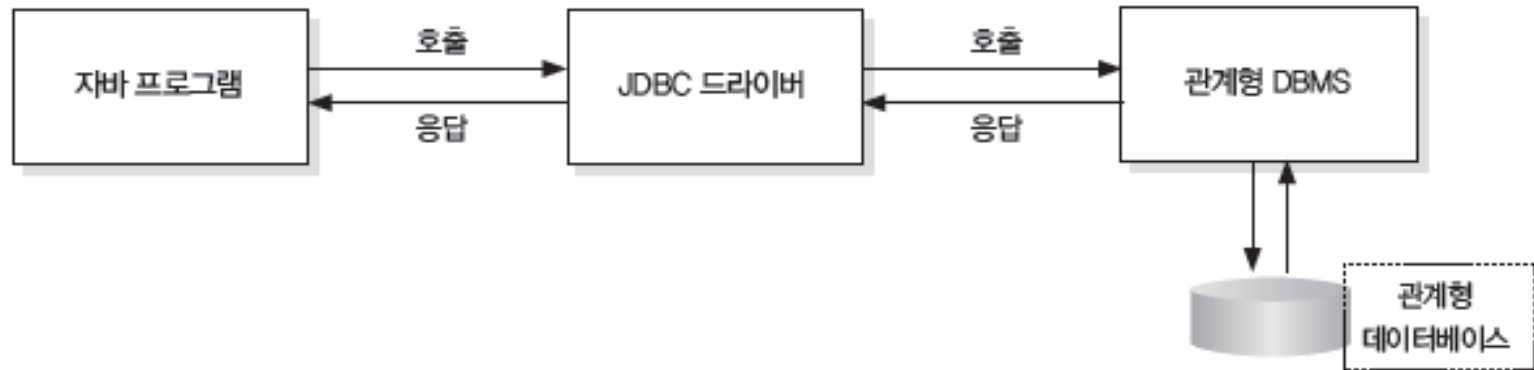


데이터베이스 사용하기

JDBC

- 데이터베이스(database)는 파일과 마찬가지로 보조기억장치에 데이터를 저장하는 수단입니다.
- 자바 프로그램에서 데이터베이스를 사용하려면 JDBC 드라이버가 필요합니다.



❖ 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`

❖ **DBMS URL: DBMS와의 연결을 위한 식별 값**

❖ **JDBC 드라이버에 따라 형식 다름**

❖ **일반적인 구성**

- **jdbc:[DBMS]:[데이터베이스식별자]**

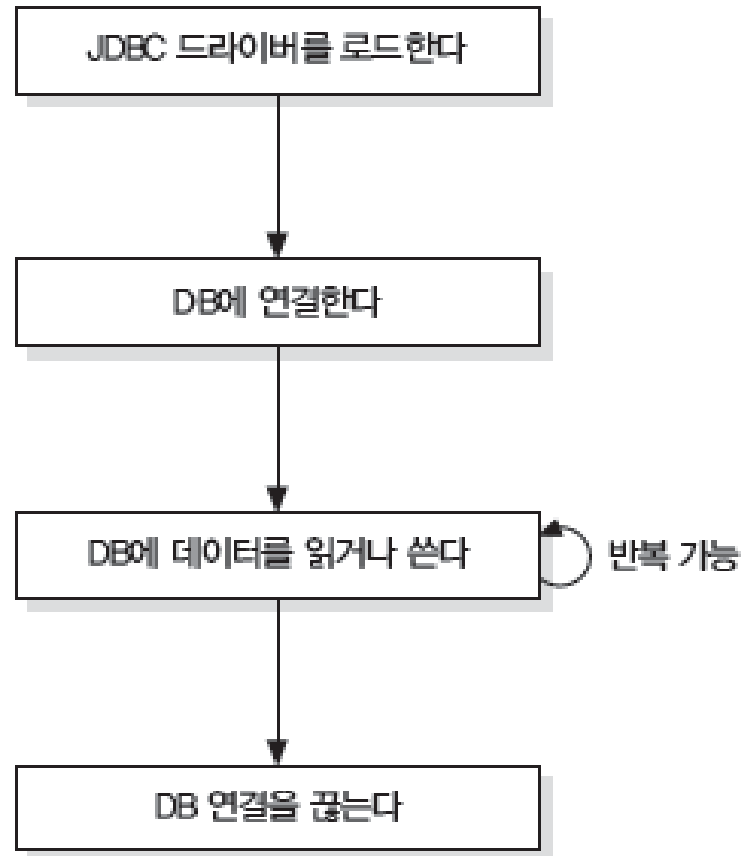
❖ **주요 DBMS의 JDBC URL 구성**

- **MySQL : jdbc:mysql://HOST[:PORT]/DBNAME**

- **Oracle: jdbc:oracle:thin:@HOST:PORT:SID**

- **MSSQL : jdbc:sqlserver://HOST[:PORT];databaseName=DB**

데이터베이스 사용



❖ JDBC 드라이버를 로드하고, 데이터베이스로 연결하고, 연결을 끊는 방법

- 자바 프로그램에서 관계형 데이터베이스로 연결을 하기 위해서는 먼저 다음과 같은 방법으로 JDBC 드라이버를 로드해야 한다.

