

Chapter

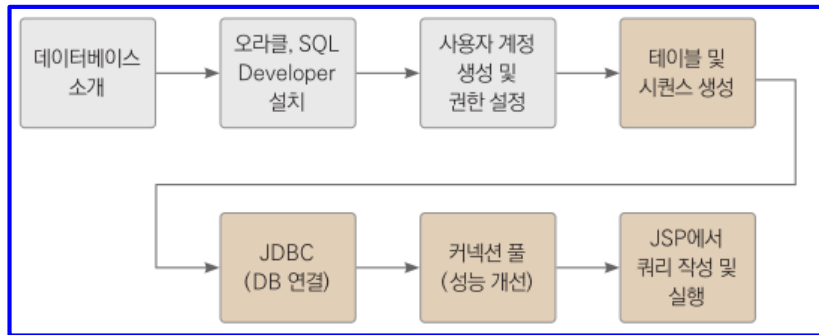
05

데이터베이스(Oracle)

■ 학습 목표

- 데이터 베이스 관리 시스템(DBMS) 중에서 오라클을 설치하고 JDBC API를 이용해서 JSP와 연동하는 방법을 학습합니다.

■ 학습 순서



■ 활용 사례

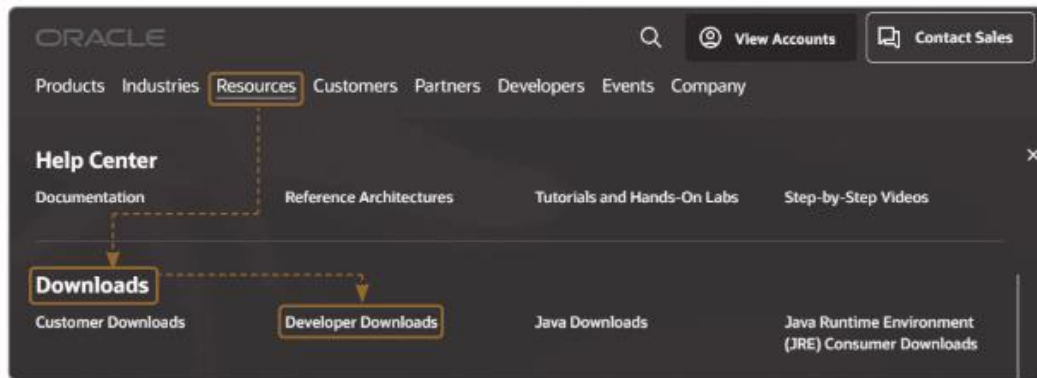
- 사용자 정보부터 상품/판매 정보와 각종 통계까지 수많은 데이터를 저장여 서비스합니다.

■ 데이터베이스란..??

- 우리가 매일 접하게 되는 거의 모든 웹 애플리케이션은 데이터베이스를 사용
- 매일 업데이트되는 뉴스나 날씨 등의 정보는 데이터베이스 없이는 서비스 불가능
- JSP에서는 JDBC(Java Database Connectivity)를 통해 데이터베이스와 연동

■ 오라클 설치

- 설치경로 : C:\01DevelopKits\oracle.exe
- 공식사이트 : <https://oracle.com>



■ 오라클 설치

- Express Edition 21c 버전 다운로드

Oracle Database XE Downloads

Oracle Database 21c Express Edition

Download	Description
Oracle Database 21c Express Edition for Windows x64	(1,967,615,483 bytes - October 08, 2021) [Sha256sum: 939742c3305c466566a55f607638621b6aa7033a183175f6bcd6cffb4

- 압축 해제 후 설치

Oracle Database 21c Express Edition

대상 폴더
설치할 대상 폴더를 선택하십시오.

Oracle Database 21c Express Edition 설치 위치:
C:\%0\DevelopKits\Woradee\%1

1 변경(C)...

InstallShield

< 뒤로(B) 2 다음(N) > 취소

Oracle Database 21c Express Edition

Oracle Database 정보
데이터베이스 비밀번호를 지정하십시오.

이 비밀번호는 SYS, SYSTEM 및 PDBADMIN 계정에 사용됩니다.

데이터베이스 비밀번호 입력
데이터베이스 비밀번호 확인

1 기억하기 쉽게 "123456" 입력

InstallShield

< 뒤로(B) 2 다음(N) > 취소

Oracle Database 21c Express Edition

요약
이 설치에서 적용될 매개변수입니다.

