9장-1. 커넥션풀 (Connection Pool)

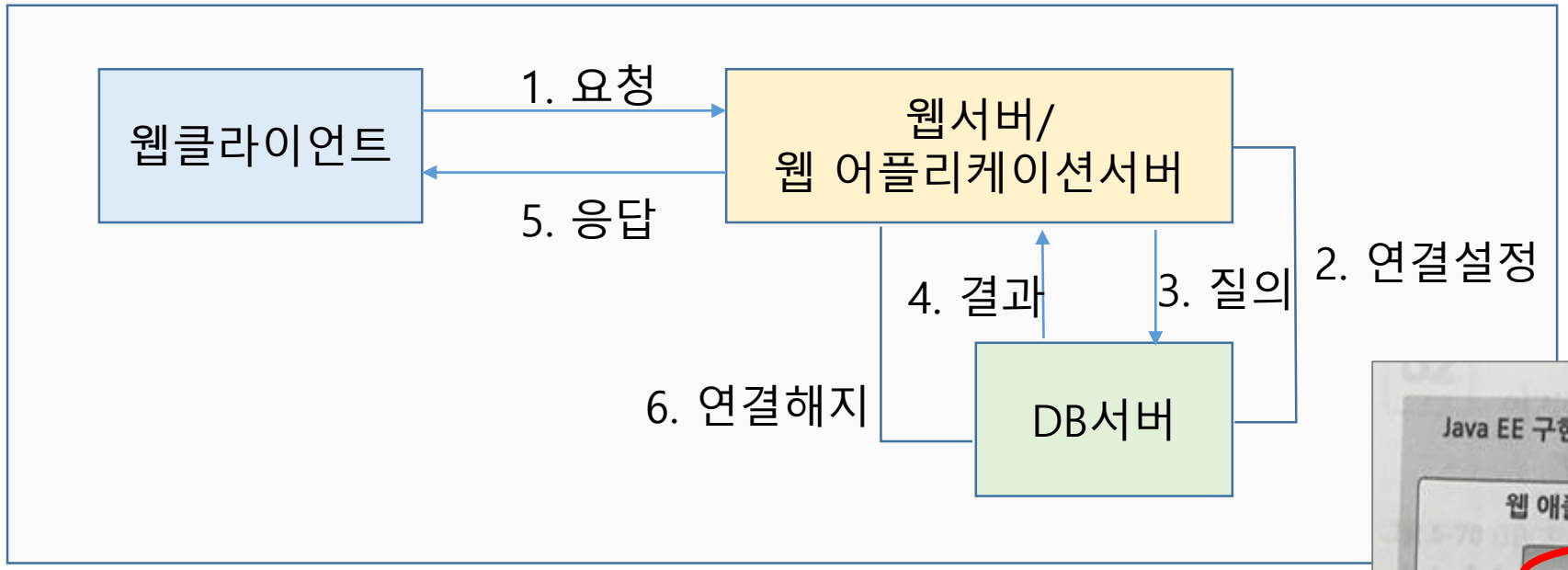
컴퓨터정보계열
김진숙

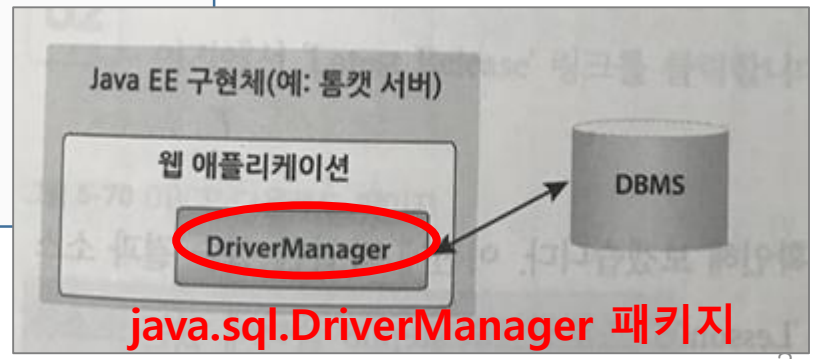
이 장에서 공부할 것

- DBCP API를 사용한 커넥션 풀 설정
- DBCP API를 사용한 커넥션 풀 사용

DBCP API를 이용한 커넥션 풀 설정

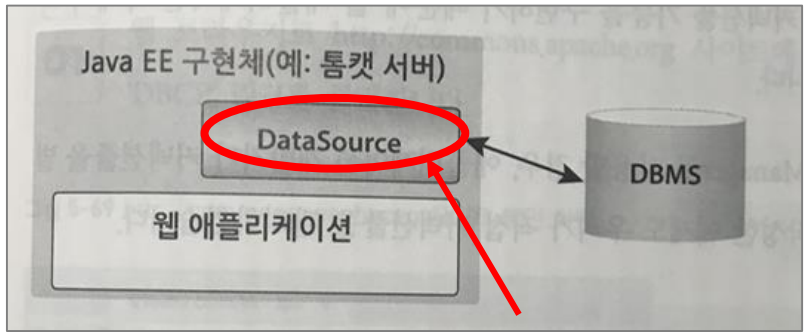
- 일반 DB 연결 프로그램의 **문제점**
  - 클라이언트의 요청 시 DriverManger객체가 Connection을 통해 DB작업을 수행한 뒤 Connection객체를 해제하는 과정 반복
  - Connection객체는 생성, 초기화, 회수 시 **많은 시스템 자원 요구**





