14장 데이터베이스 프로그래밍 기초

14.1 데이터베이스 기초

14.1.1 데이터베이스와 DBMS

■ 데이터베이스(database)
- 빠른 탐색과 검색을 위해 조직된 데이터의 집합체

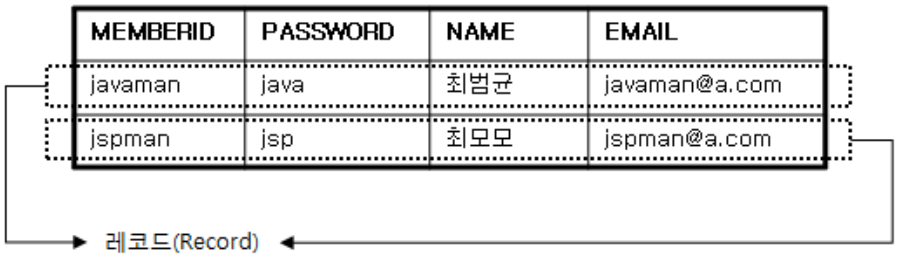
■ DBMS(Database Management System)
- 데이터베이스를 관리하기 위한 시스템
- 주요 기능
  - 데이터의 추가/조회/변경/삭제
  - 데이터의 무결성(integrity) 유지
  - 트랜잭션 관리
  - 데이터의 백업 및 복원
  - 데이터 보안

14.1.2 테이블과 레코드

■ 테이블 - 데이터가 저장되는 가상의 장소

■ 테이블은 1개 이상의 칼럼으로 구성
- 각 칼럼은 타입을 가지며, 제약(값의 길이, 가질 수 있는 값 등)을 갖는다.
- 이런 테이블의 구성을 스키마(schema)라고 함

■ 칼럼의 모음을 레코드(record)라고 표현
- 하나의 테이블은 여러 개의 레코드로 구성



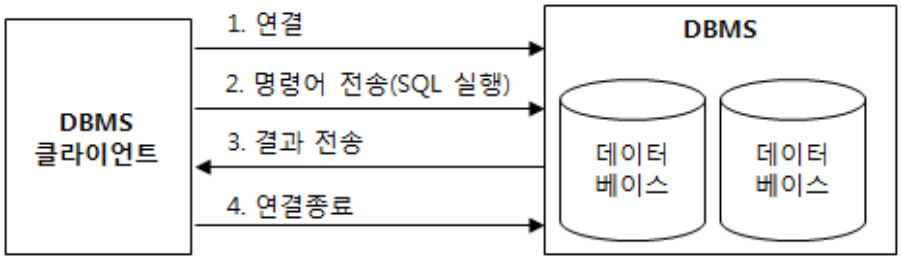
14.1.3 주요키(Primary Key)와 인덱스(Index)

■ 주요키(Primary Key)
- 각각의 레코드를 구별하기 위해 사용되는 것
- 각 레코드가 서로 다른 값을 갖는 칼럼
- 주요키 값을 이용해서 빠른 검색 가능

■ 인덱스
- 지정한 칼럼에 맞춰 데이터의 정렬 순서를 미리 계산
- 주요키도 인덱스의 종류

- 인덱스로 사용되는 칼럼은 중복된 값을 가질 수도 있음

14.1.4 데이터베이스 프로그래밍의 일반적 순서



14.1.5 데이터베이스 프로그래밍의 필수 요소

- DBMS : 데이터베이스를 관리해주는 시스템
- 데이터베이스 : 데이터를 저장할 공간
- DBMS 클라이언트 : 데이터베이스를 사용하는 어플리케이션

14.2 예제 실행을 위한 데이터베이스 생성

```
create database chap14 default character set utf8;

create user 'jspexam'@'localhost' identified by 'jsppw';
grant all privileges on chap14.* to 'jspexam'@'localhost';

create user 'jspexam'@'%' identified by 'jsppw';
grant all privileges on chap14.* to 'jspexam'@'%';


drop table member;
create table member (
   memberid varchar(10) not null primary key,
   password varchar(10) not null,
   name varchar(20) not null,
   email varchar(80)
) engine=innodb default character set = utf8;

insert into member (memberid,password,name) values ('madvirus','1234','최범균');
insert into member values ('eral3','5678','최범균','madvirus@madvirus.net');


drop table member_etc;
create table member_etc (
   memberid varchar(10) not null primary key,
   birthday char(8)
) engine=innodb default character set = utf8;


drop table member_history;
create table member_history (
   memberid varchar(10) not null primary key,
```

```
   history long varchar
) engine=innodb default character set = utf8;

insert into member_history values ('madvirus',
   concat(
      '2015 스프링4 프로그래밍입문<br>',
      '2013 Spring4.0 프로그래밍<br>',
      '2012 객체 지향과 디자인 패턴<br>',
      '2012 JSP 2.2 웹프로그래밍\n'
   )
);
```

■ MySQL 워크벤치에 다음 정보를 사용하는 연결을 새로 추가한 뒤 연결이 잘 되는지 확인한다.

