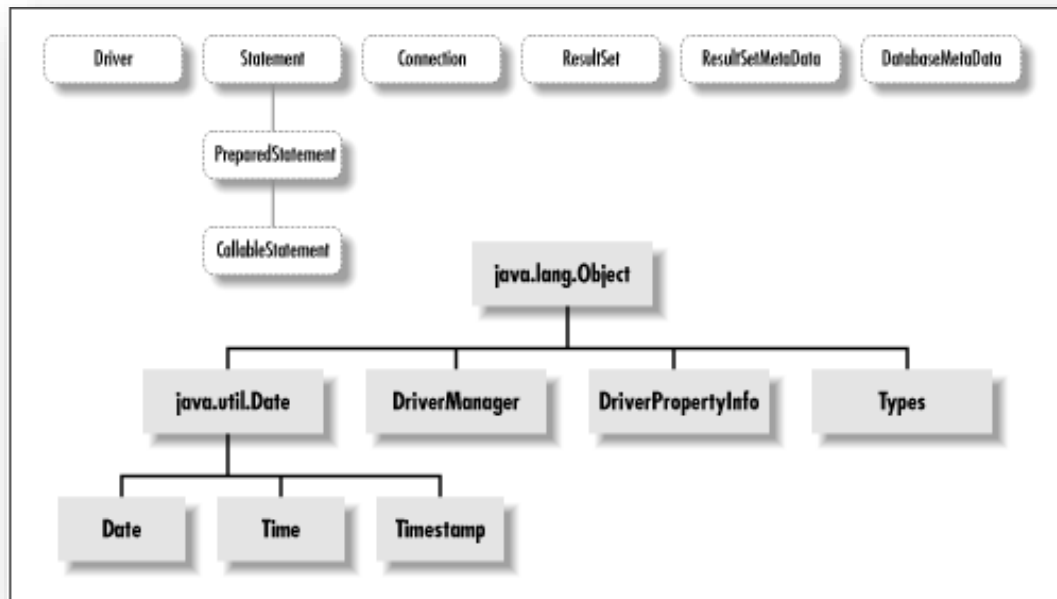


제 12 장

JDBC 프로그래밍

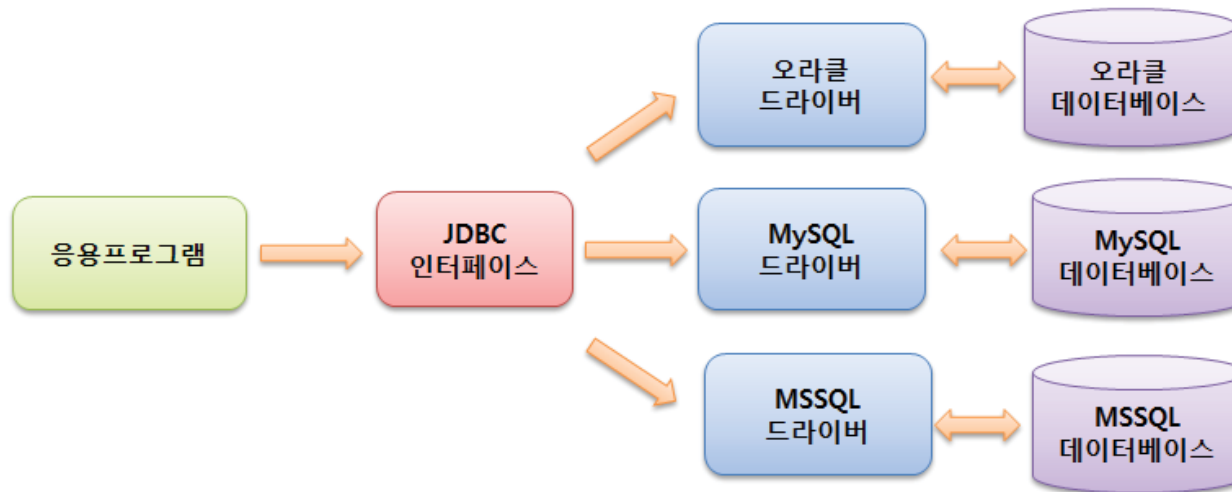
❖ JDBC(Java DataBase Connectivity)

- 자바 프로그램에서 데이터베이스와 연결하여 데이터베이스 관련 작업을 할 수 있도록 해주는 자바 프로그래밍 인터페이스를 위한 API(Application Programming Interface) 규격
 - 데이터베이스를 연결하여 테이블 형태의 자료를 참조
 - SQL 문을 질의
 - SQL 문의 결과를 처리



❖ JDBC 드라이버

- JDBC 인터페이스에 맞추어 해당 DBMS에서 JDBC 관련 API 호출이 가능하도록 관련 인터페이스와 클래스를 구현한 클래스 라이브러리
- JDBC API를 사용하면 DBMS의 종류에 상관없이 데이터베이스 작업을 처리할 수 있음