대상 폴더: C:\%0\DevelopKits\Woradee\%1
 Oracle 종: C:\%0\DevelopKits\Woradee\%1\home\XE\%1
 Oracle 기본 위치: C:\%0\DevelopKits\Woradee\%1

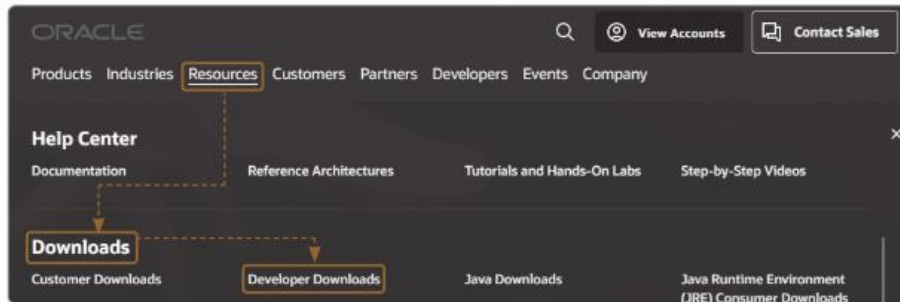
설치를 시작하려면 [설치]를 누르십시오.
 설치 설정을 검토하거나 변경하려면 [뒤로]를 누르십시오. 마법사를 종료하려면 [취소]를 누르십시오.

InstallShield

< 뒤로(B) 설치(I) 취소

■ SQL Developer 설치

- SQL Developer란 오라클 데이터베이스 용 그래픽 기반의 관리 도구
- 테이블 생성, 조회, 추가 등의 작업을 마우스 클릭만으로 쉽게 할 수 있음
- 오라클 홈페이지에서 [Downloads] → [Developer Downloads]를 클릭



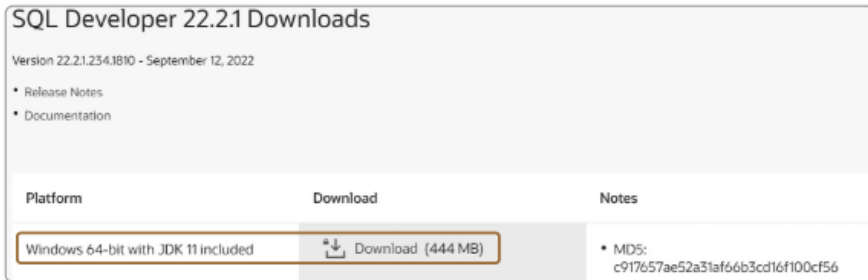
- [Developer Tools] → [SQL Developer]를 클릭



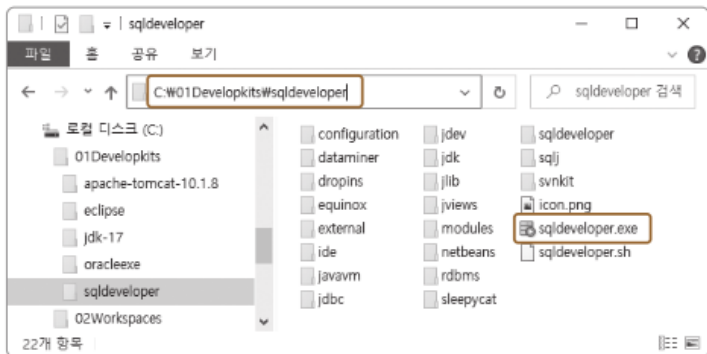
5.3 SQL Developer 설치

■ SQL Developer 설치

- 본인의 OS에 맞는 설치파일을 다운로드



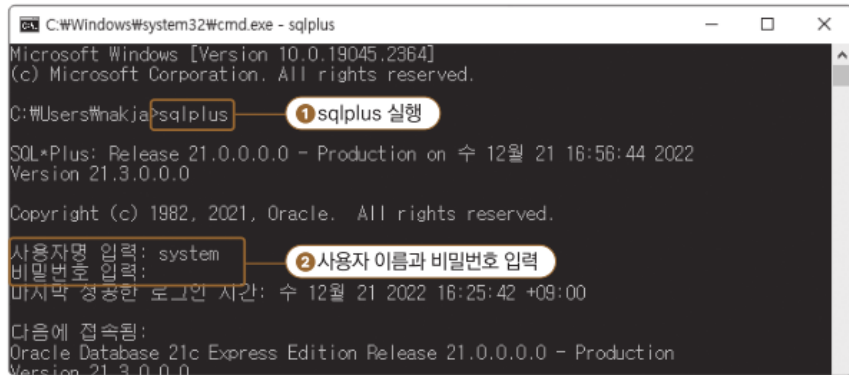
- 압축을 해제한 후 C:\01DevelopKits 폴더로 이동. 압축만 풀면 설치 완료



