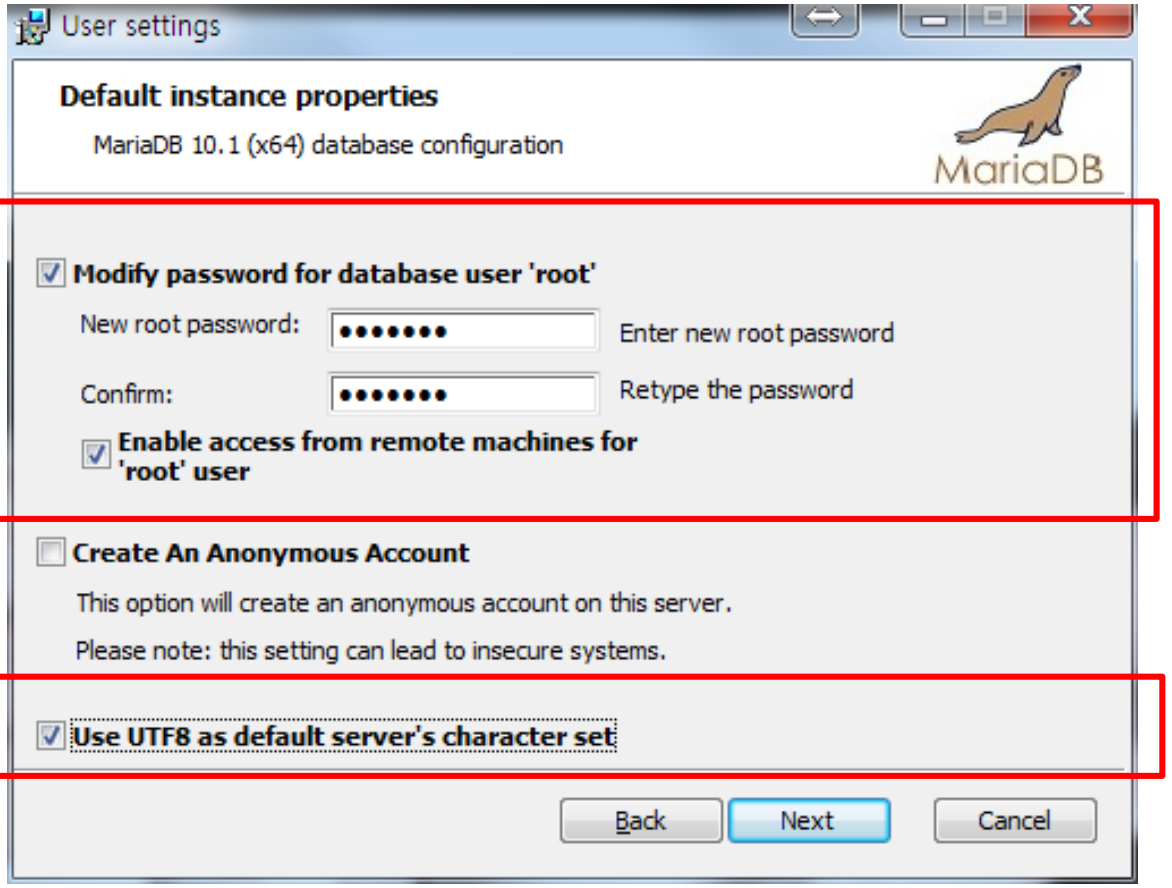


(4)