Database Connection Pool

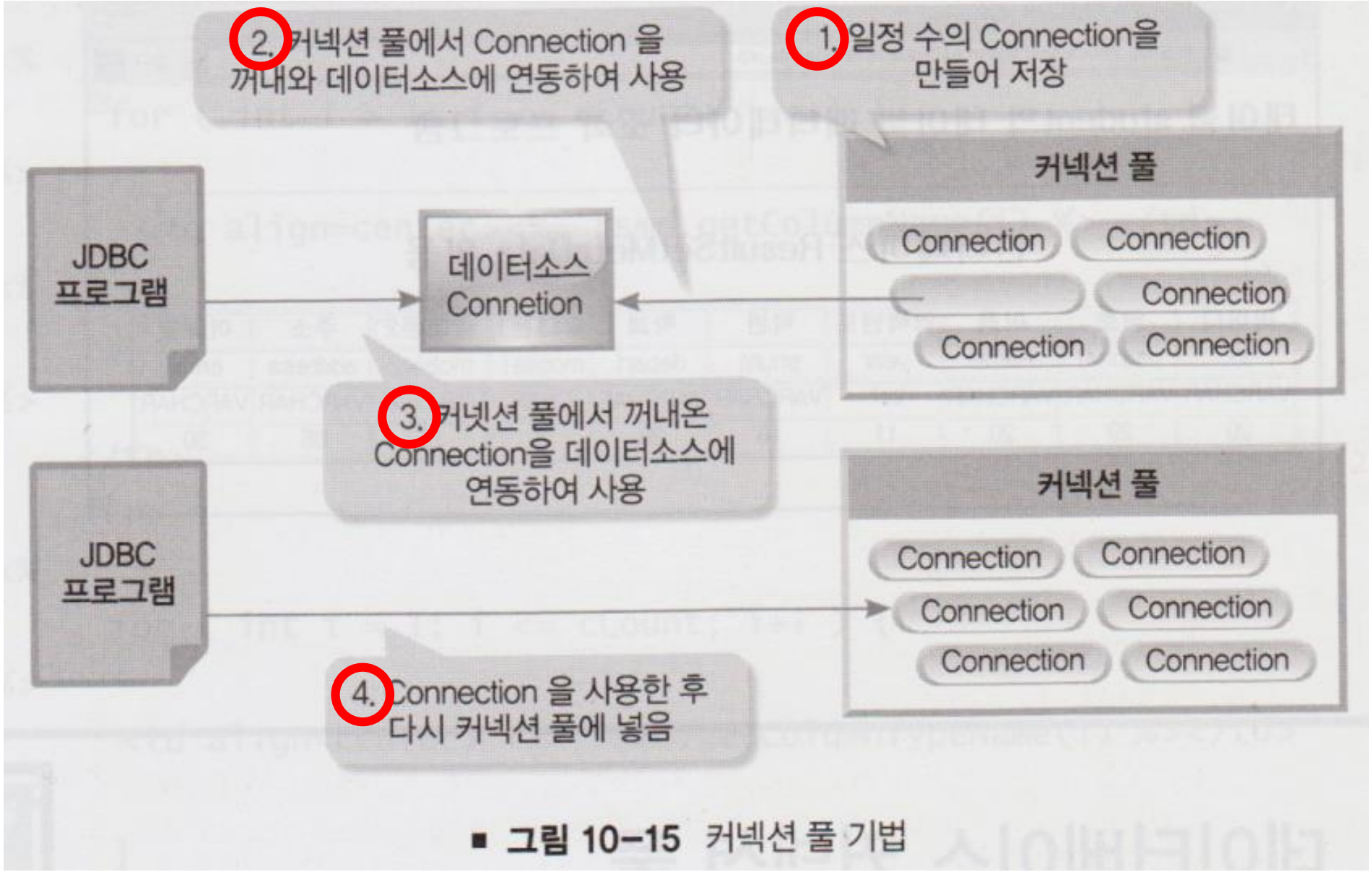
- 커넥션 풀(CP: Connection Pooling)
  - **서버**가 데이터베이스의 **연결과 해제를 직접 관리**
- 커넥션 풀의 장점
  - 빠른 실행 속도
    - 서버가 가동될 때 풀에 커넥션을 생성해 둔다.
    - 커넥션 생성과 해지에 시간, 자원이 소비되지 않는다.
    - 재사용하므로 생성되는 커넥션 수가 많지 않다.
  - 많은 접속자 수 처리 가능
    - 한 번에 생성될 수 있는 커넥션 수를 제어하므로 **접속자 수**가 많아도 처리 가능하다.
  - 커넥션 풀을 이용하면 **동시 사용자 수가 많고, DB작업이 많은** 시스템에 효율적이다.