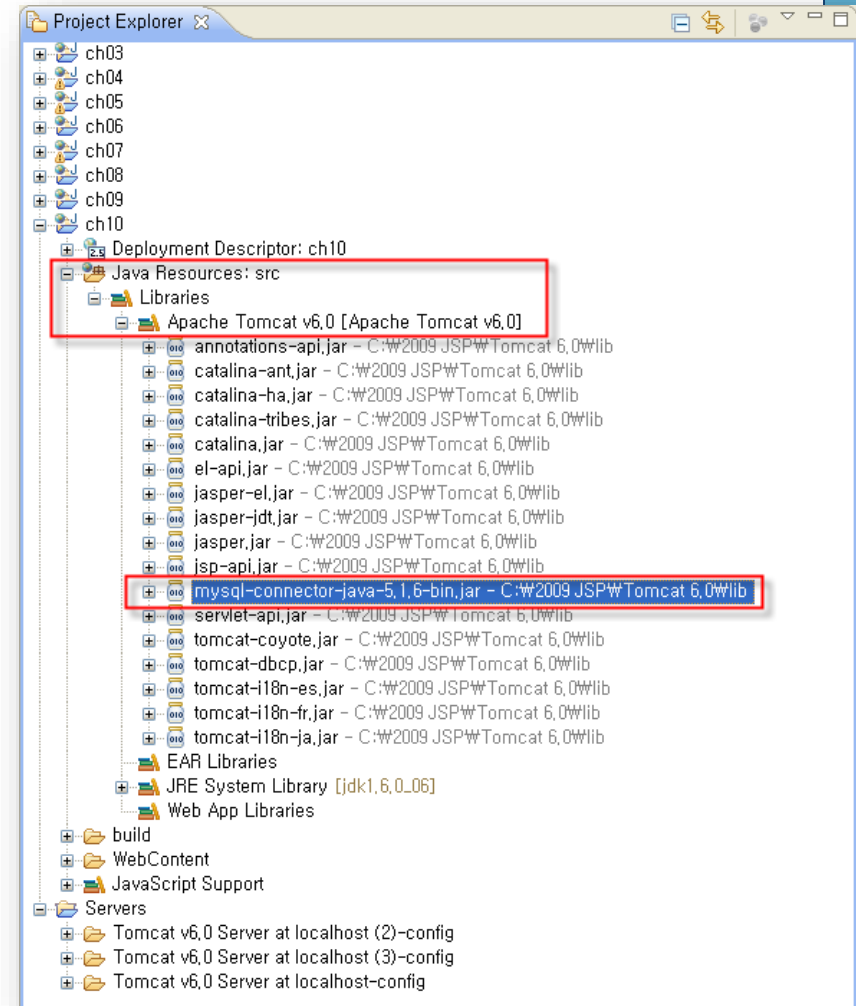


MySQL JDBC 드라이버 위치

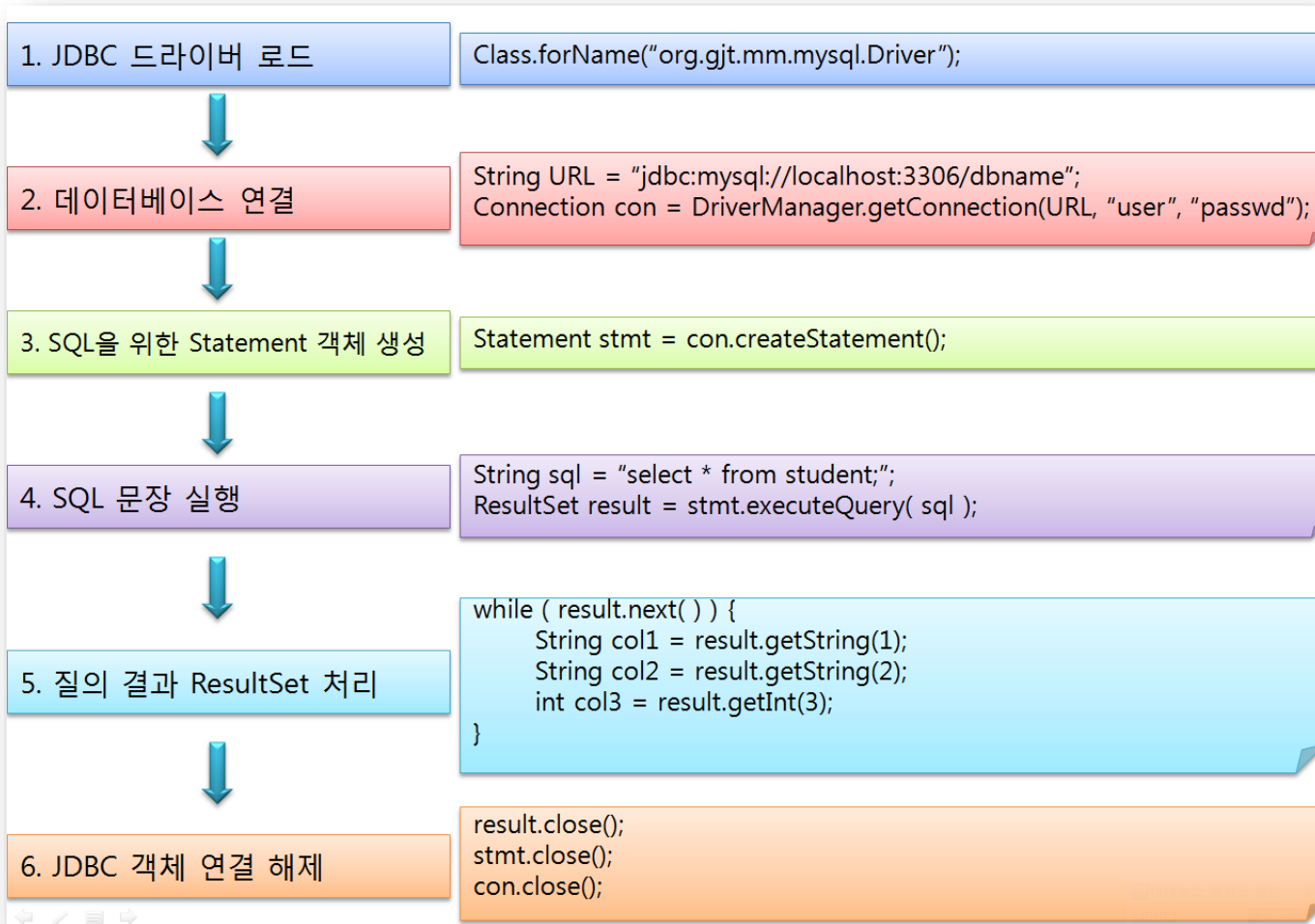
❖ JDBC 드라이버

: [mysql-connector-java-5.x.x-bin.jar] 파일을 복사하는 것으로 완료

- [JDK 설치
폴더]/[jre]/[lib]/[ext]
- [Tomcat 설치 폴더]/[lib]
- [이클립스 프로젝트 하부의
WebContent]/[WEB-INF]/[lib]



❖ JDBC 프로그래밍 절차 6단계



1단계 : JDBC 드라이버 로드



- ❖ JDBC드라이버 로딩 단계에서 드라이버 인터페이스를 구현하는 작업
- ❖ Class.forName() 메소드를 이용하여 JDBC 드라이버 로딩
- ❖ Class.forName(String className);

```
String driverName = "org.gjt.mm.mysql.Driver";  