- Connection Name : chap14 DB
- Username : jspexam
- Password : jsppw
- Default Schema : chap14

14.3 SQL 기초

14.3.1 주요 SQL 타입

SQL 타입	설명
CHAR	확정 길이의 문자열을 지정한다. 표준의 경우 255글자까지만 지정할 수 있다.
VARCHAR	가변 길이의 문자열을 지정한다. 표준의 경우 255글자까지만 지정할 수 있다.
LONG VARCHAR	긴 가변 길의의 문자열을 지정한다.
NUMERIC	숫자를 지정한다.
DECIMAL	십진수를 지정한다.
INTEGER	정수를 지정한다.
TIMESTAMP	날짜와 시간을 지정한다.
TIME	시간을 지정한다.
DATE	날짜를 지정한다.
CLOB	대량의 문자열 데이터를 지정한다.
BLOB	대량의 이진 데이터를 지정한다.

14.3.2 테이블 생성 쿼리

```
create table member (
   memberid varchar(10) not null primary key,
   password varchar(10) not null,
   name varchar(20) not null,
   email varchar(80)
) engine=innodb default character set = utf8;
```

14.3.3 데이터 삽입 쿼리

```
insert into member (memberid,password,name) values ('madvirus','1234','최범균');
insert into member values ('eral3','5678','최범균','madvirus@madvirus.net');
```

14.3.4 데이터 조회 쿼리 - 조회 및 조건

```
select * from member where name = '최범균';
```

14.3.5 데이터 쿼리 조회 - 정렬

```
select * from member order by name asc, memberid asc;
```

14.3.6 데이터 쿼리 조회 - 집합

```
select count(*) from member where name like '최%';
```

14.3.7 데이터 수정 쿼리

```
update member set name = '최범균';
```

14.3.8 데이터 삭제 쿼리

```
delete from member where memberid = 'eral3';
```

14.3.9 조인

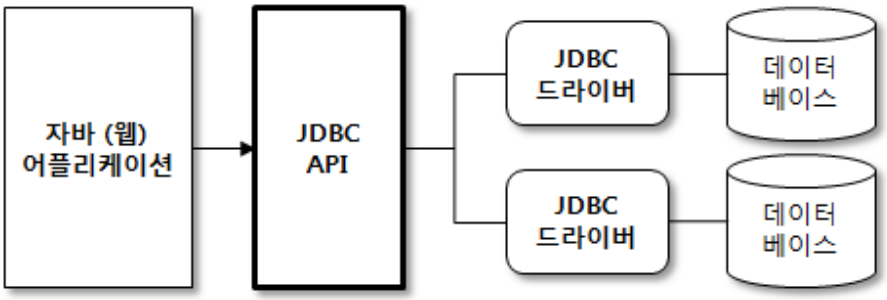
```
select * from member as a, member_etc as b
where a.memberid = b.memeberid;
```

14.4 프로젝트 준비

14.5 JSP에서 JDBC 프로그래밍하기

- Java Database Connectivity
- 자바에서 DB 프로그래밍을 하기 위해 사용되는 API

14.5.1 JDBC의 구조



14.5.2 JDBC 드라이버 준비하기

- JDBC 드라이버를 다운로드 받아 웹 어플리케이션 폴더의 WEB-INF\lib 폴더에 JDBC 드라이버 파일을 복사한다.
  - MySQL : http://dev.mysql.com/downloads/connector/j/
    - mysql-connector-java-5.1.35-bin.jar
  - Oracle : http://www.oracle.com/technetwork/database/features/jdbc/index.html
    - ojdbc6.jar

-

- 혹은 Apache Maven 설정을 위해 pom.xml 파일에 아래와 같이 추가한다.

```
<dependencies>
        <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
        <dependency>
                <groupId>mysql</groupId>
                <artifactId>mysql-connector-java</artifactId>
                <version>5.1.35</version>
        </dependency>
</dependencies>
```

14.5.3 JDBC 프로그래밍의 코딩 스타일

- JDBC 프로그램의 전형적인 실행 순서
  1. JDBC 드라이버 로딩
  2. 데이터베이스 커넥션 구함
  3. 쿼리 실행을 위한 Statement 객체 생성
  4. 쿼리 실행
  5. 쿼리 실행 결과 사용
  6. Statement 종료
  7. 데이터베이스 커넥션 종료