Javax.sql.DataSourse 패키지

https://docs.oracle.com/javase/8/docs/api/index.html?java/sql/package-summary.html

Database Connection Pool



■ **그림 10-15** 커넥션 풀 기법

커넥션 풀(CP: Connection Pooling)

- DBCP에 사용된 라이브러리 commons-dbcp, commons-pool, commons-collections는 **톰캣 6.0** 부터 **tomcat-dbcp.jar**로 통합
- 여기서는 **tomcat-dbcp**에 대해 다룰 예정
  - common-dbcp와 Tomcat-dbcp는 사용하는 Parameter가 유사하나 다른 API임
  - 설정 시 반드시 공식 사이트의 매뉴얼을 확인하고 적용해야 함

관련 참고 사항 사이트

http://tomcat.apache.org/tomcat-9.0-doc/jdbc-pool.html

http://tomcat.apache.org/tomcat-9.0-doc/jndi-resources-howto.html

https://docs.oracle.com/javase/8/docs/api/

DataSource

- 커넥션 풀의 **Connection 관리 객체**
  - DriverManager 역할(Connection 객체)을 대체
- JNDI(Java Naming and Directory Interface)를 통해 사용
- DataSource 객체를 통해 필요한 Connection 획득, 반납 작업
- **javax.sql.DataSource**
- DataSource의 장점
  - 코드 이식성 높여줌
  - 코드 관리 용이