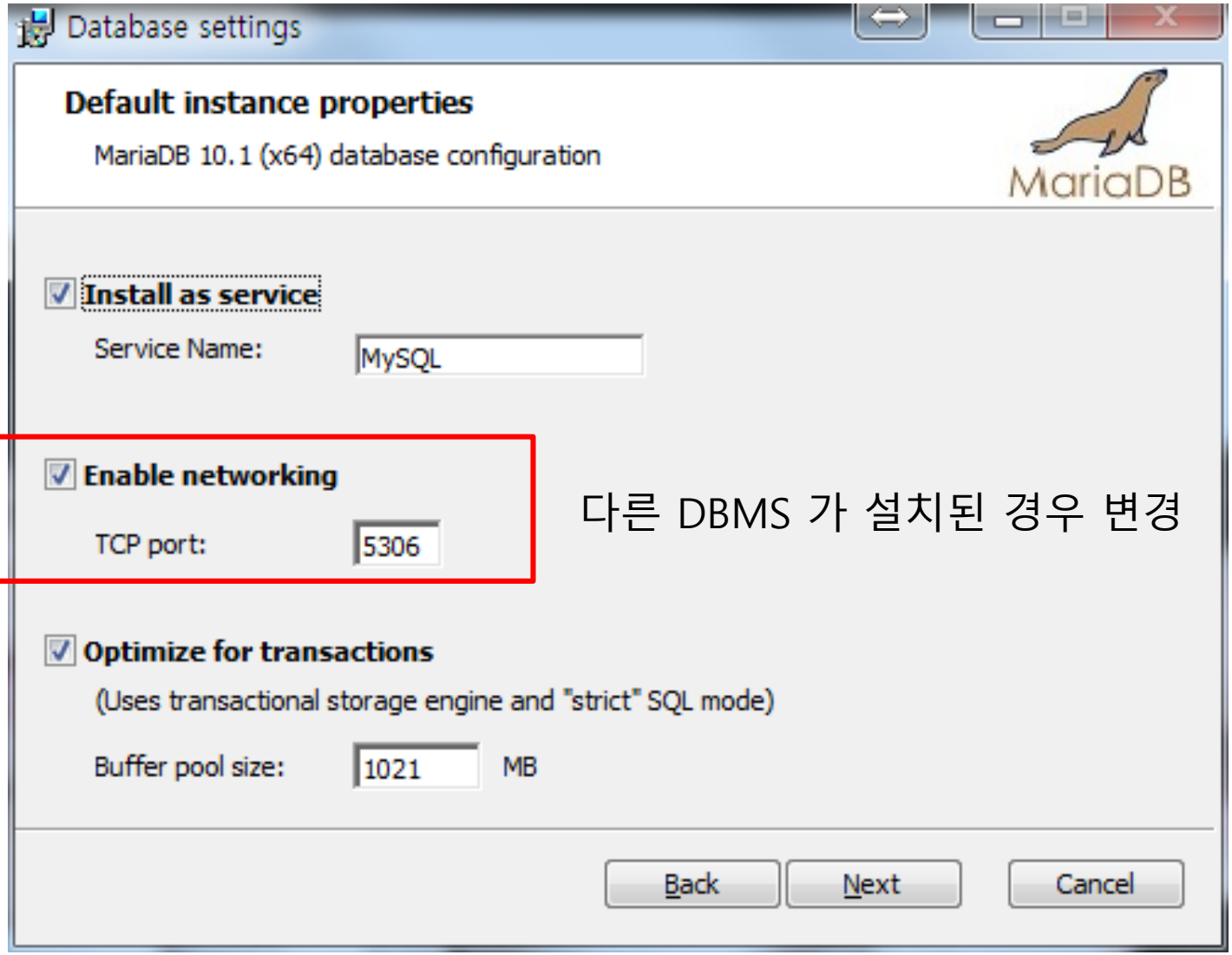


MariaDB 설치하기

- 비밀번호 설정 및 원격접속 허용

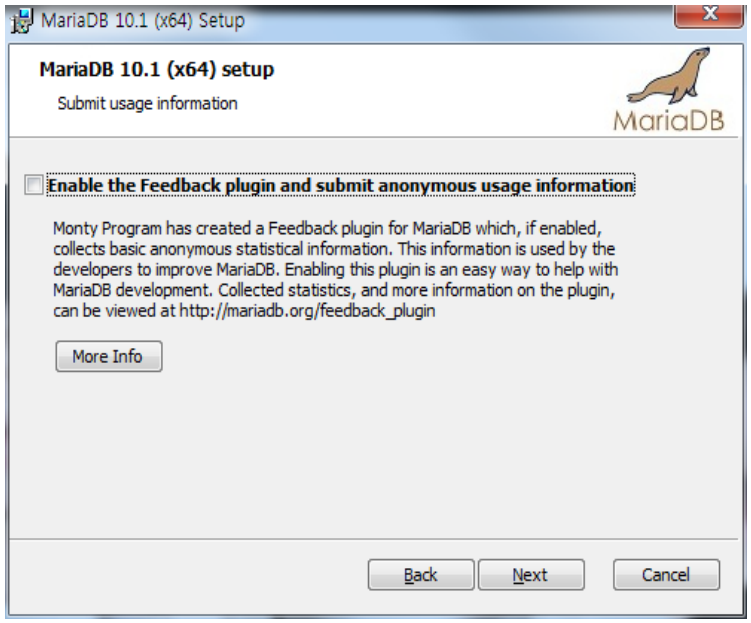


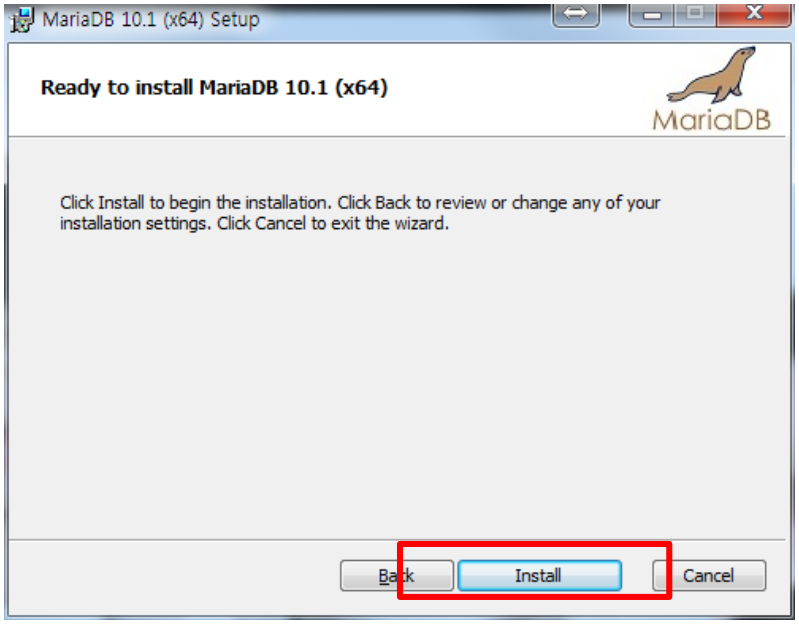
MariaDB 설치하기



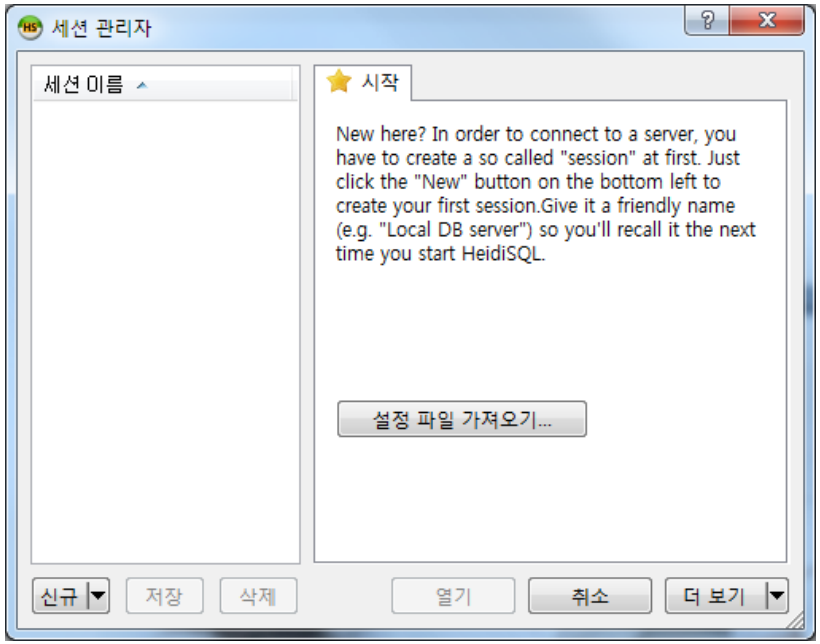