[chap14/WebContent/viewMemberList.jsp]

```
01  <%@ page contentType = "text/html; charset=utf-8" %>
02  <%@ page import = "java.sql.DriverManager" %>
03  <%@ page import = "java.sql.Connection" %>
04  <%@ page import = "java.sql.Statement" %>
05  <%@ page import = "java.sql.ResultSet" %>
06  <%@ page import = "java.sql.SQLException" %>
07
08  <html>
09  <head><title>회원 목록</title></head>
```

```
<body>

MEMBER 테이블의 내용
<table width="100%" border="1">
<tr>
        <td>이름</td><td>아이디</td><td>이메일</td>
</tr>
<%
        // 1. JDBC 드라이버 로딩
        Class.forName("com.mysql.jdbc.Driver");

        Connection conn = null;
        Statement stmt = null;
        ResultSet rs = null;

        try {
                String jdbcDriver = "jdbc:mysql://localhost:3306/chap14?" +

        "useUnicode=true&characterEncoding=utf8";
                String dbUser = "jspexam";
                String dbPass = "jsppw";

                String query = "select * from MEMBER order by MEMBERID";

                // 2. 데이터베이스 커넥션 생성
                conn = DriverManager.getConnection(jdbcDriver, dbUser, dbPass);

                // 3. Statement 생성
                stmt = conn.createStatement();

                // 4. 쿼리 실행
                rs = stmt.executeQuery(query);

                // 5. 쿼리 실행 결과 출력
                while(rs.next()) {
%>
<tr>
        <td><%= rs.getString("NAME") %></td>
        <td><%= rs.getString("MEMBERID") %></td>
        <td><%= rs.getString("EMAIL") %></td>
</tr>
<%
                }
        } catch(SQLException ex) {
                out.println(ex.getMessage());
                ex.printStackTrace();
        } finally {
                // 6. 사용한 Statement 종료
                if (rs != null) try { rs.close(); } catch(SQLException ex) {}
                if (stmt != null) try { stmt.close(); } catch(SQLException ex) {}

                // 7. 커넥션 종료
                if (conn != null) try { conn.close(); } catch(SQLException ex) {}
        }
%>
</table>

</body>
</html>
```

14.5.4 DBMS와의 통신을 위한 JDBC 드라이버

- DBMS와 통신을 담당하는 자바 클래스

- DBMS 별로 알맞은 JDBC 드라이버 필요
  - 보통 jar 파일로 제공

- JDBC 드라이버 로딩
  - DBMS와 통신하기 위해서는 먼저 로딩해 주어야 함
  - 로딩 코드
    - Class.forName("JDBC드라이버 클래스의 완전한 이름");
  - 주요 DBMS의 JDBC 드라이버
    - MySQL : com.mysql.jdbc.Driver
    - 오라클 : oracle.jdbc.driver.OracleDriver
    - MS SQL 서버 : com.microsoft.sqlserver.jdbc.SQLServerDriver

14.5.5 데이터베이스 식별을 위한 JDBC URL

- DBMS와의 연결을 위한 식별 값

- JDBC 드라이버에 따라 형식 다름

- 일반적인 구성
  - jdbc:[DBMS]:[데이터베이스식별자]

- 주요 DBMS의 JDBC URL 구성
  - MySQL : jdbc:mysql://HOST[:PORT]/DBNAME[?param=value&param1=value2&...]
  - Oracle: jdbc:oracle:thin:@HOST:PORT:SID
  - MS SQL : jdbc:sqlserver://HOST[:PORT];databaseName=DB

```
//MySQL
jdbc:mysql://localhost:3306/chap14
jdbc:mysql://localhost:3306/chap14?useUnicode=true&characterEncoding=utf8

//Oracle
jdbc:oracle:thin:@127.0.0.1:1521:ORCL
```

14.5.6 데이터베이스 커넥션

- DriverManager를 이용해서 Connection 생성
  - DriverManager.getConnection(String jdbcURL)
  - DriverManager.getConnection(String jdbcURL, String user, String password)

```
// 일반적인 코드 구성
Connection conn = null;
try {
    String jdbcDriver = "jdbc:mysql://localhost:3306/chap14?" +
                        "useUnicode=true&characterEncoding=utf8";
    String dbUser = "jspexam";
    String dbPass = "jspex";

    conn = DriverManager.getConnection(jdbcDriver, dbUser, dbPass);
```

```
    ...
} catch(SQLException ex) {
    // 에러 발생
} finally {
    if (conn != null) try { conn.close(); } catch(SQLException ex) {}
}
```