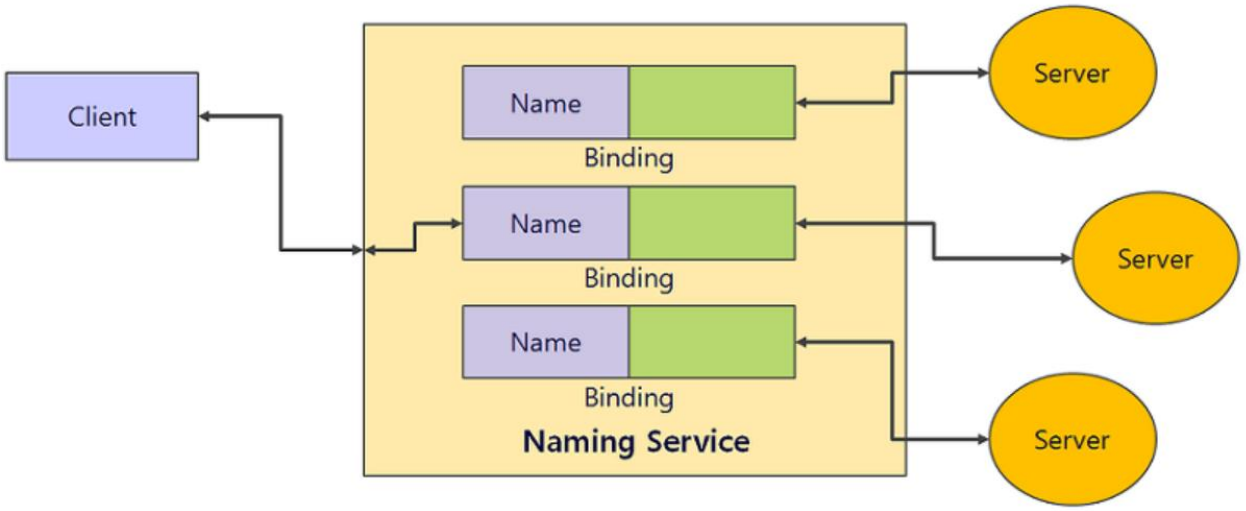
DataSource

- 사용 절차
  1. JNDI Server에서 lookup() 메소드를 통해 DataSource 객체 획득
  2. DataSource 객체의 getConnection()을 통해 커넥션 풀에서 사용 가능한 Connection 을 획득
  3. Connection 객체를 통한 DBMS작업 수행
  4. 모든 작업이 끝나면 DataSource 객체를 통해 Connection 반납

[참고] JNDI

- JNDI(Java Naming and Directory Interface)는 디렉터리 서비스에서 제공하는 데이터 및 객체를 발견하고 참고하기 위한 자바 API

- **Naming & Directory 서비스**
  - 분산 환경에서 자원을 연결하는 기능
  - 서버나 애플리케이션에서 서비스하고자 하는 자원을 이 Naming & Directory 서버에 이름값과 실제 자원을 연결하여 등록
  - 자원을 이용하고자 하는 다른 애플리케이션에서 Naming & Directory 서버에 접근하여 이름 값만을 가지고 자원을 연결하여 이용
  - 대표적인 예 : DNS 서버



[Naming Service 구조]

