

Java프로그래밍

15강. JDBC 프로그래밍 (교재 14장)

컴퓨터과학과 김희천 교수

오늘의 학습목차

1. JDBC와 MySQL
2. JDBC 프로그래밍
3. DatabaseMetaData 객체와 Statement 객체
4. ResultSet 객체
5. PreparedStatement 객체와
ResultSetMetaData 객체

1. JDBC와 MySQL

1. JDBC와 MySQL

1) 데이터베이스 프로그래밍

◆ JDBC(Java DataBase Connectivity) API

- ✓ Java 프로그램에서 관계형 데이터베이스를 사용하기 위한 API
규격
 - JDK의 일부로 포함되어 있음(java.sql)
- ✓ 데이터베이스에 연결하고, 데이터베이스에 대해 질의와 갱신을
요청하고, 결과를 받기 위한 프로그래밍 방법을 제공

JDBC 드라이버

- ◆ JDBC API는 DBMS 제조사가 제공하는 JDBC 드라이버를 통해
구현됨
- ◆ 사용하고자 하는 특정 DBMS의 JDBC 드라이버를 다운로드 받
아 설치해야, JDBC API를 사용한 프로그램을 실행할 수 있음

1. JDBC와 MySQL


2) DBMS 설치

◆ MySQL을 설치하기로 함

- ✓ MySQL Community Server의 최신 버전을 설치
 - Oracle 회원 가입 필요
- ✓ <https://dev.mysql.com/downloads/>에 접속
 - MySQL Installer for Windows를 클릭하면
 - Windows (x86, 32-bit), MSI Installer가 등장함
 - mysql-installer-community-8.0.23.0.msi
- ✓ MySQL Installer 창의 Setup Type에서 "Server only"로 설치
 - 설치 과정에서 root 계정의 암호를 입력하고 반드시 기억해야 함
 - 기본적으로 이름이 MYSQL80인 Window Service로 등록됨

1. JDBC와 MySQL

3) MySQL의 사용(1)

- ◆ MySQL데몬(mysql.exe)이 실행 중인지 확인
 - ✓ 데몬을 실행하려면 “Windows 관리 도구”에서 “서비스” 이용
 - 윈도우 시작 버튼을 클릭하고 “서비스”를 타이핑하면 나옴
- ◆ MySQL 8.0 Command Line Client 프로그램을 실행
 - ✓ 시작 메뉴>MySQL>MySQL 8.0 Command Line Client
 - ✓ 또는 명령 프롬프트 창에서 `mysql -u root -p`를 실행

1. JDBC와 MySQL

3) MySQL의 사용(2)

- ◆ Command Line Client를 실행하고, root 계정의 암호를 입력하여, MySQL 서버에 접속함

```
mysql> show databases;  
mysql> create database test;  
mysql> show databases;  
mysql> use test;  
mysql> create table book  
-> ( title varchar(100),  
-> author varchar(20),  
-> price int);  
mysql> show tables;  
mysql> show columns from book;  
mysql> insert into book values( ' 데이터베이스 ' , ' 정재현 ' , 200);  
mysql> insert into book values( ' C 프로그래밍 ' , ' 홍길동 ' , 300);
```

title	author	price
데이터베이스	정재현	200
C 프로그래밍	홍길동	300

1. JDBC와 MySQL

3) MySQL의 사용(3)

```
mysql> select * from book;
```

title	author	price
데이터베이스	정재현	200
C 프로그래밍	홍길동	300

```
2 rows in set (0.00 sec)
```

```
mysql> update book set price=500 where author='정재현';
```

```
Query OK, 1 row affected (0.02 sec)
```

```
Rows matched: 1 Changed: 1 Warnings: 0
```

```
mysql> delete from book where author='정재현';
```

```
Query OK, 1 row affected (0.03 sec)
```

```
mysql> quit
```


1. JDBC와 MySQL

4) JDBC 드라이버 설치

- ◆ DBMS 사이트에서 JDBC 드라이버를 다운로드 받아 설치함
- ◆ MySQL을 사용하는 경우
 - ✓ <https://dev.mysql.com/downloads/>에 접속
 - ✓ Connector/J를 다운로드 받음
 - Platform Independent에서 zip파일을 다운 받음
 - ✓ 이클립스 설정
 - Eclipse에서는 Project>Properties>Java Build Path>Libraries>Classpath>Add External JARs...를 이용하여 mysql-connector-java-8.0.23.jar를 추가함