Class.forName('oracle.jdbc.driver.OracleDriver');

Mysql : com.mysql.jdbc.Driver

- 데이터베이스로 연결을 하기 위해서는 먼저 프로토콜, 서브프로토콜, 서브네임으로 이루어진 데이터베이스 URL을 알아두어야 한다.
- Mysql : jdbc:mysql://localhost:3306/db명
- “~~jdbc~~:~~oracle~~:thin:@127.0.0.1:1521:orcl”;

프로토콜
(protocol)

서브프로토콜
(subprotocol)

서브네임(subname)
IP 주소: 포트번호/DB이름

- 서브네임의 작성 방법은 DBMS의 종류마다 다르므로 다른 DBMS를 사용할 때는 관련 매뉴얼을 찾아보아야 한다.

❖ JDBC 드라이버를 로드하고, 데이터베이스로 연결하고, 연결을 끊는 방법

- 데이터베이스로 연결을 맺기 위해서는 `java.sql.DriverManager` 클래스의 `getConnection` 메서드를 호출해야 한다.

- `con = DriverManager.getConnection(url, "scott", "tiger");`

데이터베이스 URL

사용자 ID

패스워드

- 이 메서드는 데이터베이스로의 연결에 성공하면 `java.sql.Connection` 인터페이스 타입의 객체를 만들어서 리턴한다.
- 데이터베이스로의 연결을 끊기 위해서는 `Connection` 객체에 대해 `close` 메서드를 호출하면 된다.

`conn.close()`

데이터베이스로의
연결을 끊는 메서드

❖ JDBC 드라이버를 로드하고, 데이터베이스로 연결하고, 연결을 끊는 방법

JDBC 드라이버의 로드, 데이터베이스로 연결하고 연결 끊기 DBTest.jsp

```
<% @page contentType= "text/html; charset=euc-kr" %>
<%@page import= "java.sql.*" %>
<HTML>
  <HEAD><TITLE>데이터베이스로 연결하기</TITLE></HEAD>
  <BODY>
    <H3>데이터베이스 연결 테스트</H3>
    <%
      Class.forName( "oracle.jdbc.driver.OracleDriver" );
      Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@127.0.0.1:1521:orcl",
        "scott ", "tiger ");
      if (conn != null) {
        out.println( "webdb 데이터베이스로 연결했습니다.<BR> " );
        conn.close();
        out.println( "webdb 데이터베이스로의 연결을 끊었습니다.<BR> " );
      }
      else {
        out.println( "webdb 데이터베이스로 연결할 수 없습니다.<BR> " );
      }
    %>
  </BODY>
</HTML>
```


❖ 데이터베이스의 데이터를 읽어오는 방법

- 데이터베이스에 있는 데이터를 읽어오려면 우선 Connection 객체에 대해 createStatement 메서드를 호출해서 java.sql.Statement 타입 객체를 구해야 한다.

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

getConnection 메서드가
리턴한 Connection 객체

Statement 객체를
만들어서 리턴하는 메서드

- Statement 객체에 대해 executeQuery 메서드를 호출하면 데이터베이스에 있는 데이터를 읽어올 수 있다.

```
ResultSet rs = stmt.executeQuery( "select * from goodsinfo where code='10002'; ");
```

select 문을 실행하는 메서드

❖ 데이터베이스의 데이터를 읽어오는 방법

- executeQuery 메서드가 리턴한 ResultSet 객체에 대해 next 메서드를 호출하면 데이터베이스로부터 읽은 데이터를 순서대로 가져올 수 있다.

```
boolean exists = rs.next();
```

데이터베이스로부터 읽은 데이터의
첫 번째/다음 행 위치로 이동하는 메서드

- 이 메서드는 다음 위치에 데이터가 있을 때는 true, 없을 때는 false를 리턴한다.
- next 메서드를 호출한 다음에 ResultSet 객체에 대해 getInt, getString, getFloat 등의 메서드를 호출하면 특정 데이터 항목 값을 가져올 수 있다.

```
String code = rs.getString( "code " );
```