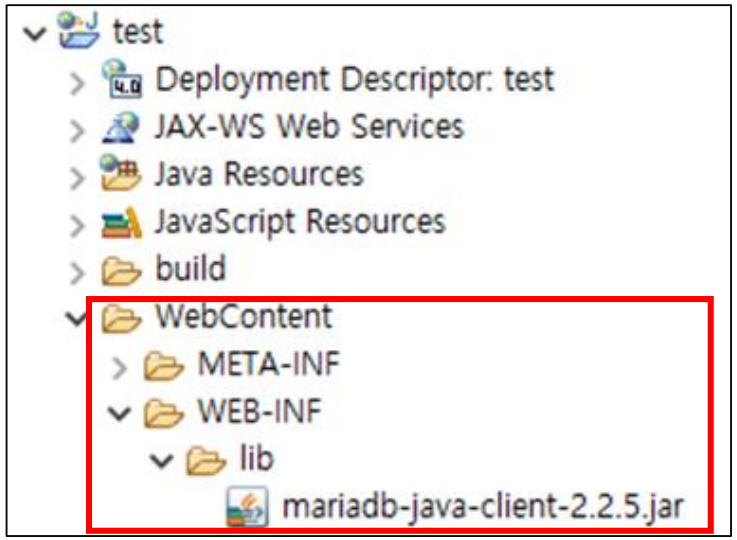
Database Connection Pool 설정

- DBCP API를 사용해서 커넥션 풀을 사용하기 위한 절차
  1. DBCP API관련 설치 파일
     - mariadb-java-client-2.7.5.jar (jdbc driver)
  2. DBCP에 관한 정보 설정 위치
     - context.xml
     - web.xml
  3. JSP페이지에서 커넥션 풀 사용하여 코딩

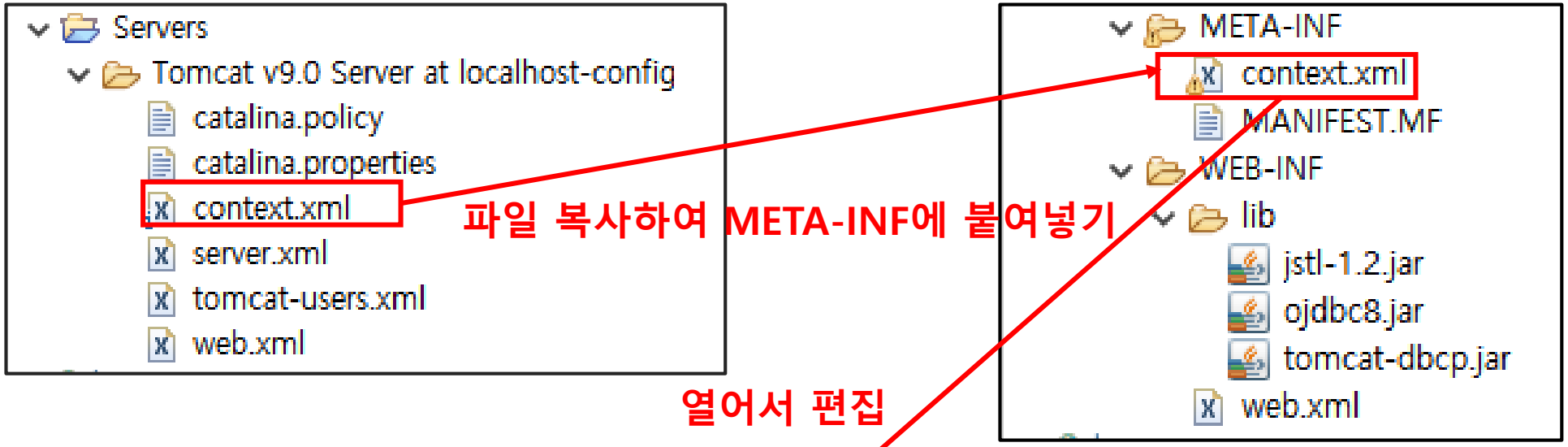
JNDI DataSource 관련 :
http://tomcat.apache.org/tomcat-9.0-doc/jndi-datasource-examples-howto.html

자카르타 DBCP API를 이용한 커넥션 풀 설정

1. jdbc driver 설치 확인



2. DBCP에 관한 정보 설정 – 이클립스 환경



커넥션 정보를 META-INF의 context.xml에 입력할 것

```
<Resource name ="jdbc/jskim"
    auth = "Container"
    type="javax.sql.DataSource"
    driverClassName = "org.mariadb.jdbc.Driver"
    username="jinsook"
    password="1111"
    url="jdbc:mariadb://localhost:3306/jinsookdb"
    maxWait = "5000"
/>
```