다른 DBMS 가 설치된 경우 변경

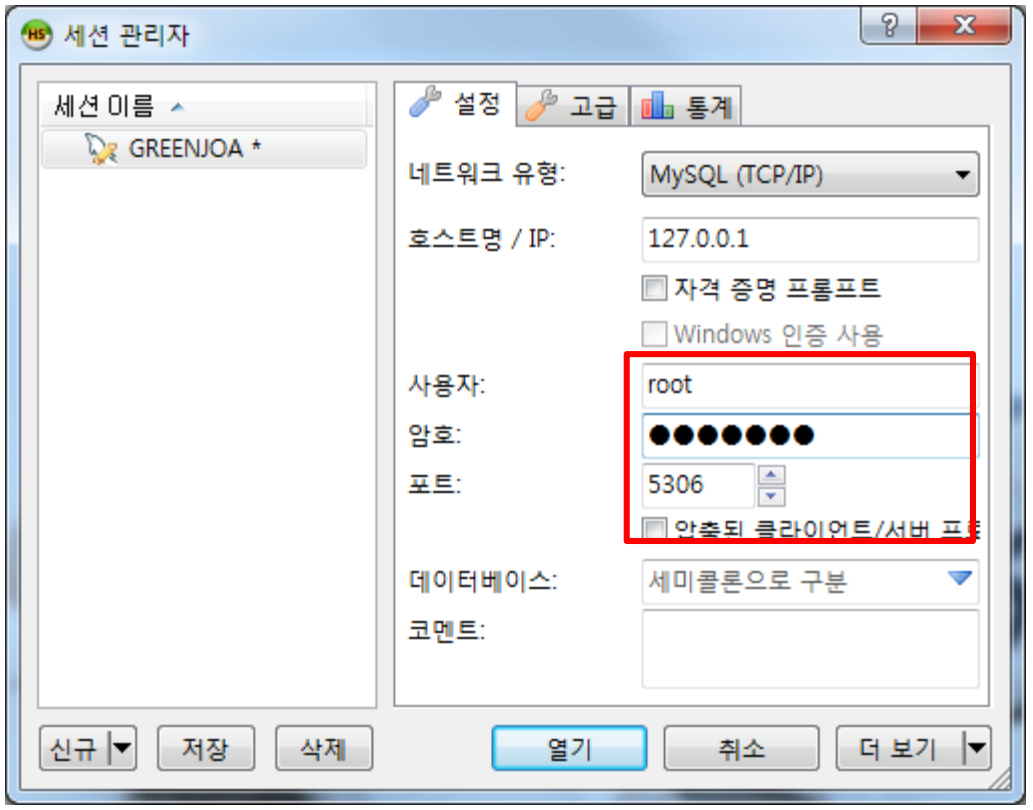
MariaDB 설치하기



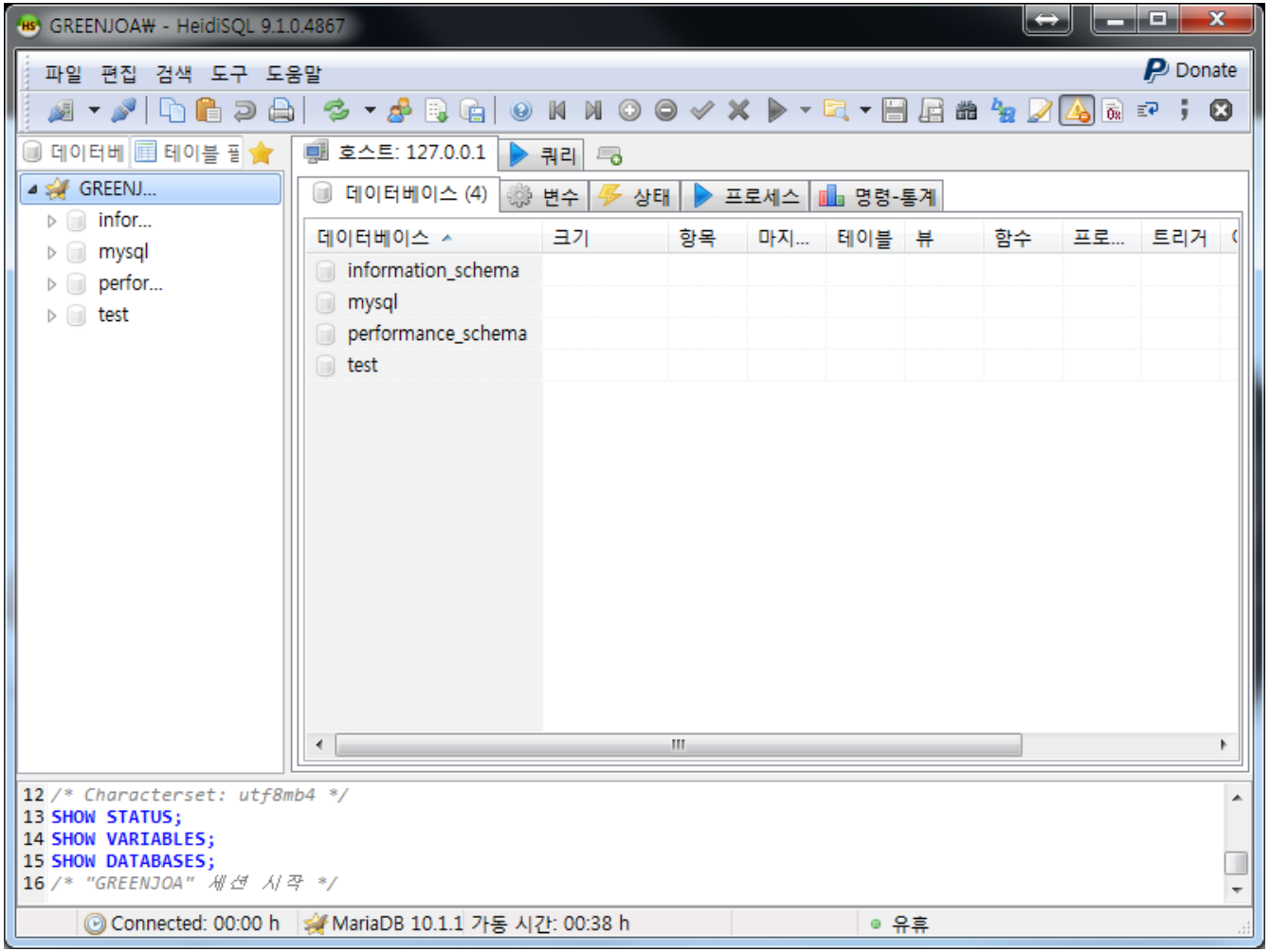


HeidiSQL


세션 관리자
세션 이름
시작
New here? In order to connect to a server, you have to create a so called "session" at first. Just click the "New" button on the bottom left to create your first session.Give it a friendly name (e.g. "Local DB server") so you'll recall it the next time you start HeidiSQL.
설정 파일 가져오기...
신규
저장
삭제
열기
취소
더 보기