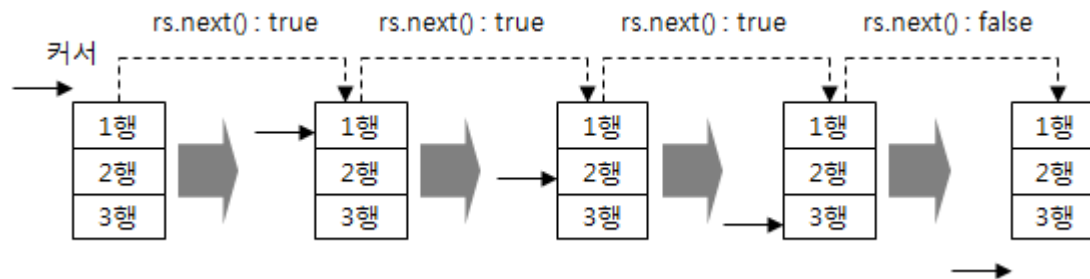
문자 데이터를 가져오는 메서드

```
int price = rs.getInt( "price " );
```

정수 데이터를 가져오는 메서드

데이터베이스 사용

❖ next() 메서드로 데이터 조회 여부 확인



데이터베이스 사용

❖ 1개 행 처리

```
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 않음
}
```

❖ 1개 이상 행 처리

```
rs = stmt.executeQuery(...);
while(rs.next()) {
    String name = rs.getString("NAME");
    ...
}
```

```
rs = stmt.executeQuery(...);
if (rs.next()) {
    do {
        String name = rs.getString("NAME");
        ...
    } while( rs.next() );
}
```

❖ 데이터베이스의 데이터를 읽어오는 방법

- 필요한 데이터를 모두 가져온 다음에는 ResultSet 객체가 더 이상 필요치 않기 때문에 close 메서드를 호출해야 한다.

```
rs.close();
```



ResultSet을 닫는 메서드

- Statement 객체도 모두 사용하고 난 다음에는 close 메서드를 호출해서 닫아야 한다.

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



Statement를 닫는 메서드

❖ 데이터베이스의 데이터 1건을 읽어오는 방법

상품 정보 테이블을 읽는 JSP 페이지 deptInput.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><head><meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>Insert title here</title>
<script type="text/javascript">
function chk() {
    if (!frm.deptno.value) {
        alert("부서코드를 입력하십시오");
        frm.deptno.focus();
        return false;
    }
    return true;
}
</script>
</head>
<body>
<h2>보고싶은 부서코드를 입력하십시오</h2>
<form action="oracleTest3.jsp" name="frm" onsubmit="return chk()">
    부서코드 : <input type="text" name="deptno"><p>
    <input type="submit" value="확인">
</form>
</body>
</html>
```

oraTest3.jsp

```
<% @ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR" import="java.sql.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><head><meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>Insert title here</title></head><body>
<%
    String dno = request.getParameter("deptno");
    String sql = String.format("select * from dept where deptno=%s",dno);
    String driver = "oracle.jdbc.driver.OracleDriver";
    String url = "jdbc:oracle:thin:@localhost:1521:xe";
    try { Class.forName(driver);
        Connection conn = DriverManager.getConnection(url, "scott", "tiger");
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(sql);
        if (rs.next()) {
            request.setAttribute("deptno", rs.getInt(1));
            request.setAttribute("dname", rs.getString(2));
            request.setAttribute("loc", rs.getString(3));
        } else out.println("그런 부서 없는데 ㅠㅠ");
        rs.close(); stmt.close(); conn.close();
        RequestDispatcher rd = request.getRequestDispatcher("oracleTest3Result.jsp");
        rd.forward(request, response);
    } catch (Exception e) {
        out.println("DB 연결에 실패했습니다" + e.getMessage());
    }
%>
</body>
</html>
```

❖ 데이터베이스의 데이터 1건을 읽어오는 방법

oraTest3resul.jsp

```
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><head><meta http-equiv="Content-Type" content="text/html;
charset=EUC-KR">
<title>Insert title here</title>
</head>
<body>
<H2>부서정보</H2>
부서코드 : ${dno}<p>
부서명 : ${dname}<p>
전화번호 : ${phone}<p>
근무지 : ${position}
</body>
</body>
</html>
```


oraTest5.jsp