- server.xml에 접속 정보를 설정하면 프로젝트 마다 접속 정보 변경 시server.xml파일을 수정해야 함으로 효율적이지 않음
- context.xml파일을 META-INF 폴더에 넣으면 .war 파일에 포함되어 다른 컴퓨터에 포팅 할 때 따로 설정 변경을 할 필요 없어 편리함.

context.xml의 <Resource>의 속성

속성	설명
name	java:comp/env 컨텍스트와 관련되어 생성되는 **자원 이름** jdbc/ 는 필수이고 "/"이후는 임의 작성 가능
auth	자원 관리자 지정. 대부분 컨테이너를 자원 관리자로 기술
type	웹에서 사용할 때 실제로 이용되는 클래스 javax.sql.DataSource jdbc/jsptest 라는 이름을 찾으면 이름에 해당되는 객체형인 javax.sql.DataSource 반환
driverClassName	JDBC 드라이버
username	DB 접속 계정명
password	DB 접속 암호명
url	JDBC 연결 문자열로 오라클의 경우에는 *jdbc:oracle:thin:@localhost:1521/xepdb1*
maxWait	커넥션 풀에 사용 가능한 Connection 이 없을 경우 커넥션 회수를 기다리는 시간으로 ms(millisecond단위)
maxActive	최대 연결 가능한 Connection수 (기본 20개)
maxIdle	Connection pool 유지를 위해 최대 대기 connection 숫자

JSP페이지에서 커넥션 풀 사용

1. Context객체 생성
   - JNDI를 사용하기 위한 객체 생성

```
Context  initCtx = new InitialContext();
```

2. JNDI에 등록된 Naming 자원들을 모두 가져옴
   - JNDI의 모든 이름은 기본적으로 **java:comp/env**에 등록되어 있음
   - 등록된 JNDI 서비스로부터 자원을 검색

```
Context envCtx = (Context)  initCtx.lookup("java:comp/env");
```

   - **java:comp/env : 모든 설정된 엔트리와 자원(Resource)은 JNDI namespace 의 java:comp/env 부분에 놓이게 되고 자원에 대해 접근하려면 lookup("java:comp/env")을 통해 로컬리소스에 접근**

3. DataSource 객체 얻어냄

- java:comp/env 영역에서 "jdbc/jskim"으로 설정된 이름의 DataSource 객체 획득
- lookup() 메소드에서 "jdbc/"는 필수이며 그 뒤에 나오는 이름은 web.xml의 <res-ref-name>과 같아야 함

```
DataSource ds = (DataSource)envCtx.lookup("jdbc/jskim");
```

4. Connection 객체 얻어냄

- ds.getConnection()메소드를 사용해서 커넥션 풀로부터 커넥션 객체를 할당 받음
- 이 Connection 객체는 Container의 DBCP에 의해 관리 됨

```
Connection con = ds.getConnection();
```

2, 3의 과정을 다음과 같이 하나로 통합할 수 있음

```
DataSource ds = (DataSource) initCtx.lookup("java:comp/env/jdbc/jskim");
```

- 기존 JDBC 연동

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@page import="java.sql.*"%>

<%
  //연결 정보와 SQL
  String url = "jdbc:mariadb://localhost:3306/jinsookdb";
  String user = "jinsook";
  String passwd = "1111";

  //1. DB 연동 드라이버 로드
  Class.forName("org.mariadb.jdbc.Driver");

  //2. 연결 객체 생성
  Connection con = DriverManager.getConnection(url, user, passwd);

  //3. 생성된 연결을 통해 SQL문 실행 의뢰 준비
  String sql = "SELECT * FROM LOGIN";
  Statement st = con.createStatement();

  //4. SQL 실행
  ResultSet rs = st.executeQuery(sql);
%>
```

실습1 – DBCP 로 DB연동

```jsp
<%@page import="java.sql.Connection"%>
<%@page import="javax.sql.DataSource"%>
<%@page import="javax.naming.InitialContext"%>
<%@page import="javax.naming.Context"%>