Class.forName(driverName);
```

또는

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

- ❖ JDBC 드라이버가 로딩되면 자동으로 객체가 생성되고 DriverManager 클래스에 등록
- ❖ JDBC 드라이버 로딩은 프로그램 수행 시 한 번만 필요

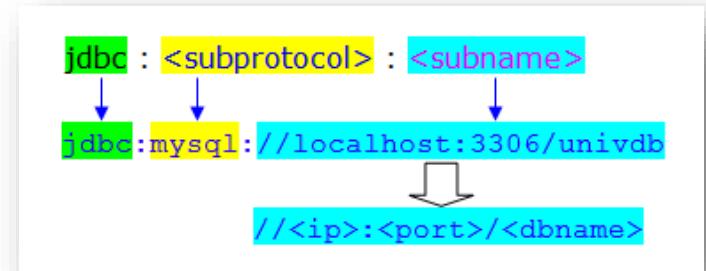
2단계 : 데이터 베이스 연결

❖ Connection 객체 생성하기

- JDBC 드라이버에서 데이터베이스와 연결된 커넥션을 가져오기 위해 **DriverManager** 클래스의 **getConnection()** 메소드 사용
- **DriverManager** 클래스로 **Connection** 객체를 생성할 때 JDBC 드라이버를 검색하고, 검색된 드라이버를 이용하여 **Connection** 객체를 생성한 후 이를 반환

❖ DriverManager.getConnection()