```
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR" import="java.sql.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><head><meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>Insert title here</title></head><body><h2>직원 정보</h2><table border=1><tr><th>사
번</th><th>이름</th><th>급여</th><th>업무</th></tr>
<%
    String sql = "select empno, ename, sal, job from emp";
    String driver = "oracle.jdbc.driver.OracleDriver";
    String url = "jdbc:oracle:thin:@localhost:1521:xe";
    try { Class.forName(driver);
        Connection conn = DriverManager.getConnection(url, "scott", "tiger");
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(sql);
        while (rs.next()) { out.println("<tr>");
            int empno = rs.getInt(1); String ename = rs.getString(2);
            int sal = rs.getInt(3); String job = rs.getString(4);
            out.println("<td>" + empno + "</td><td>" + ename + "</td><td>" +
sal + "</td><td>" + job + "</td>");
            out.println("</tr>");
        }
        rs.close(); stmt.close(); conn.close();
    } catch (Exception e) { out.println("DB 연결에 실패했습니다" + e.getMessage()); }
%></table></body></html>
```

oraTest6.jsp

```
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR" import="java.sql.*,java.util.ArrayList,ch10.Emp"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><head><meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>Insert title here</title></head><body>
<%   String sql = "select empno, ename, sal, job from emp";
    String driver = "oracle.jdbc.driver.OracleDriver";
    String url = "jdbc:oracle:thin:@localhost:1521:xe";
    try { Class.forName(driver);
        ArrayList<Emp> al = new ArrayList<Emp>();
        Connection conn = DriverManager.getConnection(url, "scott", "tiger");
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(sql);
        while (rs.next()) {   Emp emp = new Emp();           emp.setEmpno(rs.getInt(1));
                               emp.setEname(rs.getString(2)); emp.setSal(rs.getInt(3));
                               emp.setJob(rs.getString(4));   al.add(emp);
        }
        rs.close(); stmt.close(); conn.close();
        request.setAttribute("al", al);
        RequestDispatcher rd = request.getRequestDispatcher("oraTest6Result.jsp");
        rd.forward(request, response);
    }catch(Exception e) { out.println("DB 연결에 실패했습니다" + e.getMessage()); }
%>
</body></html>
```

oraTest6Result.jsp

```
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR" %>
<%@page import="java.util.ArrayList,ch10.Personal"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><head><meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>Insert title here</title></head>
<body>  <table border="1" width=500 align=center>
        <tr><td colspan="4" align=center><h2>직원 정보</h2></td></tr>
        <tr bgcolor=pink>
        <th>사번</th><th>이름</th><th>급여</th><th>업무</th></tr>
<%
    ArrayList<Personal> al = (ArrayList<Personal>)request.getAttribute("al");
    for (int i = 0 ; i < al.size(); i++) {
        Personal p = al.get(i);
        if (i%2==0) out.println("<tr align=center bgcolor=green>");
        else out.println("<tr align=center bgcolor=yellow>");

        out.println("<td>" + p.getPno() + "</td>");
        out.println("<td>" + p.getPname() + "</td>");
        out.println("<td>" + p.getPay() + "</td>");
        out.println("<td>" + p.getJob() + "</td>");

    }
%>
</table></body></html>
```

로그인

아이디 :

비밀번호 :

로그인

fieldset요소가 폼 요소(컨트롤)들을 구조적으로 감싼 것을 볼 수 있습니다.

form요소안에 그룹화할 입력요소들이 없으면 **fieldset**을 생략 해도 됩니다.
다만, **fieldset**요소는 form요소안에서만 사용 할 수 있다는 것을 알아두세요.

2. fieldset

fieldset요소는 내용이 같은 입력요소들을 그룹화 하는 요소로 form요소 안에 정의합니다.

fieldset요소에도 table의 caption처럼 제목을 부여 할 수 있는데 바로 legend 요소입니다.

실제 예를 보겠습니다.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ko" xml:lang="ko">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>폼요소</title>
</head>
<body>
<form action="#" method="post">
  <fieldset>
    <legend>로그인</legend>
    <p>아이디 : <input type="text" /></p>
    <p>비밀번호 : <input type="password" /></p>
    <p><input type="button" value="로그" /></p>
  </fieldset>
</form>
</body>
```

데이터베이스 사용

- ❖ 멀티 select(unique 속성이 아닌 컬럼을 가지고 조회하는 경우)
- ❖ 반복문을 이용해서 데이터를 읽어내야 하며 데이터를 저장하는 경우 ArrayList나 LinkedList와 같은 데이터 구조를 이용해야 합니다.
- ❖ 위와 같은 경우 데이터를 표현하는 별도의 클래스를 만들어야 하는 경우가 많습니다.