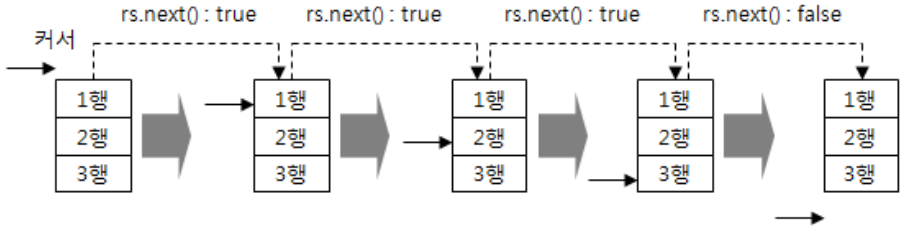
14.5.7 Statement를 사용한 쿼리 실행

- Connection.createStatement()로 Statement 생성
- Statement가 제공하는 메서드로 쿼리 실행
  - ResultSet executeQuery(String query) - SELECT 쿼리를 실행
  - int executeUpdate(String query) - INSERT, UPDATE, DELETE 쿼리를 실행

```
Statement stmt = null;
ResultSet rs = null;
try {
    stmt = conn.createStatement();
    int insertedCount = stmt.executeUpdate("insert …..");
    rs = stmt.executeQuery("select * from ….");
    …
} catch(SQLException ex) {
    …
} finally {
    if (rs != null) try { rs.close(); } catch(SQLException ex) {}
    if (stmt != null) try { stmt.close(); } catch(SQLException ex) {}
}
```

14.5.8 ResultSet에서 값 읽어오기

- next() 메서드로 데이터 조회 여부 확인



- 데이터 조회 위한 주요 메서드
  - getString()
  - getInt(), getLong(), getFloat(), getDouble()
  - getTimestamp(), getDate(), getTime()

```
rs = stmt.executeQuery("select * from member");
if (rs.next()) {  // 다음 행(첫 번째 행)이 존재하면 rs.next()는 true를 리턴
    // rs.next()에 의해 다음 행(첫 번째 행)으로 이동
    String name = rs.getString("NAME");
} else {
    // 첫 번째 행이 존재하지 않는다. 즉, 결과가 없다.
}
```

14.5.9 ResetSet에서 LONG VARCHAR 타입 값 읽어오기

- SQL의 LONG VARCHAR 타입은 대량의 텍스트 데이터를 저장할 때 사용한다.
- getCharacterStream() 메서드를 사용해야 한다.

```
String data = null;
java.io.Reader reader = null;
try {
   // 1. ResultSet의 getCharacterStream()으로 Reader 구함
   reader = rs.getCharacterStream("FIELD");

   if (reader != null) {
      // 2. 스트림에서 읽어온 데이터를 저장할 버퍼를 생성한다.
      // 3. 스트림에서 데이터를 읽어와 버퍼에 저장한다.
      // 4. 버퍼에 저장한 내용을 String으로 변환한다.
   }
} catch (IOException ex) {
   // 5. IO 관련 처리 도중 문제가 있으면 IOException이 발생한다.
} finally {
   // 6. Reader를 종료한다.
}
```

14.5.10 Statment를 이용한 쿼리 실행 시 작은따옴표 처리

```
update TABLENAME set SOMEFIELD = 'king"s choice' where ...
```

14.5.11 PreparedStatement를 사용한 쿼리 실행

- SQL의 틀을 미리 정해 놓고, 나중에 값을 지정하는 방식
- 쿼리 실행 관련 메서드
  - ResultSet executeQuery() - SELECT 쿼리를 실행할 때 사용되며 ResultSet을 결과값으로 리턴한다.
  - int executeUpdate() - INSERT, UPDATE, DELETE 쿼리를 실행할 때 사용되며, 실행 결과 변경된 레코드의 개수를 리턴한다

```
pstmt = conn.prepareStatement(
   "insert into MEMBER (MEMBERID, NAME, EMAIL) values (?, ?, ?)");
pstmt.setString(1, "madvirus"); // 첫번째 물음표의 값 지정
pstmt.setString(2, "최범균");   // 두번째 물음표의 값 지정
pstmt.executeUpdate();
```

메서드	설명
setString(int index, String x)	지정한 인덱스의 파라미터 값을 x로 지정한다.
setInt(int index, int x)	지정한 인덱스의 파라미터 값을 int 값 x로 지정한다.
setLong(int index, long x)	지정한 인덱스의 파라미터 값을 long 값 x로 지정한다.

setDouble(int index, double x)	지정한 인덱스의 파라미터 값을 double 값 x로 지정한다.
setFloat(int index, float x)	지정한 인덱스의 파라미터 값을 float 값 x로 지정한다.
setTimestamp(int index, Timesetamp x)	지정한 인덱스의 값을 SQL TIMESTMAP 타입을 나타내는 java.sql.Timestamp 타입으로 지정한다.
setDate(int index, Date x)	지정한 인덱스의 값을 SQL DATE 타입을 나타내는 java.sql.Date 타입으로 지정한다.
setTime(int index, Time x)	지정한 인덱스의 값을 SQL TIME 타입을 나타내는 java.sql.Time 타입으로 지정한다.