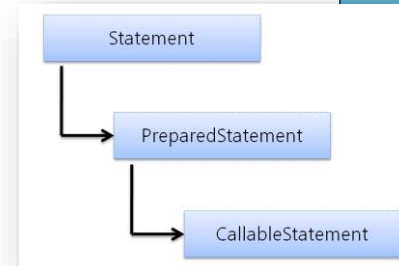
```
String dbURL = "jdbc:mysql://localhost:3306/univdb";  
Connection con = DriverManager.getConnection(dbURL, "root", "");
```



| 표현 요소 | 표현 내용 | 다른 표현 | 의미 |
|-----------------------------------|--------------------------|------------------------------|---|
| <code>//<host or ip></code> | <code>//localhost</code> | <code>//203.214.34.67</code> | MySQL이 실행되는 DBMS 서버를 지정, IP주소 또는 도메인 이름 |
| <code>:<port></code> | <code>:3306</code> | <code>:3308</code> | DBMS 서비스 포트 번호로, 3306으로 서비스된다면 생략 가능 |
| <code>/<dbname></code> | <code>/univdb</code> | <code>/mydb</code> | 접속할 데이터베이스 이름 |

3단계 : SQL을 위한 Statement 객체 생성

- ❖ Statement 객체로 데이터 접근하기
- ❖ 질의 문장인 SQL 문을 추상화 시킨 인터페이스 객체를 생성
 - Statement
 - 정적인 쿼리에 사용
 - 하나의 쿼리를 사용하고 난 후 더는 사용할 수 없음
 - 하나의 쿼리를 끝내면 `close()`를 사용하여 객체를 즉시 해제해야 함
 - 복잡하지 않은 간단한 쿼리문을 사용하는 경우에 좋음
 - `Statement stmt = con.createStatement();`
 - PreparedStatement
 - 동적인 쿼리에 사용
 - 하나의 객체로 여러 번의 쿼리를 실행할 수 있으며, 동일한 쿼리문을 특정 값만 바꾸어서 여러 번 실행해야 할 때, 매개변수가 많아서 쿼리문을 정리해야 할 때 유용
 - 준비 및 처리 단계를 분리
 - `PreparedStatement pstmt = con.prepareStatement(SQL);`
 - CallableStatement
 - PreparedStatement를 확장하고 출력 및 입력/출력 매개변수에 대한 지원을 제공
 - 입력 및 출력 매개변수와 리턴값을 가짐
 - 여러 ResultSet를 리턴할 수 있는 기능을 가짐
 - `CallableStatement cstmt = con.prepareCall(SQL);`



4단계 : SQL 문장 실행



❖ Statement 객체로 데이터 접근하기

- executeQuery(SQL) 메소드로 데이터 조회
- executeUpdate(SQL) 메소드로 데이터 삽입(insert), 수정(update), 삭제(delete)
- execute(SQL) 메소드

```
Connection conn=null;
```

```
..생략..
```

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

```
ResultSet result = stmt.executeQuery("select * from student;");
```

```
int rowCount = stmt.executeUpdate("delete from student where name = '홍길동';");
```

```
stmt.execute("select * from student;");
```

```
ResultSet result = stmt.getResultSet();
```

```
stmt.execute("delete from student where name = '홍길동';");
```

```
int updateCount = stmt.getUpdateCount();
```

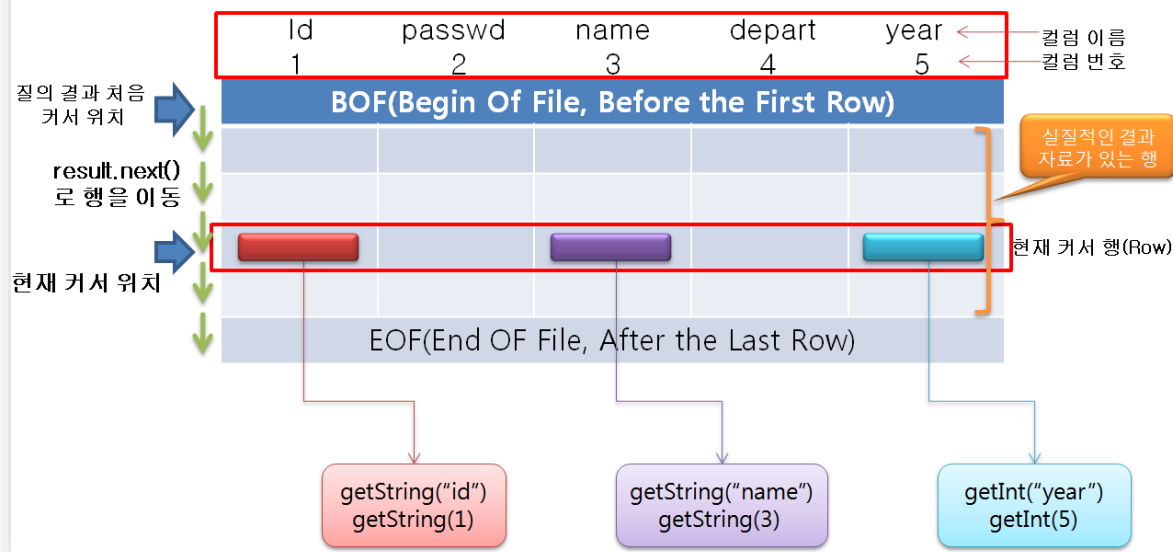
```
stmt.close();
```

❖ ResultSet 객체

- Statement 또는 PreparedStatement 객체로 select 문을 사용하여 얻어온 레코드 값을 테이블 형태로 가진 객체
- ResultSet result = stmt.executeQuery("select * from student;");

```
while ( result.next() ) {
    <%= result.getString(1) %>
    <%= result.getString("passswd") %>
    <%= result.getString(3) %>
    <%= result.getString("depart") %>
    <%= result.getInt(5) %>
}
```