5.4 사용자 계정 생성 및 권한 설정

■ 사용자 계정 생성 및 권한 설정

- 명령 프롬프트 실행 후 sqlplus 명령 실행 ⇒ system / 123456 입력



```
C:\Windows\system32\cmd.exe - sqlplus
Microsoft Windows [Version 10.0.19045.2364]
(c) Microsoft Corporation. All rights reserved.

C:\Users\nakja>sqlplus
SQL*Plus: Release 21.0.0.0.0 - Production on 수 12월 21 16:56:44 2022
Version 21.3.0.0.0

Copyright (c) 1982, 2021, Oracle. All rights reserved.

사용자명 입력: system
비밀번호 입력:
마지막 성공한 로그인 시간: 수 12월 21 2022 16:25:42 +09:00

다음에 접속함:
Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0
```

- c## 접두어 없이 계정을 생성하기 위해 세션 변경

```
SQL> alter session set "_ORACLE_SCRIPT"=true;
세션이 변경되었습니다.
SQL> create user musthave identified by 1234;
사용자가 생성되었습니다.
```

5.4 사용자 계정 생성 및 권한 설정

■ 사용자 계정 생성 및 권한 설정

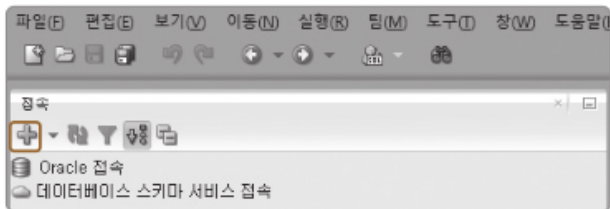
- 역할(Role)을 통한 권한 부여

```
SQL> grant connect, resource to musthave;
```

권한이 부여되었습니다.

■ 생성한 계정을 SQL Developer에 등록하기

- 접속 창의 [+] 버튼 클릭 후 정보입력



■ 테이블 및 시퀀스 생성

- 차후 회원제 게시판 작성을 위한 테이블 생성 및 외래키(Foreign key) 추가
- 일련번호 입력을 위한 시퀀스 생성

■ 테이블 생성1 - member

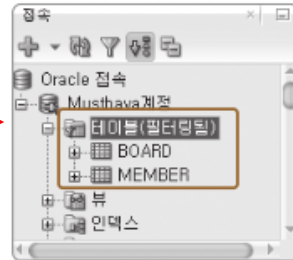
컬럼명	데이터 타입	null 허용	키	기본값	설명
id	varchar2(10)	N	기본키		아이디
pass	varchar2(10)	N			패스워드
name	varchar2(30)	N			이름
regdate	date	N		sysdate	가입 날짜

```
create table member (  
    id varchar2(10) not null,  
    pass varchar2(10) not null,  
    name varchar2(30) not null,  
    regdate date default sysdate not null,  
    primary key (id)  
);
```

■ 테이블 생성2 - board

컬럼명	데이터 타입	null 허용	키	기본값	설명
num	number	N	기본키		일련번호, 기본키
title	varchar2(200)	N			게시물의 제목
content	varchar2(2000)	N			내용
id	varchar2(10)	N	외래키		작성자의 아이디. member 테이블의 id를 참조하는 외래키
postdate	date	N		sysdate	작성일
visitcount	number(6)	Y			조회수

테이블 생성 결과



```
create table board (
    num number primary key,
    title varchar2(200) not null,
    content varchar2(2000) not null,
    id varchar2(10) not null, ①
    postdate date default sysdate not null,
    visitcount number(6)
);
```

■ 외래키로 테이블 사이의 관계 설정

- board 테이블의 id 컬럼이 member 테이블의 id 컬럼을 참조하도록 외래키를 생성
- 외래키를 통해 member 테이블에 등록되지 않은 id로는 게시물을 작성할 수 없음

```
alter table board
  add constraint board_mem_fk foreign key (id)
  references member (id);
```

■ 시퀀스 생성

- 순차적으로 증가하는 순번을 반환하는 데이터베이스 객체

```
create sequence seq_board_num
  increment by 1  ❶ 1씩 증가
  start with 1    ❷ 시작값 1
  minvalue 1      ❸ 최솟값 1
  nomaxvalue      ❹ 최댓값은 무한대
  nocycle         ❺ 순환하지 않음
  nocache;        ❻ 캐시 안 함
```