저자를 입력하세요

세익스피어

조회

상품정보

코드	제목	저자	가격
10004	로미오와 줄리엣	세익스피어	250000
10005	리어왕	세익스피어	280000

❖ src 폴더에 dto 패키지를 생성하고 Dept 클래스 생성

```
package dto;
```

```
public class Dept {
```

```
    private int deptno;        private String dname;    private String loc;
```

```
    public int getDeptno() { return deptno; }
```

```
    public void setDeptno(int deptno) {
```

```
        this.deptno = deptno;
```

```
    }
```

```
    public String getDname() { return dname; }
```

```
    public void setDname(String dname) {
```

```
        this.dname = dname;
```

```
    }
```

```
    public String getLoc() { return loc; }
```

```
    public void setLoc(String loc) {
```

```
        this.loc = loc;
```

```
    }
```

```
}
```

❖ 처리파일(multiSelect – oratest7.jsp)

```
<%@page import="java.util.Vector"%>
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR" import="java.sql.*,ch10.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><head><meta http-equiv="Content-Type" content="text/html;
charset=EUC-KR">
<title>Insert title here</title></head>
<body>
<%
    String sql = "select * from dept";
    String driver = "oracle.jdbc.driver.OracleDriver";
    String url = "jdbc:oracle:thin:@localhost:1521:xe";
    try { Class.forName(driver);
        Connection conn =
            DriverManager.getConnection(url, "scott", "tiger");
        Vector<Dept> vc = new Vector<Dept>();
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(sql);
        while (rs.next()) {
```




```
        Dept dp = new Dept();
        dp.setDeptno(rs.getInt(1));
        dp.setDname(rs.getString(2));
        dp.setLoc(rs.getString(3));
        vc.add(dp);
    }
    rs.close(); stmt.close(); conn.close();
    request.setAttribute("al", vc);
    RequestDispatcher rd =
        request.getRequestDispatcher("ora7Result.jsp");
    rd.forward(request, response);
} catch (Exception e) {
    out.println("DB 연결에 실패했습니다" + e.getMessage());
}

%>
</body>
</html>
```

ora7Result.jsp

```
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR" %>
<%@page import="java.util.Vector,ch10.Dept"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><head><meta http-equiv="Content-Type" content="text/html;
charset=EUC-KR">
<title>Insert title here</title></head>
<body>
    <table border="1" width=500 align="center">
    <tr align="center"><td colspan="3"><h2>부서 정보</h2></td></tr>
    <tr bgcolor=pink>
    <th>부서코드</th><th>부서명</th><th>근무지</th></tr>

<%
    Vector<Dept> al = (Vector<Dept>)request.getAttribute("al");
    for (int i = 0 ; i < al.size(); i++) {
        Dept dp = al.get(i);
        if (i%2==0) out.println("<tr align=center bgcolor=green>");
        else out.println("<tr align=center bgcolor=yellow>");
```



```
out.println("<td>" + dp.getDeptno() + "</td>");  
out.println("<td>" + dp.getDname() + "</td>");  
out.println("<td>" + dp.getLoc() + "</td>");
```

```
}
```

```
%>
```

```
</table>
```

```
</body>
```

```
</html>
```



❖ 데이터베이스에 데이터를 입력하는 방법

- 데이터베이스에 데이터를 입력하려면 데이터를 읽어올 때와 마찬가지로 우선 Statement 객체를 구해야 한다.

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

↑
Statement 객체를 만드는 메서드

- Statement 객체에 대해 executeUpdate라는 메서드를 호출하면 데이터베이스에 새로운 데이터를 추가할 수 있다.

insert 문을 실행하는 메서드

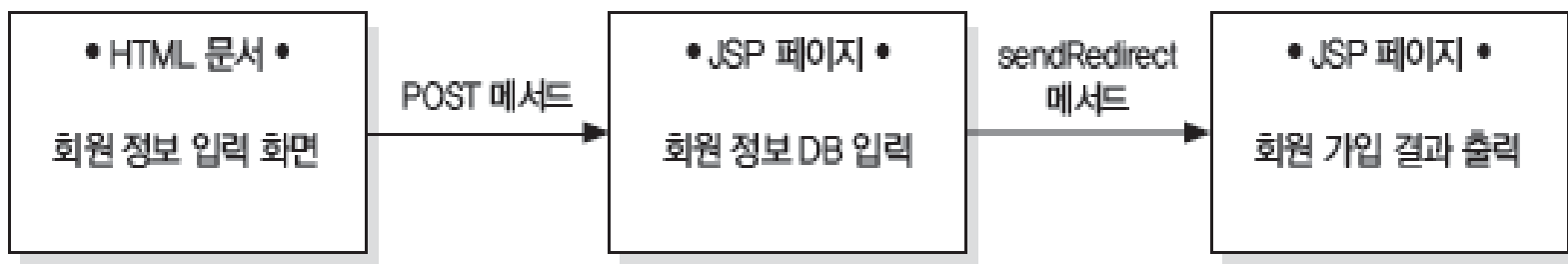
```
int rowNum = stmt.executeUpdate(
```

```
    "insert goodsinfo (code, title, writer, price) values('10001', 'Java', '누꼬' , 27000);");
```

- executeUpdate 메서드를 이용해서 데이터를 입력한 다음에는 Statement 객체에 대해 close 메서드를 호출해야 한다.