14.5.12 PreparedStatement에서 LONG VARCHAR 타입 값 지정하기

- setCharacterStream() 메서드는 Reader로부터 length 글자 수만큼 데이터를 읽어와 저장한다.

```
PreparedStatement pstmt = null;
try {
   String value = "...";
   pstmt = conn.prepareStatement(...);
   java.io.StringReader reader = new java.io.StringReader(value);
   pstmt.setCharacterStream(1, reader, value.length());
   ...
} catch (SQLException ex) {
   ...
} finally {
   if (pstmt != null) try { pstmt.close(); } catch (SQLException ex) {}
}------
```

14.5.13 PreparedStatement 쿼리를 사용하는 이유

- 반복해서 실행되는 동일 쿼리의 속도를 향상
  - DBMS가 PreparedStatement와 관련된 쿼리 파싱 회수 감소
- 값 변환 처리
  - 작은 따옴표 등 값에 포함된 특수 문자의 처리
- 코드의 간결함
  - 문자열 연결에 따른 코드의 복잡함 감소

14.6 웹 어플리케이션 구동 시 JDBC 드라이버 로딩하기

- DBMS와 통신하기 위해서는 먼저 로딩해 주어야 함
- 로딩 코드
  - Class.forName("JDBC드라이버 클래스의 완전한 이름");
- 주요 DBMS의 JDBC 드라이버
  - MySQL - com.mysql.jdbc.Driver
  - 오라클 - oracle.jdbc.driver.OracleDriver
  - MS SQL 서버 - com.microsoft.sqlserver.jdbc.SQLServerDriver

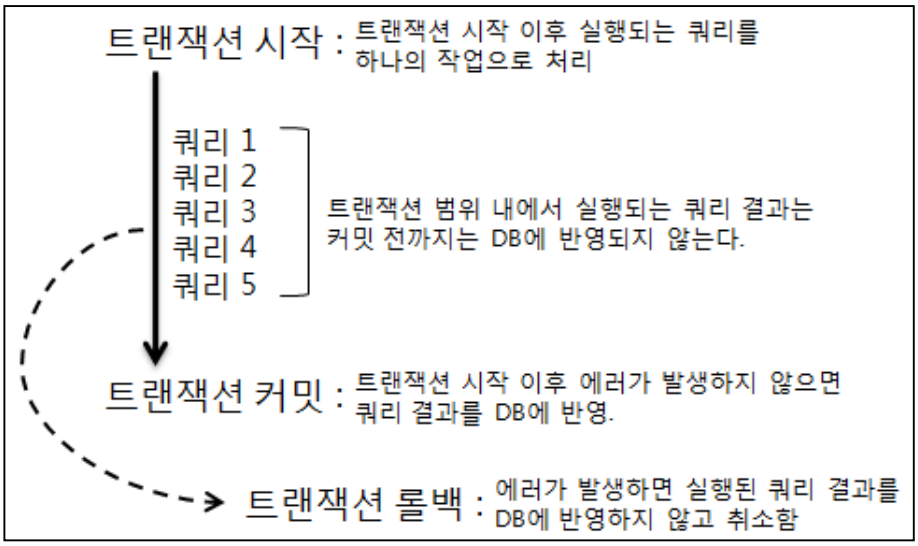
■ 웹 어플리케이션이 시작될 때 자동으로 MySQLDriverLoader 서블릿 클래스를 실행하도록 설정해야 한다. 아래와 같이 web.xml 파일을 만들어 <servlet> 태그를 추가하면 된다.