■ 테이블 스페이스 설정

- 테이블 스페이스란 디스크 공간을 소비하는 테이블(table)과 뷰(view) 같은 데이터베이스 객체들이 저장되는 장소
- 즉, 데이터를 저장하는 물리적인 공간
- musthave 계정이 사용하는 테이블 스페이스 확인

```
select username, default_tablespace from dba_users
where username in upper('musthave');
```

- USERS 테이블 스페이스에 데이터를 입력할 수 있도록 5m의 용량을 할당

```
alter user musthave quota 5m on users;
```

■ 동작확인을 위한 더미데이터 입력

```
insert into member (id, pass, name) values ('musthave', '1234', '머스트해브');
insert into board (num, title, content, id, postdate, visitcount)
values (seq_board_num.nextval, '제목1입니다', '내용1입니다', 'musthave',
sysdate, 0);
```



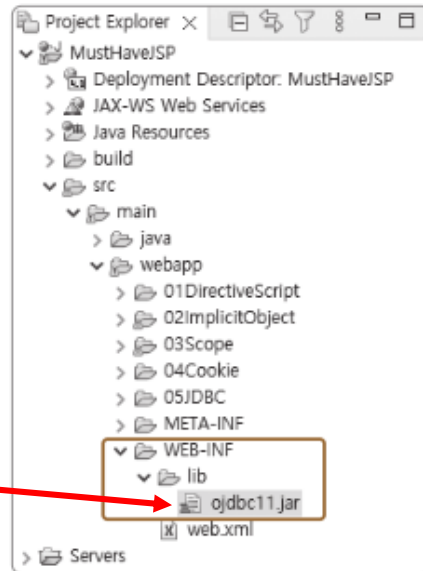
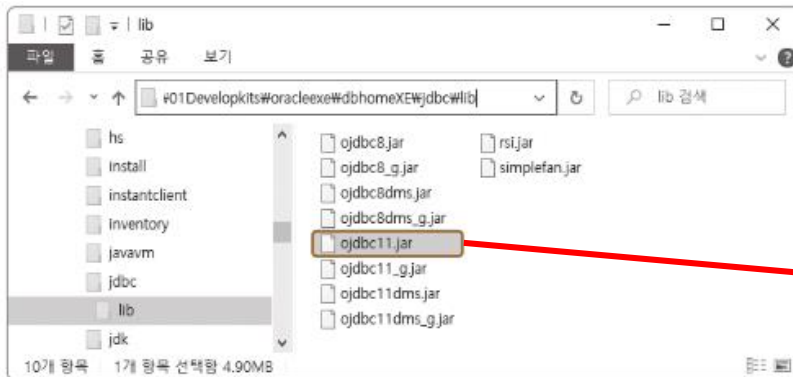
5.6 JDBC 설정 및 데이터베이스 연결

■ JDBC란..??

- JDBC(Java Database Connectivity)란 자바로 데이터베이스 연결 및 관련 작업을 할 때 사용하는 API
- 사용을 위해서는 JDBC 드라이버를 이클립스 프로젝트에 설정해야 함

■ JDBC 드라이버 설정

- 드라이버 파일 경로
 - C:\01DevelopKits\oracleexe\dbhomeXE\jdbc\lib
 - 파일명 : ojdbc11.jar





5.6 JDBC 설정 및 데이터베이스 연결

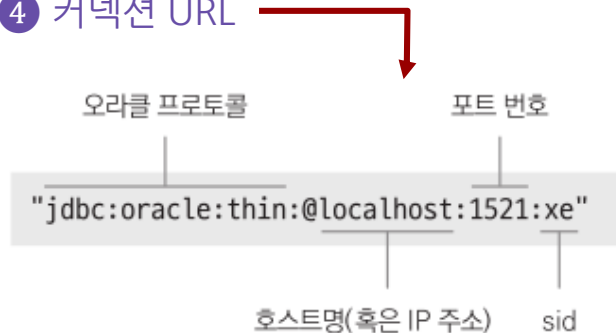
■ 연결 관리 클래스 작성

- Java Resources 하위의 common 패키지에 JDBCConnect 클래스 생성

예제 5-6] Java Resources/common/JDBCConnect.java

```
public class JDBCConnect {  
    public Connection con;  
    public Statement stmt;  
    public PreparedStatement psmt;  
    public ResultSet rs;  
  
    // 기본 생성자  
    public JDBCConnect() { ②  
        try {  
            // JDBC 드라이버 로드  
            Class.forName("oracle.jdbc.OracleDriver"); ③  
  
            // DB에 연결  
            String url = "jdbc:oracle:thin:@localhost:1521:xe"; ④  
            String id = "musthave"; ⑤  
            String pwd = "1234"; ⑥  
            con = DriverManager.getConnection(url, id, pwd); ⑦  
  
            System.out.println("DB 연결 성공(기본 생성자)");  
        }  
    }  
}
```