질의 결과인 ResultSet 구조



5단계 : 질의 결과 ResultSet 처리 - 계속

❖ ResultSet 객체의 메소드 종류

| 메소드 | 반환 유형 | 설명 |
|---------------------------|---------|--|
| getXxx(int ColumnIndex) | XXX | 설정된 ColumnIndex(필드 순번)의 필드 값을 설정한 XXX 형으로 가져옵니다. |
| getXxx(String ColumnName) | XXX | 설정된 ColumnName(필드 순번)의 필드 값을 설정한 XXX 형으로 가져옵니다. |
| absolute(int row) | boolean | 설정된 row 행으로 커서를 이동합니다. |
| beforeFirst() | void | 첫 번째 행의 이전으로 커서를 이동합니다. |
| afterLast() | void | 마지막 행의 다음으로 커서를 이동합니다. |
| first() | void | 첫 번째 행으로 커서를 이동합니다. |
| last() | void | 마지막 행으로 커서를 이동합니다. |
| next() | boolean | 다음 행으로 커서를 이동합니다. |
| previous() | boolean | 현재 행의 이전 행으로 커서를 이동합니다. |
| close(): | void | ResultSet 객체를 반환할 때 사용합니다. |

6단계 : JDBC 객체 연결 해제



- ❖ JDBC 프로그래밍의 마지막 단계
 - 이미 사용한 JDBC 객체의 연결을 해제하는 일
 - `con.close();`
 - `result.close();`
 - `pstmt.close();`

```

01 <%@ page contentType="text/html; charset=utf-8"%>
02 <html>
03 <head>
04 <title>Database SQL</title>
05 </head>
06 <body>
07     <form method="post" action="insert01_process.jsp">
08         <p>아이디 : <input type="text" name="id">
09         <p>비밀번호 : <input type="password" name="passwd">
10         <p>이름 : <input type="text" name="name">
11         <p><input type="submit" value="전송">
12     </form>
13 </body>
14 </html>

```

Database SQL

/ch16/insert01.jsp

아이디 : 1

비밀번호 :

이름 : 홍길순

전송

```

id VARCHAR(20) NOT NULL,
passwd VARCHAR(20),
name VARCHAR(30),
PRIMARY KEY (id)

```



JSPBook/WebContent/ch16/dbconn.jsp

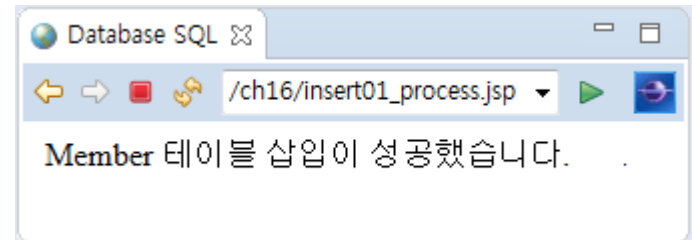
```
01 <% page import="java.sql.*"%>
02 <%
03     Connection conn = null;
04
05     String url = "jdbc:mysql://localhost:3306/JSPBookDB";
06     String user = "root";
07     String password = "1234";
08
09     Class.forName("com.mysql.jdbc.Driver");
10     conn = DriverManager.getConnection(url, user, password);
11 %>
```



```

01 <%@ page contentType="text/html; charset=utf-8"%>
02 <%@ page import="java.sql.*"%>
03 <html>
04 <head>
05 <title>Database SQL</title>
06 </head>
07 <body>
08     <% include file="dbconn.jsp" %>
09     <%
10         request.setCharacterEncoding("utf-8");
11
12         String id = request.getParameter("id");
13         String passwd = request.getParameter("passwd");
14         String name = request.getParameter("name");
15
16         Statement stmt = null;
17