2. JDBC 프로그래밍

2. JDBC 프로그래밍

1) JDBC 프로그래밍 기본

- ◆ JDBC 패키지를 import
 - ✓ `import java.sql.*;`
- ◆ JDBC 드라이버를 동적으로 로드
 - ✓ `Class.forName(com.mysql.cj.jdbc.Driver)` 호출
 - ✓ 최신 JDBC 버전에서는 생략해도 됨
- ◆ DBMS와 연결 설정
 - ✓ `DriverManager.getConnection()` 호출
- ◆ SQL 실행
 - ✓ `Connection`, `Statement`, `ResultSet` 객체 사용
- ◆ 연결 해제
 - ✓ 사용 중인 데이터베이스 자원을 반납
 - `finally` 블록에서 `close()` 실행

2. JDBC 프로그래밍

2) DBMS와 연결하기

◆ 방법

- ✓ DriverManager.getConnection() 메서드는 URL, 사용자 아이디, 비밀번호를 이용하여 MySQL 서버에 접속을 시도
- ✓ 성공하면 Connection 유형의 객체를 리턴

```
String url = "jdbc:mysql://서버주소:3306/db이름";  
String user = "사용자아이디";  
String password = "비밀번호";  
Connection conn =  
    DriverManager.getConnection(url, user, password);
```

2. JDBC 프로그래밍

3) select 구문의 실행과 결과 받아 오기

◆ SQL 구문 실행

- ✓ Statement 객체를 생성하고 executeQuery()를 호출

◆ 쿼리 결과 받기

- ✓ ResultSet 객체를 사용

```
Statement stmt = conn.createStatement( );
ResultSet rs =
    stmt.executeQuery("select 속성명 from 테이블");
while(rs.next( )) {
    System.out.println(rs.getString("속성명"));
}
rs.close( ); stmt.close( ); conn.close( );
```

2. JDBC 프로그래밍

4) insert/update/delete 구문의 실행

◆ SQL 구문 실행

- ✓ Statement 객체를 생성하고 executeUpdate()를 호출
- ✓ 영향을 받은 행의 개수가 리턴됨

```
Statement stmt = conn.createStatement( );  
int resultCount =  
    stmt.executeUpdate("insert into 테이블명 values(…)");  
stmt.close( ); conn.close( );
```

2. JDBC 프로그래밍

5) JDBC 프로그램 예제

```
import java.sql.*;

public class JDBCTest1 {
    public static void main(String[] args) {
        Connection conn = null; Statement stmt = null; ResultSet rs = null;
        try {
            //Class.forName("com.mysql.cj.jdbc.Driver").newInstance( );
            String url = "jdbc:mysql://localhost/test";
            String user = "root";
            String pass = "비밀번호";
            conn = DriverManager.getConnection(url, user, pass);
            stmt = conn.createStatement( );
            rs = stmt.executeQuery("SELECT * FROM book");
            System.out.println("제목\t\t저자\t가격");
            while(rs.next( )) {
                System.out.print(rs.getString("title")+"\t");
                System.out.print(rs.getString("author")+"\t");
                System.out.println(rs.getInt("price")+"\t");
            }
        } catch (Exception ex) {
            System.out.println(ex);
        }
        ... ..
    }
}
```

제목	저자	가격
데이터베이스	정재헌	200
C 프로그래밍	홍길동	300

3. DatabaseMetaData 객체와 Statement 객체

3. DatabaseMetaData 객체와 Statement 객체

1) DatabaseMetaData 객체

- ◆ DBMS의 정보를 가지는 객체
- ◆ 객체는 Connection 객체의 getMetaData() 메소드로 생성됨

주요 메소드

- ◆ String getDriverName()
- ◆ String getURL()
- ◆ String getUsername()

3. DatabaseMetaData 객체와 Statement 객체

2) DBMS 정보 알아내기

```
import java.sql.*;

public class DatabaseMetaDataTest{
    public static void main(String[ ] args) {
        Connection conn = null;
        try {
            String url = "jdbc:mysql://localhost/test";
            String user = "사용자";
            String pass = "비밀번호";
            conn = DriverManager.getConnection(url, user, pass);
            DatabaseMetaData dbmd = conn.getMetaData( );
            String driver_name = dbmd.getDriverName( );
            System.out.println(driver_name);
            System.out.println(dbmd.getURL( ));
            System.out.println(dbmd.getUserName( ));
        } catch (Exception ex) {
            System.out.println(ex);
        }
        ... ..
    }
}
```

```
MySQL Connector/J
jdbc:mysql://localhost/test
root@localhost
```

3. DatabaseMetaData 객체와 Statement 객체

3) Statement 객체(1)