[chap14/WebContent/WEB-INF/web.xml]

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                  http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
         version="3.1">

         <servlet>
                  <servlet-name>mysqlDriverLoader</servlet-name>
                  <servlet-class>jdbc.MySQLDriverLoader</servlet-class>
                  <load-on-startup>1</load-on-startup>
         </servlet>

         <servlet>
                  <servlet-name>DBCPInit</servlet-name>
                  <servlet-class>jdbc.DBCPInit</servlet-class>
                  <load-on-startup>1</load-on-startup>
         </servlet>

</web-app>
```

14.7 JDBC에서 트랜잭션 처리

■ 데이터의 무결성을 위해 하나의 작업을 위한 쿼리는 트랜잭션으로 처리될 필요



■ 트랜잭션 구현 방법: 오토 커밋 해제, JTA 이용 방식

- Connection.setAutoCommit(false)

```
try {
    conn = DriverManager.getConnection(...);
    // 트랜잭션 시작
    conn.setAutoCommit(false);
    ... // 쿼리 실행
```

```
    ... // 쿼리 실행
    // 트랜잭션 커밋
    conn.commit();
} catch(SQLException ex) {
    if (conn != null) {
        // 트랜잭션 롤백
        conn.rollback();
    }
} finally {
    if (conn != null) {
        try {
            conn.close();
        } catch(SQLException ex) {}
    }
}
```

```
-- jspexam 계정으로 로그인하여 아래 두 개의 테이블을 생성한다.
create table ITEM (
          ITEM_ID int not null primary key,
    NAME varchar(100)
) engine=InnoDB default character set = utf8;

create table ITEM_DETAIL (
          ITEM_ID int not null primary key,
    DETAIL varchar(200)
) engine=innodb default character set = utf8;
```

[chap14/WebContent/insertItem.jsp]

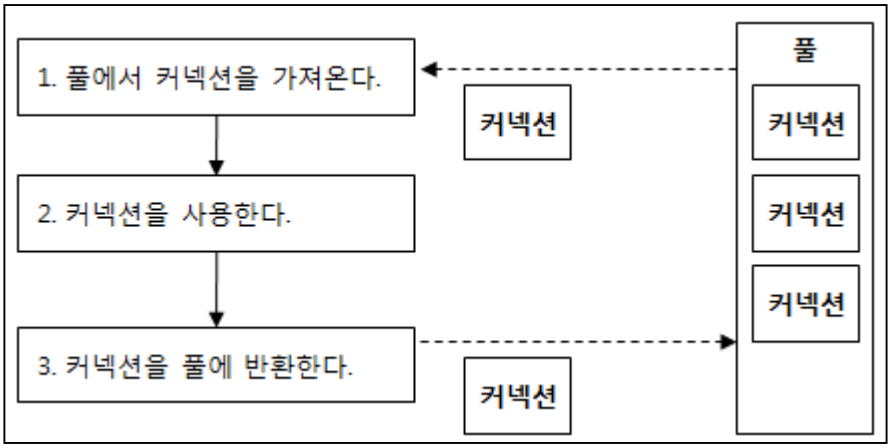
```
01    <%@ page contentType = "text/html; charset=utf-8" %>
02    <%@ page import = "java.sql.DriverManager" %>
03    <%@ page import = "java.sql.Connection" %>
04    <%@ page import = "java.sql.PreparedStatement" %>
05    <%@ page import = "java.sql.SQLException" %>
06    <%
07              String idValue = request.getParameter("id");
08
09              Connection conn = null;
10              PreparedStatement pstmtItem = null;
11              PreparedStatement pstmtDetail = null;
12
13              String jdbcDriver = "jdbc:mysql://localhost:3306/chap14?" +
14                                                              "useUnicode=true&characterEncoding=utf8";
15              String dbUser = "jspexam";
16              String dbPass = "jsppw";
17
18              Throwable occuredException = null;
19
20              try {
21                        int id = Integer.parseInt(idValue);
22
23                        conn = DriverManager.getConnection(jdbcDriver, dbUser, dbPass);
24                        conn.setAutoCommit(false);
25
26                        pstmtItem = conn.prepareStatement("insert into ITEM values (?, ?)");
27                        pstmtItem.setInt(1, id);
28                        pstmtItem.setString(2, "상품 이름 " + id);
29                        pstmtItem.executeUpdate();
30
31                        if (request.getParameter("error") != null) {
32                                  throw new Exception("의도적 익셉션 발생");
33                        }
```