❖ 데이터베이스에 데이터를 입력하는 방법

- 이 애플리케이션은 다음과 같은 세 개의 모듈로 구현한다.



[그림 12-30] 회원 가입 애플리케이션의 구성도

- HTML 문서와 JSP 페이지의 URL은 다음과 같이 정하기로 한다.

<http://localhost:8181/dataInput.html>

회원 정보 입력 화면
HTML 문서의 URL

<http://localhost:8181/oraDeptInput2.jsp>

회원 정보 DB 입력
JSP 페이지의 URL

❖ 데이터베이스에 데이터를 입력하는 방법

회원 정보를 입력받는 HTML 문서 dataInput.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><head><meta http-equiv="Content-Type" content="text/html; charset=EUC-
KR">
<title>Insert title here</title>
<script type="text/javascript">
    function chk() {
        if (!frm.deptno.value) {      alert("부서코드를 입력하십시오");
            frm.deptno.focus();      return false;
        }
        if (!frm.dname.value) {      alert("부서명을 입력하십시오");
            frm.dname.focus();      return false;
        }
        if (!frm.loc.value) {      alert("근무지를 입력하십시오");
            frm.loc.focus();      return false;
        }
        return true;
    }
</script></head>
```

<body>

<h2>부서 정보를 입력하세요</h2>

<form action="oraDeptInput2.jsp" name="frm" onsubmit="chk()">

부서코드 : <input type="text" name="deptno"><p>

부서명 : <input type="text" name="dname"><p>

근무지 : <input type="text" name="loc"><p>

<input type="submit" value="입력완료"><p>

<input type="reset" value="입력취소"><p>

</form>

</body>

</html>

❖ 데이터베이스에 데이터를 입력하는 방법

회원 정보를 데이터베이스에 입력하는 JSP 페이지 oraDeptInput.jsp

```
<% @ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR" import="java.sql.*"%><!DOCTYPE html PUBLIC "-//W3C//DTD HTML
4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html><head><meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>Insert title here</title></head><body>
<%
    String deptno = request.getParameter("deptno");
    String dname = request.getParameter("dname");
    String loc = request.getParameter("loc");
    String sql = String.format("insert into dept values(%s,'%s','%s')",
        deptno, dname, loc);
    String driver = "oracle.jdbc.driver.OracleDriver";
    String url = "jdbc:oracle:thin:@localhost:1521:xe";
    Connection conn = null;      Statement stmt = null;
    try { Class.forName(driver);
        conn=DriverManager.getConnection(url,"scott","tiger");
        stmt = conn.createStatement();
        int result = stmt.executeUpdate(sql);
        out.println("입력 성공 ㅋㅋㅋㅋ");
    } catch (Exception e) { out.println("입력실패 ㅠㅠ 메시지 : "+e.getMessage());
    } finally { stmt.close(); conn.close();
    }
%>
</body></html>
```


oraDeptInput2.jsp

```
<% @ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR" import="java.sql.*"%><!DOCTYPE html PUBLIC "-//W3C//DTD
HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html><head><meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>Insert title here</title></head><body>
<%
    int deptno = Integer.parseInt(request.getParameter("deptno"));
    String dname = request.getParameter("dname");
    String loc = request.getParameter("loc");
    String sql = "insert into dept values(?,?,?)";
    String driver = "oracle.jdbc.driver.OracleDriver";
    String url = "jdbc:oracle:thin:@localhost:1521:xe";
    Connection conn = null;          PreparedStatement pstmt = null;
    try { Class.forName(driver);
        conn=DriverManager.getConnection(url,"scott","tiger");
        pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, deptno);          pstmt.setString(2, dname);
        pstmt.setString(3, loc);
        int result = pstmt.executeUpdate();
        out.println("입력 성공 ㅋㅋㅋㅋ");
    } catch (Exception e) {
        out.println("입력실패 ㅠㅠ 메시지 :"+e.getMessage());
    } finally { pstmt.close(); conn.close(); }
%>
</body></html>
```

❖ 데이터베이스의 데이터를 수정하고 삭제하는 방법

- 데이터베이스에 있는 데이터를 수정할 때는 executeUpdate 메서드에 update 문을 파라미터로 넘겨줘야 한다.