```



실습3)-계속



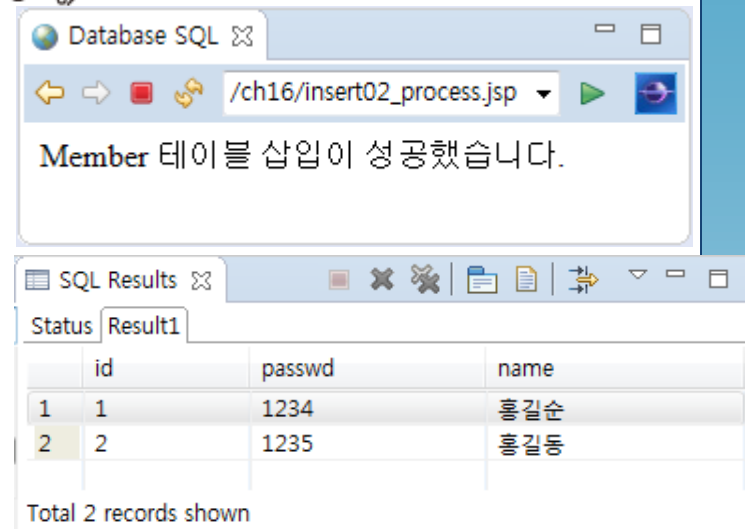
```
18         try {
19             String sql = "INSERT INTO Member(id, passwd, name) VALUES('" +
                id + "',''" + passwd + "',''" + name + "')";
20             stmt = conn.createStatement();
21             stmt.executeUpdate(sql);
22             out.println("Member 테이블 삽입이 성공했습니다.");
23         } catch (SQLException ex) {
24             out.println("Member 테이블 삽입이 실패했습니다.<br>");
25             out.println("SQLException: " + ex.getMessage());
26         } finally {
27             if (stmt != null)
28                 stmt.close();
29             if (conn != null)
30                 conn.close();
31         }
32     %>
33 </body>
34 </html>
```



```

01 <%@ page contentType="text/html; charset=utf-8"%>
02 <%@ page import="java.sql.*"%>
03 <html>
04 <head>
05 <title>Database SQL</title>
06 </head>
07 <body>
08     <%@ include file="dbconn.jsp" %>
09     <%
10         request.setCharacterEncoding("utf-8");
11
12         String id = request.getParameter("id");
13         String passwd = request.getParameter("passwd");
14         String name = request.getParameter("name");
15
16         PreparedStatement pstmt = null;

```



The screenshot shows a web browser window with the URL `/ch16/insert02_process.jsp`. The page displays the message "Member 테이블 삽입이 성공했습니다." (Member table insertion was successful). Below the browser window, an "SQL Results" window is open, showing a table with two columns: "id" and "passwd", and a "name" column. The table contains two rows of data: (1, 1234, 홍길순) and (2, 1235, 홍길동). The status bar indicates "Total 2 records shown".

| | id | passwd | name |
|---|----|--------|------|
| 1 | 1 | 1234 | 홍길순 |
| 2 | 2 | 1235 | 홍길동 |

Total 2 records shown



```
18     try {
19         String sql = "insert into member(id, passwd, name) values(?,?,?)";
20         pstmt = conn.prepareStatement(sql);
21         pstmt.setString(1, id);
22         pstmt.setString(2, passwd);
23         pstmt.setString(3, name);
24         pstmt.executeUpdate();
25         out.println("Member 테이블 삽입이 성공했습니다.");
26     } catch (SQLException ex) {
27         out.println("Member 테이블 삽입이 실패했습니다.<br>");
28         out.println("SQLException: " + ex.getMessage());
29     } finally {
30         if (pstmt != null)
31             pstmt.close();
32         if (conn != null)
33             conn.close();
34     }
35     %>
36 </body>
37 </html>
```



```

01 <%@ page contentType="text/html; charset=utf-8"%>
02 <%@ page import="java.sql.*"%>
03 <html>
04 <head>
05 <title>Database SQL</title>
06 </head>
07 <body>
08     <% include file="dbconn.jsp" %>
09     <table width="300" border="1">
10         <tr>
11             <th>아이디</th>
12             <th>비밀번호</th>
13             <th>이름</th>
14         </tr>
15         <%
16             ResultSet rs = null;
17             Statement stmt = null;    또는 PreparedStatement psmt = null;
18