```
34
35                      pstmtDetail = conn.prepareStatement(
36                              "insert into ITEM_DETAIL values (?, ?)");
37                      pstmtDetail.setInt(1, id);
38                      pstmtDetail.setString(2, "상세 설명 " + id);
39                      pstmtDetail.executeUpdate();
40
41                      conn.commit();
42              } catch(Throwable e) {
43                      if (conn != null) {
44                              try {
45                                      conn.rollback();
46                              } catch(SQLException ex) {}
47                      }
48                      occuredException = e;
49              } finally {
50                      if (pstmtItem != null)
51                              try { pstmtItem.close(); } catch(SQLException ex) {}
52                      if (pstmtDetail != null)
53                              try { pstmtDetail.close(); } catch(SQLException ex) {}
54                      if (conn != null) try { conn.close(); } catch(SQLException ex) {}
55              }
56      %>
57      <html>
58      <head><title>ITEM 값 입력</title></head>
59      <body>
60
61      <%  if (occuredException != null) { %>
62      에러가 발생하였음: <%= occuredException.getMessage() %>
63      <%  } else { %>
64      데이터가 성공적으로 들어감
65      <%  } %>
66      </body>
67      </html>
```

14.8 커넥션 풀

14.8.1 커넥션 풀이란

■ 데이터베이스와 연결된 커넥션을 미리 만들어서 풀(pool) 속에 저장해 두고 있다가 필요할 때에 커넥션을 풀에서 가져다 쓰고 다시 풀에 반환하는 기법



■ 특징

- 커넥션을 생성하는 데 드는 연결 시간이 소비되지 않는다.
- 커넥션을 재사용하기 때문에 생성되는 커넥션 수가 많지 않다.

14.8.2 DBCP를 이용해서 커넥션 풀 사용하기

(1) 필요한 jar 파일 복사하기

■ 필요 라이브러리 복사
- Commons-DBCP API : commons-dbcp2-2.1.jar
- Commons-Pool API : commons-pool2-2.4.1.jar
- Commons Logging API : common-logging-1.2.jar

■ 혹은 Maven 설정

```
<!-- https://mvnrepository.com/artifact/org.apache.commons/commons-dbcp2 -->
<dependency>
        <groupId>org.apache.commons</groupId>
        <artifactId>commons-dbcp2</artifactId>
        <version>2.1</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.apache.commons/commons-pool2 -->
<dependency>
        <groupId>org.apache.commons</groupId>
        <artifactId>commons-pool2</artifactId>
        <version>2.4.1</version>
</dependency>
<!-- https://mvnrepository.com/artifact/commons-logging/commons-logging -->
<dependency>
        <groupId>commons-logging</groupId>
        <artifactId>commons-logging</artifactId>
        <version>1.2</version>
</dependency>
```

(2) 커넥션 풀 초기화 서블릿 클래스

■ 실제 커넥션을 생성할 ConnectionFactory를 생성한다.

■ 커넥션 풀로 사용할 PoolableConnection을 생성하는 PoollableConnectionFactory를 생성한다.

■ 커넥션 풀의 설정 정보를 생성한다.

■ 커넥션 풀을 사용할 JDBC 드라이버를 등록한다.

[chap14/src/jdbc/DBCPInit.java]

```
01    package jdbc;
02
03    import java.sql.DriverManager;
04    import java.util.Properties;
05
06    import javax.servlet.ServletException;
07    import javax.servlet.http.HttpServlet;
08
09    import org.apache.commons.dbcp2.BasicDataSource;
```

```
10  import org.apache.commons.dbcp2.BasicDataSourceFactory;
11  import org.apache.commons.dbcp2.ConnectionFactory;
12  import org.apache.commons.dbcp2.DriverManagerConnectionFactory;
13  import org.apache.commons.dbcp2.PoolableConnection;
14  import org.apache.commons.dbcp2.PoolableConnectionFactory;
15  import org.apache.commons.dbcp2.PoolingDriver;
16  import org.apache.commons.pool2.ObjectPool;
17  import org.apache.commons.pool2.impl.GenericObjectPool;
18  import org.apache.commons.pool2.impl.GenericObjectPoolConfig;
19
20  public class DBCPInit extends HttpServlet {
21
22          @Override
23          public void init() throws ServletException {
24                  loadJDBCDriver();
25                  initConnectionPool();
26          }
27
28          private void loadJDBCDriver() {
29                  try {
30                          Class.forName("com.mysql.jdbc.Driver");
31                  } catch (ClassNotFoundException ex) {
32                          throw new RuntimeException("fail to load JDBC Driver", ex);
33                  }
34          }
35
36          private void initConnectionPool() {
37                  try {
38                          String jdbcUrl = "jdbc:mysql://localhost:3306/chap14?" +
39  "useUnicode=true&characterEncoding=utf8";
40                          String username = "jspexam";
41                          String pw = "jsppw";
42
43                          ConnectionFactory connFactory = new
44  DriverManagerConnectionFactory(jdbcUrl, username, pw);
45
46                          PoolableConnectionFactory poolableConnFactory = new
47  PoolableConnectionFactory(connFactory, null);
48                          poolableConnFactory.setValidationQuery("select 1");
49
50                          GenericObjectPoolConfig poolConfig = new GenericObjectPoolConfig();
51                          poolConfig.setTimeBetweenEvictionRunsMillis(1000L * 60L * 5L);
52                          poolConfig.setTestWhileIdle(true);
53                          poolConfig.setMinIdle(4);
54                          poolConfig.setMaxTotal(50);
55
56                          GenericObjectPool<PoolableConnection> connectionPool = new
57  GenericObjectPool<>(poolableConnFactory,
58                                                  poolConfig);
59                          poolableConnFactory.setPool(connectionPool);
60
61                          Class.forName("org.apache.commons.dbcp2.PoolingDriver");
62                          PoolingDriver driver = (PoolingDriver)
63  DriverManager.getDriver("jdbc:apache:commons:dbcp:");
64                          driver.registerPool("chap14", connectionPool);
65                  } catch (Exception e) {
66                          throw new RuntimeException(e);
67                  }
68          }
69  }
```

(3) 커넥션 풀 초기화 서블릿 설정

■ 아래와 같은 코드를 web.xml 파일에 추가해주면 된다.

[chap14/WebContent/WEB-INF/web.xml]

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                  http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
         version="3.1">

         <servlet>
                  <servlet-name>mysqlDriverLoader</servlet-name>
                  <servlet-class>jdbc.MySQLDriverLoader</servlet-class>
                  <load-on-startup>1</load-on-startup>
         </servlet>

         <servlet>
                  <servlet-name>DBCPInit</servlet-name>
                  <servlet-class>jdbc.DBCPInit</servlet-class>
                  <load-on-startup>1</load-on-startup>
         </servlet>

</web-app>
```