- ① DB연결, 쿼리실행, select 결과 저장을 위한 멤버변수 선언
- ② DB연결을 위한 기본생성자
- ③ 드라이버 로드 및 DB접속
- ④ 커넥션 URL





5.7 커넥션 풀로 성능 개선

■ 동작확인

- JDBCConnect 클래스를 이용해 실제로 DB 연결을 테스트하는 jsp 파일을 생성

예제 5-7] 05JDBC/ConnectionTest.jsp

```
<body>
  <h2>JDBC 테스트 1</h2>
  <%
    JDBCConnect jdbc1 = new JDBCConnect(); ❷
    jdbc1.close(); ❸
  %>
</body>
```

Console x Servers
Tomcat v10.1 Server at localhost [Apache Tomcat] C:\#01Developkits\#jdk-17\#b
DB 연결 성공 (기본 생성자)
JDBC 자원 해제



5.6 JDBC 설정 및 데이터베이스 연결

■ 연결 설정 개선

- 앞에서 작성한 클래스는 실무에서 사용하기 적합하지 않음
- 서버 정보가 변경되는 경우 클래스를 수정한 후 다시 컴파일 해야 하기 때문
- 배포서술자인 web.xml에 접속 정보를 입력한 후 application 내장객체를 통해 읽어서 구현

예제 5-8] WEB-INF/web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi=... 생략 ...>
  ... 생략 ...
```

```
<context-param>
```

```
<param-name>OracleDriver</param-name> ❶ 드라이버 이름
```

```
<param-value>oracle.jdbc.OracleDriver</param-value>
```

```
</context-param>
```

[Note] 접속URL, 아이디, 패스워드는 동일한 패턴을 작성하면 됨

■ 연결 설정 개선

예제 5-9] Java Resources/common/JDBConnect.java

... 생략 ...

// 두 번째 생성자

```
public JDBConnect(String driver, String url, String id, String pwd) { ❶
```

```
    try {
```

```
        // JDBC 드라이버 로드
```

```
        Class.forName(driver); ❷
```

```
        // DB에 연결
```

```
        con = DriverManager.getConnection(url, id, pwd); ❸
```

```
        System.out.println("DB 연결 성공(인수 생성자 1)");
```

```
    }
```

DB접속에 필요한 모든 정보를
매개변수를 통해 받을 수 있도록
정의



5.6 JDBC 설정 및 데이터베이스 연결

■ 연결 설정 개선

예제 5-10] 05JDBC/ConnectionTest.jsp

... 생략 ...

```
<h2>JDBC 테스트 2</h2>
```

```
<%
```

```
String driver = application.getInitParameter("OracleDriver");
```

```
String url = application.getInitParameter("OracleURL");
```

```
String id = application.getInitParameter("OracleId");
```

```
String pwd = application.getInitParameter("OraclePwd");
```

```
JDBCConnect jdbc2 = new JDBCConnect(driver, url, id, pwd);
```

```
jdbc2.close();
```

```
%>
```

web.xml의 컨텍스트 초기화 파라미터를 읽어서 생성자의 인수로 전달

Console × Servers
Tomcat v10.1 Server at localhost [Apache Tomcat] C:\W01\Developkits\jdk-17\bin\W
DB 연결 성공(기본 생성자)
JDBC 자원 해제
DB 연결 성공(인수 생성자 1)
JDBC 자원 해제



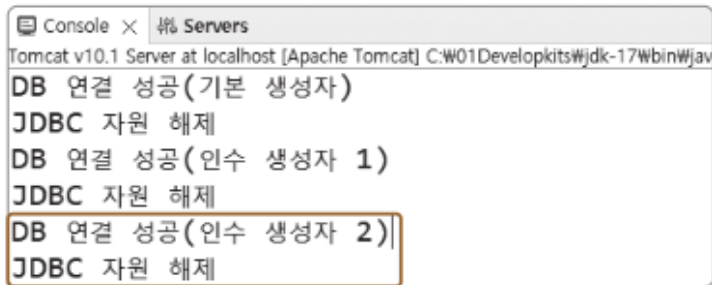
5.6 JDBC 설정 및 데이터베이스 연결

■ 연결 설정 개선2

- 컨텍스트 초기화 파라미터를 JSP에서 가져오면 동일한 코드가 반복되게 됨
- 생성자에서 직접 가져올 수 있도록 정의하면 반복을 제거할 수 있음

```
public class JDBCConnect {  
    ... 생략 ...  
  