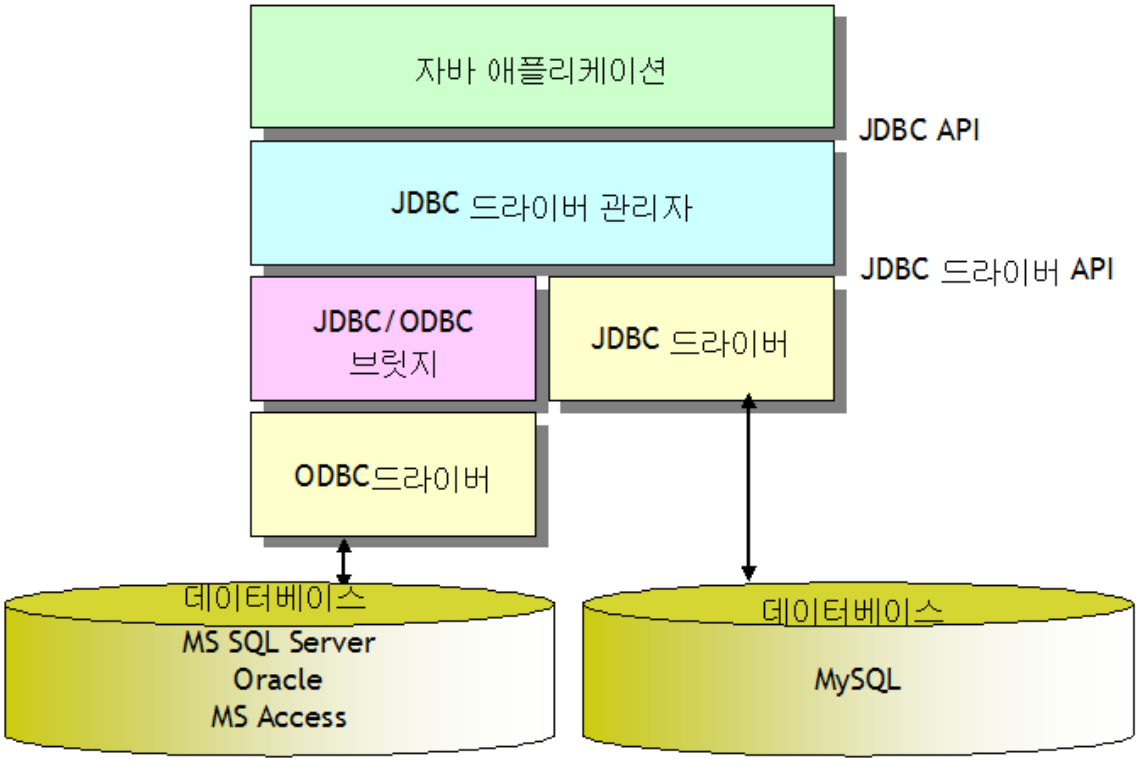

세션 관리자
세션 이름
GREENJOA *
설정
고급
통계
네트워크 유형:
MySQL (TCP/IP)
호스트명 / IP:
127.0.0.1
자격 증명 프롬프트
Windows 인증 사용
사용자:
root
암호:
●●●●●●●
포트:
5306
데이터베이스:
세미콜론으로 구분
코멘트:
신규
저장
삭제
열기
취소
더 보기

HeidiSQL


GREENJOA₩ - HeidiSQL 9.1.0.4867
파일 편집 검색 도구 도움말
Donate
데이터베
테이블 필
GREENJ...
infor...
mysql
perfor...
test
호스트: 127.0.0.1
쿼리
데이터베이스 (4)
변수
상태
프로세스
명령-통계
데이터베이스
크기
항목
마지...
테이블
뷰
함수
프로...
트리거
information_schema
mysql
performance_schema
test
12 /* Characterset: utf8mb4 */
13 SHOW STATUS;
14 SHOW VARIABLES;
15 SHOW DATABASES;
16 /* "GREENJOA" 세션 시작 */
Connected: 00:00 h
MariaDB 10.1.1 가동 시간: 00:38 h
유휴