```

| 아이디 | 비밀번호 | 이름 |
|-----|------|-----|
| 1 | 1234 | 홍길순 |
| 2 | 1235 | 홍길동 |

실습5)-계속



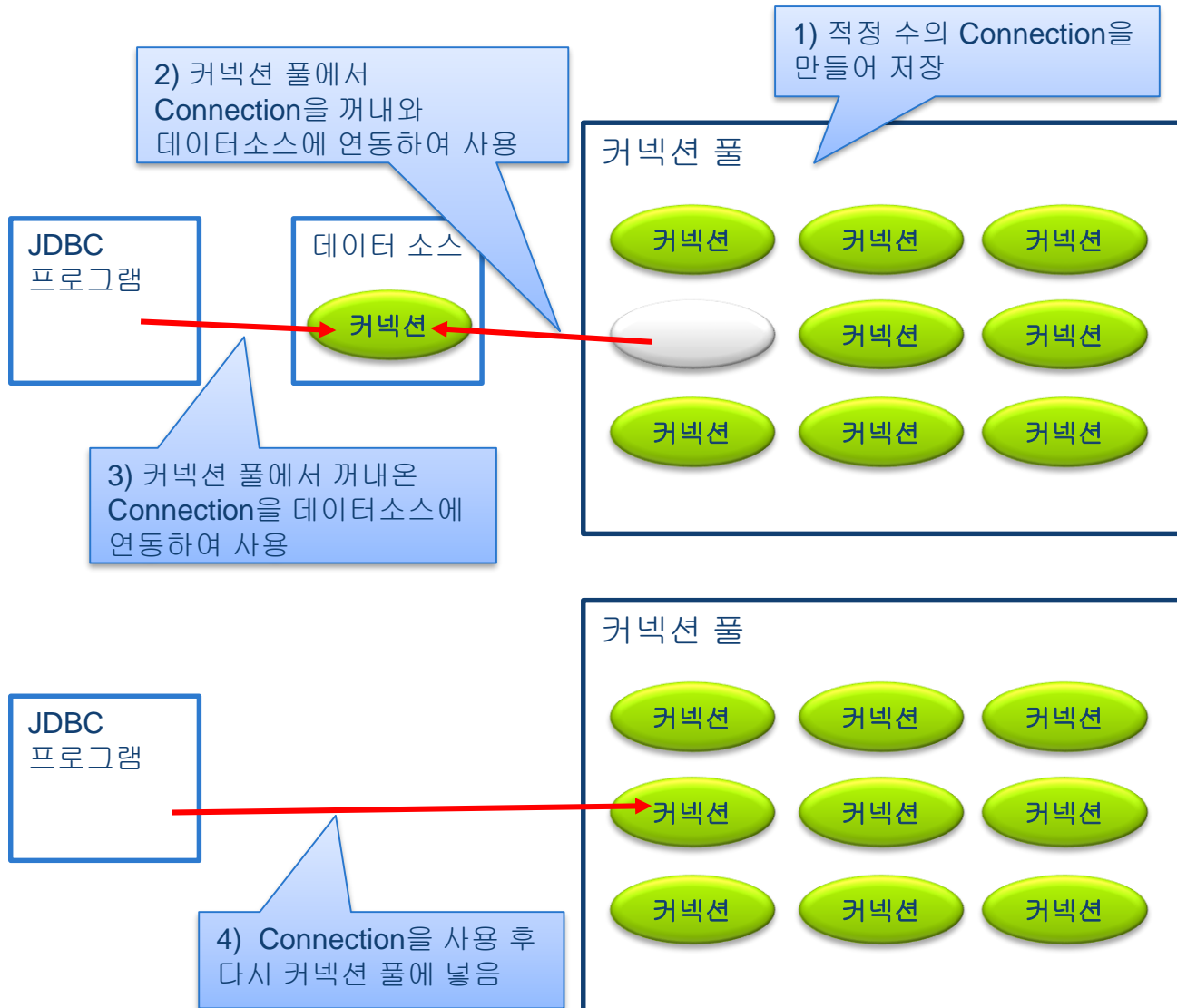
```
19         try {
20             String sql = "select * from member";
21             stmt = conn.createStatement();    또는 pstmt = conn.prepareStatement(sql);
22             rs = stmt.executeQuery(sql);      rs = pstmt.executeQuery();
23
24             while (rs.next()) {
25                 String id = rs.getString("id");
26                 String pw = rs.getString("passwd");
27                 String name = rs.getString("name");
28             }
29             <tr>
30                 <td>X%=id%</td>
31                 <td>X%=pw%</td>
32                 <td>X%=name%</td>
33             </tr>
34             <%
35                 }
36             } catch (SQLException ex) {
37                 out.println("Member 테이블 호출이 실패했습니다.<br>");
38                 out.println("SQLException: " + ex.getMessage());
39             } finally {
40                 if (rs != null)
41                     rs.close();
42                 if (stmt != null)
43                     stmt.close();
44                 if (conn != null)
45                     conn.close();
46             }
47         }
48     </table>
49 </body>
50 </html>
```



- ❖ 데이터베이스 커넥션 풀(Database Connection Pool) 관리 기법
 - 데이터베이스 연결 작업은 서버의 자원을 이용하는 작업으로, 계속적으로 발생한다면 시스템에 상당히 부하를 주는 요소
 - 일관된 커넥션 관리가 필요

- ❖ 커넥션 풀(Connection Pool)
 - 미리 여러 개의 데이터베이스 커넥션을 만들어 확보해 놓고 클라이언트의 요청이 있는 경우, 커넥션을 서비스해 주는 커넥션 관리 기법