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
  //1. Context 객체 얻기
  Context initCtx = new InitialContext();

  //2.  "java:comp/env" 에 해당하는 객체를 찾아서 envCtx에 삽입
  Context envCtx = (Context) initCtx.lookup("java:comp/env");

  //3. "jdbc/jskim"에 해당하는 객체를 찾아서 ds에 삽입
  DataSource ds = (DataSource) envCtx.lookup("jdbc/jskim");

  //4. 커넥션 풀로 부터 커넥션 객체를 얻어냄
  Connection conn = ds.getConnection();

  String str ="연결 성공";
%>
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>DBCP 연결</title>
</head>
<body>
  <%=str %>
</body>
</html>
```

하나의 문장으로 합칠 수 있음

```jsp
DataSource ds = (DataSource) initCtx.lookup("java:comp/env/jdbc/jskim");
```

실습2 – DBCP로 목록 검색

- jdbc-dbcp 폴더를 작성하여 jdbc-login 폴더안의 파일 복사하여 넣음
- JDBC에서 사용한 테이블 사용(LOGIN)함

```
CREATE TABLE login(
   id VARCHAR(10),
   name VARCHAR(20),
   pwd VARCHAR(20)
);

INSERT INTO login(id, name, pwd) VALUES('gildong', '홍길동', '1111');
INSERT INTO login(id, name, pwd) VALUES('minwoo', '김민우', '2222');
INSERT INTO login(id, name, pwd) VALUES('sohee', '송소희', '3333');

COMMIT;
```

```jsp
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.Connection"%>
<%@page import="javax.sql.DataSource"%>
<%@page import="javax.naming.InitialContext"%>
<%@page import="javax.naming.Context"%>

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
  //1. JNDI를 이용하기 위한 객체 생성
  Context initCtx = new InitialContext();

  //2. 등록된 naming 서비스로부터 등록된 자원을 가져옴
  Context envCtx = (Context) initCtx.lookup("java:comp/env");

  //3. 등록된 자원들 중 원하는(jdbc/jskim) 것을 가져옴
  DataSource ds = (DataSource) envCtx.lookup("jdbc/jskim");

  //4. 해당 서비스를 통해 DBCP에서 Connection을 가져옴
  Connection con = ds.getConnection();

  String sql = "SELECT * FROM LOGIN";
  Statement st = con.createStatement();

  ResultSet rs = st.executeQuery(sql);
%>
```

```jsp
<html>
<head>
  <meta charset="UTF-8">
  <title>로그인 정보</title>

  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"></script>
</head>
<body>
  <div class="container">
  <h1 class="text-center font-weight-bold">로그인 정보</h1>
  <br><br>
  <table class = "table table-hover">
    <tr>
      <th>아이디</th>
      <th>이름</th>
      <th>비밀번호</th>
    </tr>
<%//결과집합 화면 출력
  while(rs.next()){
    String id=rs.getString("ID");
    String name=rs.getString("NAME");
    String pwd=rs.getString("PWD");
%>
    <tr>
      <td><%=id %></td>
      <td><%=name %></td>
      <td><%=pwd %></td>
    </tr>
<%}
  rs.close();
  st.close();
  con.close();

%>

  </table>
  </div>
</body>
</html>
```

이전 코드와 동일

실제 웹 서비스 환경에서 실행하기

- .war 파일 생성하기
  - 해당 프로젝트를 선택하고 [Export]-[WAR file] 실행
    - Destination은 [톰캣홈]₩webapps 로 지정
  - 톰캣 실행 : [톰캣홈]₩bin₩startup.bat
  - 브라우저 실행
    - http://localhost:8080/jspProject/jdbc-login/index.jsp

[과제]

- 지난 과제(게시판관리, 회원관리)를 dbcp를 사용하여 코드를 개선하시오.
  - 각자의 github에 **dbcp**라는 폴더를 만들어서 업로드 하시오.