- ◆ SQL 구문을 실행하고 결과를 반환해 주는 객체
- ◆ 객체는 Connection 객체의 createStatement() 메소드를 통해 생성됨

주요 메소드

- ◆ boolean execute(String sql)
 - ✓ SQL 구문을 실행하며, select 구문을 실행하는 경우 true를 리턴
 - ✓ 이어서 getResultSet() 또는 getUpdateCount()를 호출함

```
stmt = conn.createStatement( );  
if(stmt.execute("SELECT * FROM book")) rs=stmt.getResultSet( );  
  
stmt = conn.createStatement( );  
if(!stmt.execute("delete from book where ..."))  
    System.out.println(stmt.getUpdateCount( )+ " 개의 행 삭제");
```

3. DatabaseMetaData 객체와 Statement 객체

3) Statement 객체(2)

주요 메소드

- ◆ ResultSet getResultSet()
 - ✓ SQL 구문(select 구문)을 실행한 결과를 리턴함
- ◆ int getUpdateCount()
 - ✓ SQL 구문(select 구문 제외)의 실행으로 영향을 받은 행의 개수를 리턴함
- ◆ ResultSet executeQuery(String sql)
 - ✓ select 구문을 실행할 때 사용
 - ✓ 실행 결과를 나타내는 테이블인 ResultSet 객체를 리턴함
- ◆ int executeUpdate(String sql)
 - ✓ update, insert, delete 구문을 실행할 때 사용
 - ✓ 영향 받은 행의 개수를 리턴함

3. DatabaseMetaData 객체와 Statement 객체

4) Statement 객체의 사용 예

◆ update, insert, delete 구문의 실행

```
Connection conn = DriverManager.getConnection(url, user, pass);
Statement stmt = conn.createStatement( );

int resultCount = stmt.executeUpdate("insert into book values (...");
System.out.println(resultCount + " 개의 행이 삽입되었습니다.");

resultCount = stmt.executeUpdate("update book set name = value, ...");
System.out.println(resultCount + " 개의 행이 변경되었습니다.");

resultCount = stmt.executeUpdate("delete from book where ...");
System.out.println(resultCount + " 개의 행이 삭제되었습니다.");
```

4. ResultSet 객체

4. ResultSet 객체

1) ResultSet 객체

- ◆ select 구문의 실행 결과를 나타내는 테이블
 - ✓ Statement 객체의 `getResultSet()`, `executeQuery()` 메소드가 리턴하는 객체
 - ✓ 테이블에서 한 행을 가리키는 커서를 가짐

⇒

Title	author	price
데이터베이스	정재헌	200
C 프로그래밍	홍길동	300

- ◆ select 구문을 실행하여 ResultSet 객체가 생성되면 커서가 만들어지고, select 구문의 실행 결과를 가리킴
 - ✓ 커서는 행을 가리키는 포인터, 기본적으로 위에서 아래로 진행
 - ✓ 커서의 초기값은 첫 행의 직전 행을 가리킴

4. ResultSet 객체

2) ResultSet의 메소드

- ◆ boolean next()
 - ✓ 커서를 다음 행으로 이동시킨다.
- ◆ boolean previous()
 - ✓ 커서를 이전 행으로 이동시킨다.
- ◆ Statement getStatement()
 - ✓ 현재 ResultSet을 생성시킨 Statement 객체를 리턴함
- ◆ String getString(int index), String getString(String columnName)
 - ✓ ResultSet 객체에서 해당 열의 문자열을 리턴함
- ◆ int getInt(int index), int getInt(String columnName)
 - ✓ ResultSet 객체에서 해당 열의 int 값을 리턴함

4. ResultSet 객체

3) ResultSet 객체의 사용 예

```
try {  
    ... ..  
    conn = DriverManager.getConnection(url, user, pass);  
    stmt = conn.createStatement( );  
    rs = stmt.executeQuery("SELECT * FROM book");  
    System.out.println("제목\t\t저자\t가격");  
    while(rs.next( )) {  
        System.out.print(rs.getString(1)+"\t");  
        System.out.print(rs.getString(2)+"\t");  
        System.out.println(rs.getInt(3)+"\t");  
    }  
} catch (Exception ex) {  
    System.out.println(ex);  
}
```