자바와 데이터베이스

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

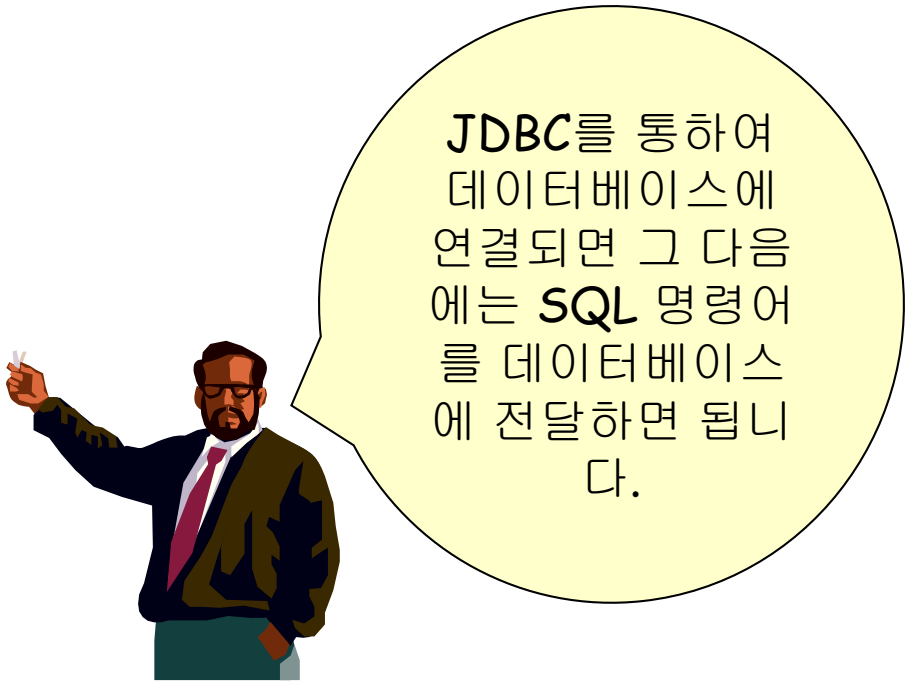


데이터베이스 프로그램 개발 절차

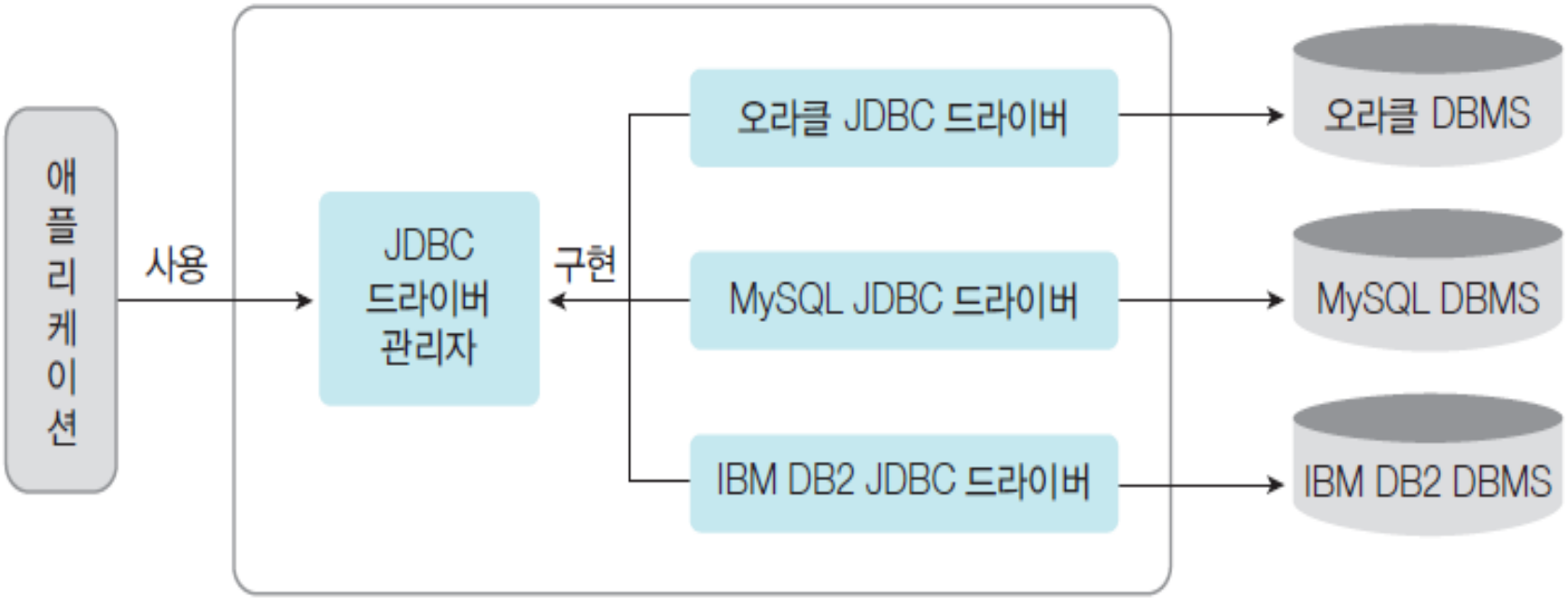
① DBMS(DataBase Management System)를 설치

② 자신이 설치한 DBMS에 필요한 JDBC 드라이버를 설치한다.

③ JDBC가 제공하는 기능을 이용하여 데이터베이스 응용 프로그램을 개발한다.