    // 세 번째 생성자  
    public JDBCConnect(ServletContext application) { ②  
        try {  
            // JDBC 드라이버 로드  
            String driver = application.getInitParameter("OracleDriver");  
            Class.forName(driver);  
  
            // DB에 연결  
            String url = application.getInitParameter("OracleURL");  
            String id = application.getInitParameter("OracleId");  
            String pwd = application.getInitParameter("OraclePwd");  
            con = DriverManager.getConnection(url, id, pwd);  
  
            System.out.println("DB 연결 성공(인수 생성자 2)");  
        }  
    }  
}
```

② 매개변수로 application 내장객체를 받으면 메서드에서 web.xml에 접근 가능





5.6 JDBC 설정 및 데이터베이스 연결

■ 연결 설정 개선2

- 컨텍스트 초기화 파라미터를 JSP에서 가져오면 동일한 코드가 반복되게 됨
- 생성자에서 직접 가져올 수 있도록 정의하면 반복을 제거할 수 있음

```
public class JDBCConnect {  
    ... 생략 ...  
  
    // 세 번째 생성자  
    public JDBCConnect(ServletContext application) { ②  
        try {  
            // JDBC 드라이버 로드  
            String driver = application.getInitParameter("OracleDriver");  
            Class.forName(driver);  
  
            // DB에 연결  
            String url = application.getInitParameter("OracleURL");  
            String id = application.getInitParameter("OracleId");  
            String pwd = application.getInitParameter("OraclePwd");  
            con = DriverManager.getConnection(url, id, pwd);  
  
            System.out.println("DB 연결 성공(인수 생성자 2)");  
        }  
    }  
}
```

② 매개변수로 application 내장객체를 받으면 메서드에서 web.xml에 접근 가능

```
Console x Servers  
Tomcat v10.1 Server at localhost [Apache Tomcat] C:\W01Developkits\jdk-17\bin\jav  
DB 연결 성공 (기본 생성자)  
JDBC 자원 해제  
DB 연결 성공 (인수 생성자 1)  
JDBC 자원 해제  
DB 연결 성공 (인수 생성자 2)|  
JDBC 자원 해제
```

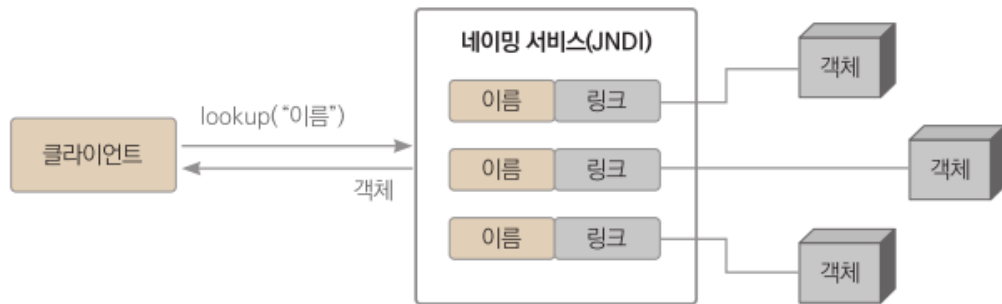
■ 커넥션 풀이란..??

- 웹의 특성상 빈번한 DB와의 연결과 해제는 시스템 성능에 큰 영향을 미침
- 이를 개선하기 위한 방법으로 “커넥션 풀”이 사용됨
- 커넥션 풀(Connection pool)이란 Connection 객체를 미리 생성해 풀(pool)에 넣어놓고, 요청이 있을 때 이미 생성된 Connection 객체를 가져다 사용하는 기법
- 워터파크의 유수풀과 비슷한 원리

■ JNDI(Java Naming and Directory Interface)

- 자바 소프트웨어에서 객체나 데이터를 전체 경로를 몰라도 ‘이름’만으로 찾아 쓸 수 있는 디렉터리 서비스
- 이름과 실제 객체와의 연결은 외부의 설정 파일에서 관리
- 따라서 세부 설정을 바꿀 때도 소스 코드를 수정하고 다시 컴파일할 필요가 없음

■ JNDI(Java Naming and Directory Interface)



● JNDI 사용 절차

- WAS(톰캣)가 시작할 때 server.xml과 context.xml에 설정한 대로 커넥션 풀을 생성
- JSP 코드에서 JNDI 서버(WAS가 제공)로부터 데이터소스 객체를 얻어옴
- 데이터소스로부터 커넥션 객체를 가져옴
- DB 작업을 수행
- 모든 작업이 끝나면 커넥션 객체를 풀로 반환

☐ 커넥션 풀 설정

예제 5-13] (톰캣 홈 디렉터리)/conf/server.xml