제목	저자	가격
데이터베이스	정재헌	200
C 프로그래밍	홍길동	300
삼국지	나승민	1000
Java	김준표	500

5. PreparedStatement 객체와 ResultSetMetaData 객체

5. PreparedStatement 객체와 ResultSetMetaData 객체

1) PreparedStatement 객체

◆ Precompile된 SQL 문을 표현

- ✓ 객체는 Connection 객체의 `prepareStatement(String sql)` 메소드를 통해 생성됨
 - 객체를 생성할 때 SQL 구문이 주어짐(SQL 구문을 실행할 때가 아님)

◆ 같은 SQL 문을 여러 번 실행할 때 효율적임

- ✓ SQL문에 매개 변수(?)를 사용하고, 실행 전에 값을 지정할 수 있음

```
String query = "insert into book values(?, ?, ?)";  
ps = conn.prepareStatement(query);  
ps.setString(1, "삼국지");  
ps.setString(2, "나관중");  
ps.setInt(3, 1000);  
ResultCount = ps.executeUpdate( );
```

5. PreparedStatement 객체와 ResultSetMetaData 객체

2) PreparedStatement 의 주요 메소드

- ◆ boolean execute()
- ◆ ResultSet executQuery()
- ◆ int executeUpdate()
 - ✓ 인자가 없음
- ◆ void setInt(int parameterIndex, int x)
- ◆ void setString(int parameterIndex, String x)
 - ✓ SQL 구문에 있는 매개 변수(?)에 값을 지정
 - ✓ SQL 구문에서 첫 번째 나온 ?의 인덱스가 1임

5. PreparedStatement 객체와 ResultSetMetaData 객체

3) PreparedStatement 객체의 사용 예

```
try {
    ... ..
    conn = DriverManager.getConnection(url, user, pass);
    String query = "select * from book where price > ?";
    ps = conn.prepareStatement(query);
    ps.setInt(1, 100);
    rs = ps.executeQuery( );
    System.out.println("제목\t\t저자\t가격");
    while(rs.next( )) {
        System.out.print(rs.getString(1)+"\t");
        System.out.print(rs.getString(2)+"\t");
        System.out.println(rs.getInt(3)+"\t");
    }
} catch (Exception ex) {
    System.out.println(ex);
}
```

5. PreparedStatement 객체와 ResultSetMetaData 객체

4) DBMS와 Java의 자료형 변환

- ◆ DBMS 테이블에서 열의 자료형과 Java의 자료형, 그리고 JDBC 메소드 간의 관계

DBMS 자료형	Java 자료형	ResultSet 메소드	PreparedStatement 메소드
CHAR	String	getString()	setString()
VARCHAR	String	getString()	setString()
INTEGER	int	getInt()	setInt()
DATE	java.sql.Date	getDate()	setDate()
... ..			

5. PreparedStatement 객체와 ResultSetMetaData 객체

5) ResultSetMetaData 객체

- ◆ ResultSet 객체에서 테이블의 이름, 열의 이름과 타입 정보를 얻을 때 사용되는 객체
 - ✓ ResultSet의 getMetaData() 메소드로 생성함

주요 메소드

- ◆ String getColumnName(int index)
 - ✓ index 위치의 컬럼 이름을 리턴
- ◆ int getColumnCount()
 - ✓ ResultSet의 컬럼 개수를 리턴
- ◆ int getColumnType(int index)
 - ✓ index 위치의 컬럼 자료형을 리턴
- ◆ String getTableName(int index)
 - ✓ index 위치의 컬럼을 포함하는 테이블의 이름을 리턴

5. PreparedStatement 객체와 ResultSetMetaData 객체

6) ResultSetMetaData 객체의 사용 예

```
try {
    ... ..
    conn = DriverManager.getConnection(url, user, pass);
    String query = "select * from book"; ps = conn.prepareStatement(query);
    rs = ps.executeQuery( );
    ResultSetMetaData rsmd = rs.getMetaData( );
    int colCount = rsmd.getColumnCount( );
    for(int i = 1; i <= colCount; i++) {
        System.out.print(rsmd.getColumnName(i) + "\t");
    }
    System.out.println( );
    while(rs.next( )) {
        for(int i = 1; i <= colCount; i++) {
            switch(rsmd.getColumnType(i)) {
                case Types.INTEGER:
                    System.out.print(rs.getInt(i) + "\t");
                    break;
                case Types.VARCHAR:
                    System.out.print(rs.getString(i) + "\t");
                    break;
            }
        }
        System.out.println( );
    }
} catch (Exception ex) { System.out.println(ex); }
```

title	author	price
데이터베이스	정재현	200
C 프로그래밍	홍길동	300
삼국지	나관중	1000

Java프로그래밍

감사합니다.
한 학기 동안 수고하셨습니다.