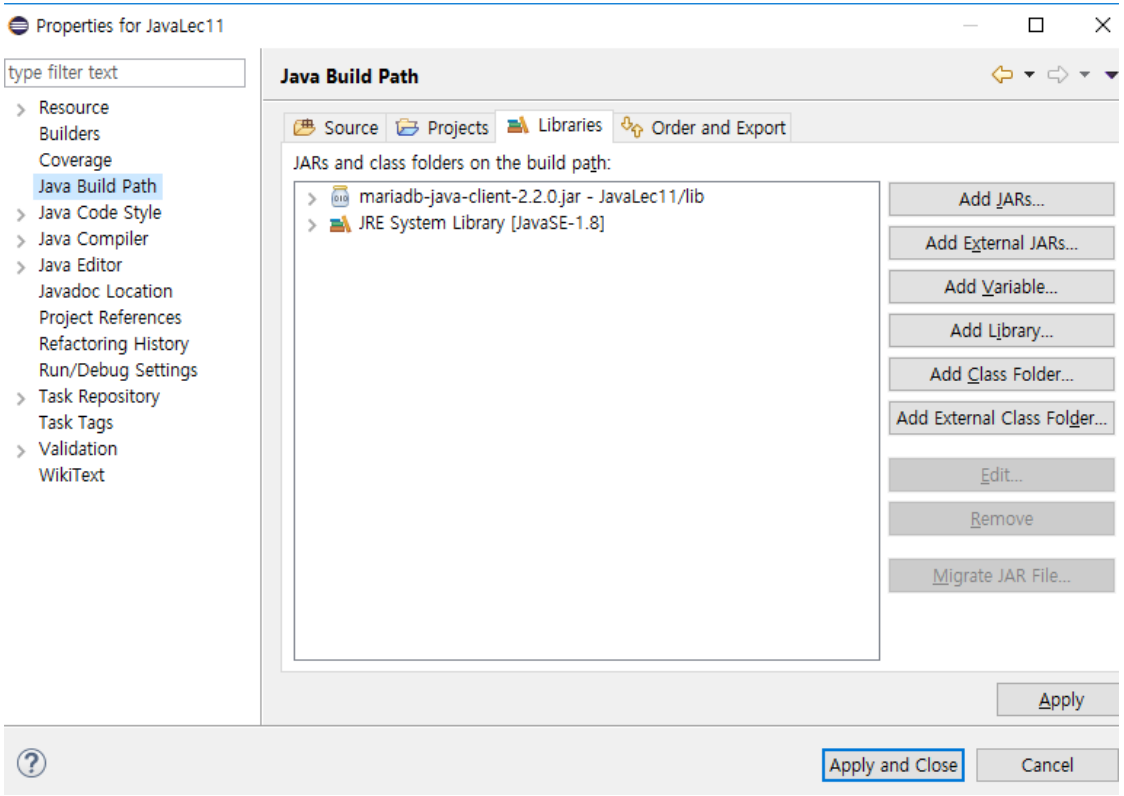


JDBC : Java Database Connectivity



JDBC 설치 및 연결 설정

- JDBC의 설치 및 연결 설정 방법
  - [java 설치 디렉터리₩jre8₩lib₩ext]에 라이브러리를 복사
  - 운영체제의 환경 변수에 CLASSPATH를 설정
  - 이클립스 프로젝트 속성에 빌드 경로를 추가



JDBC 프로그램 개발 절차

1단계 : JDBC 드라이버 로드

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

org.mariadb.jdbc.Driver

2단계 : 데이터베이스 연결

① **IP 주소 :** MySQL 데이터베이스가 설치된 컴퓨터의 IP 주소 또는 도메인이름.

② **스키마 :** 데이터베이스에서 생성한 스키마(데이터베이스) 이름.

③ **PORT :** 기본 설정값인 3306 포트를 사용할 때는 생략할 수 있다.

▶ **JDBC URL**

```
jdbc:<서브 프로토콜>:<데이터 원본 식별자>
```

```
jdbc:mysql://DB 서버의 IP 주소/스키마:PORT(옵션임)
                  ❶          ❷      ❸
```

▶ **Connection 클래스 인스턴스 레퍼런스 얻기**

```
Connection conn = DriverManager.getConnection(JDBC_URL, "아이디", "비밀번호");
                                              ❶          ❷
```

① **JDBC_URL :** 해당 데이터베이스에 맞게 미리 정의한 문자열.

② **아이디와 비밀번호 :** 시스템에 로그인하는 계정이 아닌 데이터베이스 자체에서 관리하는 계정

JDBC 프로그램 개발 절차

3단계 : Statement 생성

▶ executeQuery( ) : SELECT 문을 수행할 때 사용

```
String sql = "select * from test";
Statement stmt = conn.createStatement();
stmt.executeQuery(sql);
```

▶ executeUpdate( ) : UPDATE 문, DELETE 문 등을 수행할 때 사용한다.

```
Statement stmt = conn.createStatement();
stmt.executeUpdate("insert into test values(' "+getUsername()+" ', ' "+getEmail()+" ')");
```

▶ PreparedStatement : SQL 문을 미리 만들어 두고 변수를 따로 입력하는 방식이므로, 효율성이 나

```
PreparedStatement pstmt = conn.prepareStatement("insert into test values(?, ?)");
pstmt.setString(1, getUsername());
pstmt.setString(2, getEmail());
pstmt.executeUpdate();
```

```
stmt.close();
pstmt.close();
```