```
<GlobalNamingResources>
  ... 기존 내용 유지 ...
  <Resource auth="Container"
    driverClassName="oracle.jdbc.OracleDriver"
    type="javax.sql.DataSource"
    initialSize="0"
    minIdle="5"
    maxTotal="20"
    maxIdle="20"
    maxWaitMillis="5000"
    url="jdbc:oracle:thin:@localhost:1521:xe"
    name="dbcp_myoracle"
    username="musthave"
    password="1234" />
</GlobalNamingResources>
```

- driverClassName : JDBC 드라이버
- type : 데이터소스로 사용할 클래스명
- initialSize : 초기화 과정에서 생성할 커넥션 갯수
- minIdle : 커넥션의 최소값
- maxTotal : 동시에 사용할 커넥션 갯수
- maxIdle : 커넥션의 최대값
- maxWaitMillis : 대기시간
- url : 오라클 연결을 위한 URL
- name : 풀의 이름
- username : 계정 아이디
- password : 계정 패스워드



5.7 커넥션 풀로 성능 개선

■ 커넥션 풀 설정

예제 5-14] (톰캣 홈 디렉터리)/conf/context.xml

```
<Context>
  ... 기존 내용 유지 ...
  <ResourceLink global="dbcp_myoracle" name="dbcp_myoracle"
    type="javax.sql.DataSource"/>
</Context>
```

server.xml에 설정한 커넥션 풀을 전역적으로 사용하기 위한 설정

JSP 코드

```
:
context.lookup("자원 이름")
:
```

context.xml

```
<Context>
  <ResourceLink
    global="전역 자원 이름"
    name="자원 이름"
    type=".."/>
</Context>
```

server.xml

```
<GlobalNamingResources>
  <Resource auth="Container"
    :
    name="전역 자원 이름"
    :
  />
</GlobalNamingResources>
```




5.7 커넥션 풀로 성능 개선

■ 커넥션 풀 동작 검증

예제 5-15] Java Resources/common/DBConnPool.java

```
// 기본 생성자
public DBConnPool() {
    try {
        // 커넥션 풀(DataSource) 얻기
        Context initCtx = new InitialContext(); ❶
        Context ctx = (Context)initCtx.lookup("java:comp/env"); ❷
        DataSource source = (DataSource)ctx.lookup("dbcp_myoracle"); ❸

        // 커넥션 풀을 통해 연결 얻기
        con = source.getConnection(); ❹

        System.out.println("DB 커넥션 풀 연결 성공");
    }
    catch (Exception e) {
        System.out.println("DB 커넥션 풀 연결 실패");
        e.printStackTrace();
    }
}
```

- ❶ JNDI를 사용하기 위한 객체생성
- ❷ 웹 애플리케이션 즉 Tomcat 의 루트 디렉토리를 얻어옴
- ❸ Tomcat에 생성된 커넥션 풀인 "dbcp_myoracle" 자원을 얻어옴
- ❹ 이를 통해 DB연결



5.7 커넥션 풀로 성능 개선

■ 커넥션 풀 동작 검증

예제 5-16] 05JDBC/ConnectionTest.jsp

```
<h2>커넥션 풀 테스트</h2>
```

```
<%
```

```
DBConnPool pool = new DBConnPool();
```

```
pool.close();
```

```
%>
```

Servers Console ×

Tomcat v10.1 Server at localhost [Apache Tomcat] C:\W01\Developkits\jdk-17\bin\javaw.exe

DB 연결 성공(기본 생성자)

JDBC 자원 해제

DB 연결 성공(인수 생성자 1)

JDBC 자원 해제

DB 연결 성공(인수 생성자 2)

JDBC 자원 해제

DB 커넥션 풀 연결 성공

DB 커넥션 풀 자원 반납

■ SQL문 작성 및 실행을 위한 인터페이스

- Statement : 인파라미터가 없는 정적 쿼리를 처리할 때 사용
- PreparedStatement : 인파라미터가 있는 동적 쿼리를 처리할 때 사용
- CallableStatement : 프로시저(procedure)나 함수(function)를 호출할 때 사용

■ 동적 쿼리문으로 회원 추가

예제 5-17] 05JDBC/ExeUpdate.jsp