(4) 커넥션 풀로부터 커넥션 사용하기

■ JDBC URL: jdbc:apache:common:dbcp:설정파일명

```
// 클래스패스:/pool.jocl 파일에 설정된 풀 사용
DriverManager.getConnection("jdbc:apache:commons:dbcp:/pool");
```

[chap14/WebContent/viewMemberUsingPool.jsp]

```
<%@ page contentType = "text/html; charset=utf-8" %>
<%@ page import = "java.sql.DriverManager" %>
<%@ page import = "java.sql.Connection" %>
<%@ page import = "java.sql.Statement" %>
<%@ page import = "java.sql.ResultSet" %>
<%@ page import = "java.sql.SQLException" %>
<html>
<head><title>회원 목록</title></head>
<body>

MEMBER 테이블의 내용
<table width="100%" border="1">
<tr>
         <td>이름</td><td>아이디</td><td>이메일</td>
</tr>
<%

         Connection conn = null;
         Statement stmt = null;
         ResultSet rs = null;

         try {
                  String jdbcDriver = "jdbc:apache:commons:dbcp:chap14";
```

```
24                      String query = "select * from MEMBER order by MEMBERID";
25                      conn = DriverManager.getConnection(jdbcDriver);
26                      stmt = conn.createStatement();
27                      rs = stmt.executeQuery(query);
28                      while(rs.next()) {
29  %>
30  <tr>
31          <td><%= rs.getString("NAME") %></td>
32          <td><%= rs.getString("MEMBERID") %></td>
33          <td><%= rs.getString("EMAIL") %></td>
34  </tr>
35  <%
36                      }
37          } finally {
38                      if (rs != null) try { rs.close(); } catch(SQLException ex) {}
39                      if (stmt != null) try { stmt.close(); } catch(SQLException ex) {}
40                      if (conn != null) try { conn.close(); } catch(SQLException ex) {}
41          }
42  %>
43  </table>
44
45  </body>
46  </html>
```

(5) 커넥션 풀 속성 설명

■ DBCPInit 클래스를 보면 커넥션 풀을 설정할 때 GenericObjectPoolConfig 클래스를 사용했다. 이 클래스는 커넥션 풀의 크기, 커넥션의 검사 주기 등을 설정할 수 있는 메서드를 제공하고 있다.

메서드	설명	기본값
setMaxTotal(int)	풀이 관리하는 커넥션의 최대 개수를 설정한다.	8
setMaxIdle(int)	커넥션 풀이 보관할 수 있는 최대 유휴 개수를 지정한다.	8
setMinIdle(int)	커넥션 풀이 유지할 최소 유휴 커넥션 개수를 지정한다.	0
setBlockWhenExhausted(boolean)	풀이 관리하는 커넥션이 모두 사용중인 상태에서 커넥션을 요청할 때 풀에 커넥션이 반환될 때까지 대기할지 여부를 지정한다.	true
setMaxWaitMillis(long)	blockWhenExhausted가 true일 때, 최대 대기 시간을 설정한다.	-1L