커넥션 풀-계속





❖ 자카르타 DBCP

- 자카르타 공통 컴포넌트(Jakarta-Commons Component)로 구성
- 톰캣을 설치했다면 이미 자카르타 DBCP가 설치됨
- 톰캣의 설치 폴더 [Tomcat 설치폴더]/[lib] 하부
 - 파일 [tomcat-dbcp.jar]

파일 [server.xml]의 컨텍스트에 리소스 설정

❖ 컨텍스트에서 컨넥션 풀에서 이용할 리소스를 추가

- 데이터베이스의 접속에 필요한 드라이버 이름, 사용자이름, 암호, URL 등을 기술

<Context>

- path : URL 호출시 사용될 이름
- docBase : 실제 웹 어플리케이션이 위치한 폴더명
- debug : 로그 작성 레벨
- reloadable : 톰캣서버의 재시작 없이 수정된 내용을 불러오기 위한 옵션
- crossContext : myapp이외의 Context에서도 사용 가능하도록 하는 옵션

```
<Context docBase="ch10" path="/ch10" reloadable="true"
source="org.eclipse.jst.jee.server:ch10">

  <Resource name="jdbc/mysql" auth="Container" type="javax.sql.DataSource"
    maxActive="100" maxIdle="30" maxWait="10000"
    username="root" password="" driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost:3306/univdb?autoReconnect=true"/>

</Context>

</host>
```

<Resource>

- name : Resource 명칭
- auth : Resource 관리자, 여기서는 톰캣 컨테이너가 관리자
- type : Resource 의 형태 지정, 데이터소스 형태를 지정
- maxActive : 최대 연결 가능한 Connection 숫자를 지정
- maxIdle : Connection pool 유지를 위해 최대 대기 connection 숫자
- maxWait : Connection 재 사용을 위해 대기해야 하는 최대 시간(단위:ms, 1000 밀리초 = 1 초)
- username : DB접속 계정
- password : DB접속 암호
- driverClassName : DB와 접속하기 위한 driver 클래스 지정. 본 환경은 mysql
- url : 접속한 DB가 위치한 서버명, 포트명, 사용할 데이터베이스명 지정

설정 파일 web.xml에서 리소스 참조방법 등록

❖ 설정 파일 [web.xml]

- [server.xml] 파일에 이미 등록한 리소스를 찾는 방법을 기술
- DB Connection 생성을 위한 환경 설정
- 이클립스에서 현재 프로젝트 폴더 하부 [WebContent]/[WEB-INF]에 있는 파일 [web.xml] 수정

```
<description>MySQL Test App</description>
```

```
<resource-ref>
```

```
  <description>DB Connection</description>
```

```
  <res-ref-name>jdbc/mysql</res-ref-name>
```

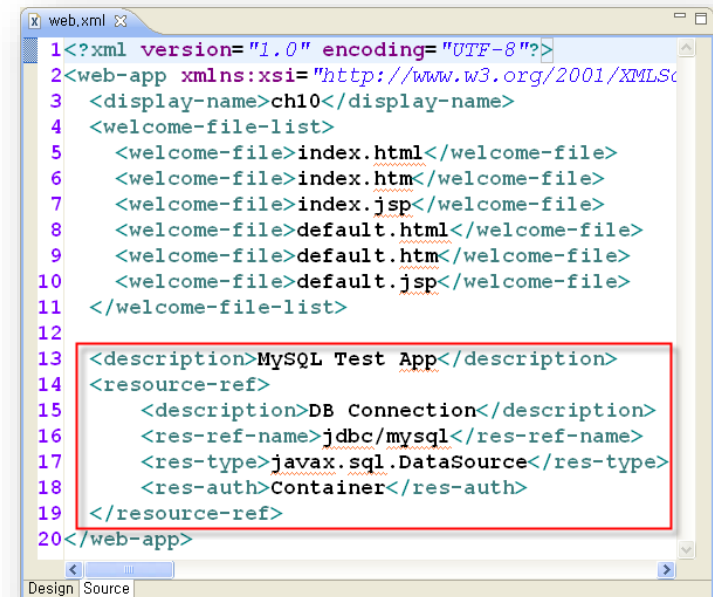
```
  <res-type>javax.sql.DataSource</res-type>
```

```
  <res-auth>Container</res-auth>
```

```
</resource-ref>
```

<resource-ref>

- description : 참조할 resource에 대한 설명
- res-ref-name : 참조할 resource의 명칭. server.xml에 정의된 이름과 일치해야함
- res-type : 참조할 resource 형태로 DataSource를 지정함
- res-auth : 참조할 resource를 관리할 주체로 톱캣 컨테이너를 지정함





[과제] 강의 영상 자료에 있는 실습1~5의 예제를 직접 프로그램 한 후 해당 소스와 출력 결과를 편집하여 제출하세요.

- ❖ 제출방법 : ppt파일에 출력 결과 편집 후 제출
- ❖ 제출마감일 : 2021년 05월31일 월요일 23시 55분까지