JDBC 프로그램 개발 절차

4단계 : SQL 문 전송

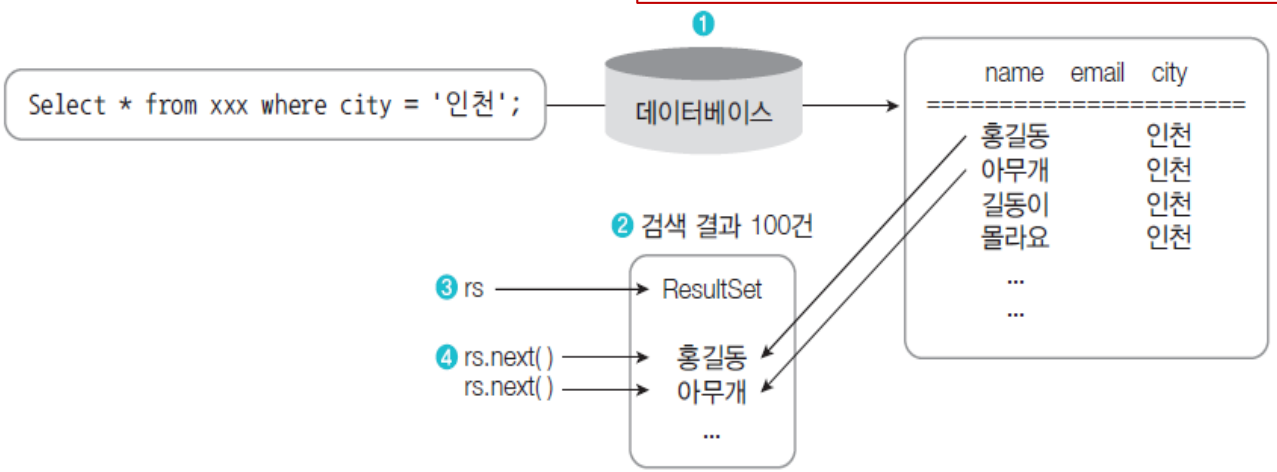
- 데이터를 입력·수정·삭제하려고 SQL 문을 만들 때는 PreparedStatement를 사용하여 변수와 적절히 조합.
- 데이터의 입력, 수정, 삭제는 Statement나 PreparedStatement의 executeUpdate( ) 메서드를 사용

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

```
pstmt.executeUpdate();
```

5 단계 : 결과 받기

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



JDBC 프로그램 개발 절차

5단계 : 결과받기

- Statement나 PreparedStatement의 executeQuery( )를 사용한다.
- 입력, 수정, 삭제와 달리 데이터를 가져올 때는 가져온 결과 데이터를 처리하는 ResultSet 객체가 필요하다.

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

```
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();
```

6단계 : 연결해제

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

UI 디자인


```
JTextArea text;
JScrollPane jscroll;
text = new JTextArea(10,30);
jscroll = new JScrollPane(text,
    JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
    JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
panel.add(jscroll);
```

DB 연결


```
public void connectDB() {
  try {

    Class.forName("org.mariadb.jdbc.Driver");
    conn = DriverManager.getConnection("jdbc:mariadb://127.0.0.1:3306/userdb",
         "root", "1234");
  } catch (ClassNotFoundException | SQLException e1) {
    e1.printStackTrace();
  }
}
```

데이터 삽입


```
public void registerUser() {
    String sql = "insert into userinfo values(?,?)";
    try {
        pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, id.getText());
        pstmt.setString(2, name.getText());
        pstmt.executeUpdate();
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

데이터 리스트 획득


```
public void printList() {
    String sql = "select * from userinfo";
    try {
        pstmt = conn.prepareStatement(sql);
        rs = pstmt.executeQuery();
        text.setText("");
        while(rs.next()) {
            String ssid = rs.getString(1);
            String ssname = rs.getString(2);
            text.append(ssid+","+ssname+"\n");
        }
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

DB 연결 해제


```
public void closeDB() {
    try {
        pstmt.close();
        rs.close();
        conn.close();
    } catch (SQLException e) {
    // TODO Auto-generated catch block
        e.printStackTrace();
    }

}
```

버튼 이벤트 처리


```
btn1 = new JButton("등록");
btn1.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
    // TODO Auto-generated method stub
        connectDB();
        registerUser();
        printList();
        closeDB();
    }
});
```

User 클래스


```
public class User {
    String userid;
    String userName;
    public User() {
        this("noId","noName");
    }
    public User(String id, String name) {
        userid = id;
        userName = name;
    }
    String getid() {
        return userid;
    }
    String getName() {
        return userName;
    }
}
```

DB 내용 배열에 담기


```
public ArrayList<User> getAll(){
    connectDB();
    ArrayList<User> list = new ArrayList<User>();
    String sql = "select * from userinfo";
    try {
        pstmt = conn.prepareStatement(sql);
        rs = pstmt.executeQuery();
        while(rs.next()) {
            String ssid = rs.getString(1);
            String ssname = rs.getString(2);
            list.add(new User(ssid, ssname));
        }
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    closeDB();
    return list;
}
```