```
<%  
// DB에 연결 ①  
JDBCConnect jdbc = new JDBCConnect();  
  
// 테스트용 입력값 준비 ②  
String id = "test1";  
String pass = "1111";  
String name = "테스트1회원";
```

■ 동적 쿼리문으로 회원 추가(계속)

// 쿼리문 생성

String sql = "INSERT INTO member VALUES (?, ?, ?, sysdate)"; ③

PreparedStatement pstmt = jdbc.con.prepareStatement(sql); ④

pstmt.setString(1, id);

pstmt.setString(2, pass);

pstmt.setString(3, name);

⑤

// 쿼리 수행

int inResult = pstmt.executeUpdate(); ⑥

out.println(inResult + "행이 입력되었습니다.");

// 연결 닫기

jdbc.close(); ⑦

%>

③ 인파라미터가 있는 동적

insert 쿼리문 작성

④ 쿼리문 실행을 위한 객체 생성

⑤ 인파라미터 설정

⑥ 쿼리문 실행 및 결과 반환

회원 추가 테스트(executeUpdate() 사용)

1행이 입력되었습니다.

The screenshot shows a Java IDE with a database connection named 'Musthave계정' selected. The 'MEMBER' table is displayed with the following data:

ID	PASS	NAME	REGDATE
1 musthave	1234	테스트하브	22/12/21
2 test1	1111	테스트1회원	22/12/21

■ 정적 쿼리문으로 회원 조회

```
<%  
// DB에 연결  
JDBCConnect jdbc = new JDBCConnect();  
  
// 쿼리문 생성  
String sql = "SELECT id, pass, name, regdate FROM member"; ❶  
Statement stmt = jdbc.con.createStatement(); ❷  
  
// 쿼리 수행  
ResultSet rs = stmt.executeQuery(sql); ❸  
// 결과 확인(웹 페이지에 출력)  
while (rs.next()) { ❹  
    String id = rs.getString(1);  
    String pw = rs.getString(2);  
    String name = rs.getString("name");  
    java.sql.Date regdate = rs.getDate("regdate");  
  
    out.println(String.format("%s %s %s %s", id, pw, name, regdate) +  
        "<br/>"); ❺  
}
```

- ❶ 인파라미터가 없는 정적 select 쿼리문
- ❷ 쿼리문 실행을 위한 객체 생성
- ❸ 쿼리문 실행
- ❹ select를 통해 인출된 레코드의 갯수만큼 반복해서 출력

회원 목록 조회 테스트(executeQuery() 사용)

```
musthave 1234 머스트해브 2022-12-21  
test1 1111 테스트1회원 2022-12-21
```

■ JDBC 프로그래밍 순서

1. JDBC 드라이버 로드
2. 데이터베이스 연결
3. 쿼리문 작성
4. 쿼리문(Statement 계열) 객체 생성
5. 쿼리 실행
6. 실행 결과 처리
7. 연결 해제

■ ResultSet에서 결과값 불러오기

최초 커서 위치

컬럼 인덱스

	1	2	3	4
	id	pass	name	regdate
next()	MUST	****	성낙현	2021.09.15
next()	HAVE	****	최현우	2021.09.15
:	JSP	****	이복연	2021.09.15
	RABBIT	****	골든래빗	2021.09.15

- select 한 결과는 ResultSet으로 반환
- 최초 커서는 첫번째 행 윗부분에 위치
- next() 메서드로 다음 행으로 이동
- 이때 다음 행이 있으면 true, 없으면 false
- 각 컬럼의 값은 getX() 메서드로 읽음
 - a. getInt() : 정수형으로 추출
 - b. getDate() : 날짜형으로 추출
 - c. getString() : 문자형으로 추출

■ 핵심요약

- 쿼리문 작성 방법
 - 정적 쿼리문 : 내용이 고정되어 값을 추가할 수 없는 쿼리문
 - 동적 쿼리문 : 인파라미터를 이용해서 값을 변경할 수 있는 쿼리문
- 쿼리문 실행을 위한 인터페이스
 - Statement : 정적 쿼리문을 실행
 - PreparedStatement : 동적 쿼리문을 실행
- 쿼리문 실행을 위한 메서드
 - executeQuery()
 - 테이블에 저장된 레코드를 변경하지 않는 SELECT문 실행
 - 반환타입은 ResultSet(조건절에 따른 레코드 인출)
 - executeUpdate()
 - 레코드를 변경하는 INSERT, UPDATE, DELETE문 실행
 - 반환타입은 int(변화된 행의 갯수를 반환)