```
int rowNum = stmt.executeUpdate( "update userinfo set password := 'dalek' where id = 'rose ' ");
```

↑
이 메서드를 이용해서 update 문을 실행할 수 있습니다

- 데이터베이스에 있는 데이터 삭제할 때는 executeUpdate 메서드에 delete 문을 파라미터로 넘겨줘야 한다.

```
int rowNum = stmt.executeUpdate( "delete from userinfo where id = 'rose ' ");
```

↑
이 메서드를 이용해서 delete 문을 실행할 수도 있습니다.

❖ 데이터베이스의 데이터를 수정하고 삭제하는 방법

상품코드를 입력받는 HTML 문서 dataInput3.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><head><meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>Insert title here</title>
<script type="text/javascript">
    function chk() {
        if (!frm.deptno.value) {            alert("부서코드를 입력하십시오");
            frm.deptno.focus(); return false;
        }
        return true;
    }
</script></head>
<body>
<h2>수정할 부서코드를 입력하세요</h2>
<form action="oraUpdateData.jsp" name="frm" onsubmit="chk()">
    부서코드 : <input type="text" name="deptno"><p>
    <input type="submit" value="입력완료"><p>
    <input type="reset" value="입력취소"><p>
</form>
</body>
</html>
```

❖ 데이터베이스의 데이터를 수정하고 삭제하는 방법

상품 정보 테이블을 읽는 JSP 페이지 *oraUpdateData.jsp*

```
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR" import="java.sql.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html> <head> <meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>Insert title here</title> </head> <body>
<%    String dno = request.getParameter("deptno");
    String sql = String.format("select * from dept where deptno=%s",dno);
    String driver = "oracle.jdbc.driver.OracleDriver";
    String url = "jdbc:oracle:thin:@localhost:1521:xe";
    try { Class.forName(driver);
        Connection conn = DriverManager.getConnection(url, "scott", "tiger");
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(sql);
        if (rs.next()) {            request.setAttribute("deptno", rs.getInt(1));
                                    request.setAttribute("dname", rs.getString(2));
                                    request.setAttribute("loc", rs.getString(3));
        } else out.println("그런 부서 없는데 ㅠㅠ");
        rs.close(); stmt.close(); conn.close();
        RequestDispatcher rd = request.getRequestDispatcher("oraUpdateForm.jsp");
        rd.forward(request, response);
    } catch (Exception e) { out.println("DB 연결에 실패했습니다" + e.getMessage()); }
%>
</body> </html>
```

oraUpdateForm.jsp

```
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR"%> <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html> <head> <meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>Insert title here</title> <script type="text/javascript">
function chk() {
    if (!frm.dname.value) {        alert("부서명을 입력하십시오");
        frm.dname.focus(); return false;    }
    if (!frm.loc.value) { alert("근무지를 입력하십시오");
        frm.loc.focus();    return false;    }
    return true;
}
</script> </head> <body> <h2>수정할 부서내용을 입력하세요</h2>
<form action="oraUpdate.jsp" name="frm" onsubmit="return chk()">
<table border=1 bgcolor=yellow>
    <tr> <td>부서코드 </td> <td> <input type="text"
        name="deptno" value="${deptno}" readonly="readonly"> </td> </tr>
    <tr> <td>부서명 </td> <td> <input type="text" name="dname"
        value="${dname}"> </td> </tr>
    <tr> <td>근무지 </td> <td> <input type="text"
        name="loc" value="${loc}"> </td> </tr>
    <tr> <td> <input type="submit" value="입력완료"> </td> </tr>
</table> </form>
</body> </html>
```

❖ 데이터베이스의 데이터를 수정하고 삭제하는 방법

상품 정보 관리 화면을 제공하는 JSP 페이지 Update.jsp

```
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR" import="java.sql.*" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html> <head> <meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>Insert title here</title> </head> <body>
<%
    String deptno = request.getParameter("deptno");
    String dname = request.getParameter("dname");
    String loc = request.getParameter("loc");
    String sql = String.format("update dept set dname='%s', loc='%s' " +
        " where deptno=%s",dname, loc, deptno);
    String driver = "oracle.jdbc.driver.OracleDriver";
    String url = "jdbc:oracle:thin:@localhost:1521:xe";
    Connection conn = null;      Statement stmt = null;
    try { Class.forName(driver);
        conn=DriverManager.getConnection(url,"scott","tiger");
        stmt = conn.createStatement();
        int result = stmt.executeUpdate(sql);
        out.println("수정 성공 ㅋㅋㅋㅋ");
    } catch (Exception e) { out.println("수정실패 ㅠㅠ 메세지 :"+e.getMessage());
    } finally { stmt.close(); conn.close();
    }
%>
</body> </html>
```

❖ 데이터베이스의 데이터를 수정하고 삭제하는 방법

상품 정보의 수정 결과를 보여주는 JSP 페이지 dataInput3.jsp

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html> <head> <meta http-equiv="Content-Type" content="text/html;
charset=EUC-KR">
<title>Insert title here</title>
<script type="text/javascript">
    function chk() {
        if (!frm.deptno.value) {    alert("부서코드를 입력하십시오");
                                   frm.deptno.focus();                return false;
        }
        return true;
    }
</script> </head> <body>
<h2>삭제할 부서코드를 입력하세요</h2>
<form action="oraDelete.jsp" name="frm" onsubmit="chk()">
    부서코드 : <input type="text" name="deptno"> <p>
    <input type="submit" value="입력완료"> <p>
    <input type="reset" value="입력취소"> <p>
</form>
</body> </html>
```

oraDelete.jsp

```
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR" import="java.sql.*" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><head><meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>Insert title here</title></head>
<body>
<%
    String deptno = request.getParameter("deptno");
    String sql= String.format("delete from dept where deptno=%s",deptno);
    String driver = "oracle.jdbc.driver.OracleDriver";
    String url = "jdbc:oracle:thin:@localhost:1521:xe";
    Connection conn = null;    Statement stmt = null;
    try { Class.forName(driver);
        conn=DriverManager.getConnection(url,"scott","tiger");
        stmt = conn.createStatement();
        int result = stmt.executeUpdate(sql);
        out.println("삭제 성공 ㅋㅋㅋㅋ");

    }catch(Exception e) {
        out.println("삭제 실패 ㅠㅠ 메시지 :"+e.getMessage());
    } finally {
        stmt.close(); conn.close(); }

%>
</body>
</html>
```


- ❖ SQL의 틀을 미리 정해 놓고, 나중에 값을 지정하는 방식
- ❖ PreparedStatement의 일반적 사용

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

- ❖ 쿼리 실행 관련 메서드

- ❖ **ResultSet executeQuery()** - SELECT 쿼리를 실행할 때 사용되며 ResultSet을 결과값으로 리턴합니다.
- ❖ **int executeUpdate()** - INSERT, UPDATE, DELETE 쿼리를 실행할 때 사용되며, 실행 결과 변경된 레코드의 개수를 리턴합니다

메서드	설명
setString(int index, String x)	지정한 인덱스의 파라미터 값을 x로 지정
setInt(int index, int x)	지정한 인덱스의 파라미터 값을 int 값 x로 지정
setLong(int index, long x)	지정한 인덱스의 파라미터 값을 long 값 x로 지정
setDouble(int index, double x)	지정한 인덱스의 파라미터 값을 double 값 x로 지정
setFloat(int index, float x)	지정한 인덱스의 파라미터 값을 float 값 x로 지정
setTimestamp(int index, Timestamp x)	지정한 인덱스의 값을 SQL TIMESTAMAP 타입을 나타내는 java.sql.Timestamp 타입으로 지정
setDate(int index, Date x)	지정한 인덱스의 값을 SQL DATE 타입을 나타내는 java.sql.Date 타입으로 지정
setTime(int index, Time x)	지정한 인덱스의 값을 SQL TIME 타입을 나타내는 java.sql.Time 타입으로 지정

- ❖ 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

PreparedStatement

```
<%@ page language="java" contentType="text/html; charset=EUC-
KR"
    pageEncoding="EUC-KR" import="java.sql.*"
    errorPage="DBError.jsp" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html><head><meta http-equiv="Content-Type" content="text/html;
charset=EUC-KR">
<body>
<%
    request.setCharacterEncoding("euc-kr");
    String code = request.getParameter("code");
    String title = request.getParameter("title");
    String writer = request.getParameter("writer");
    String pr = request.getParameter("price");
    int price = Integer.parseInt(pr);
    Class.forName("oracle.jdbc.driver.OracleDriver");
```

```
Connection conn =DriverManager.getConnection(
    "jdbc:oracle:thin:@127.0.0.1:1521:orcl","scott","tiger");
String str="update goodsInfo set title=?,writer=?,price=?" +
    " where code=?";
PreparedStatement pstmt = conn.prepareStatement(str);
pstmt.setString(1,title);
pstmt.setString(2,writer);
pstmt.setInt(3,price);
pstmt.setString(4,code);
int rt = pstmt.executeUpdate();
if (rt<1) throw new Exception("수정 하는데 에러가 발생했습니다");
pstmt.close(); conn.close();
response.sendRedirect("updateResult.jsp?code="+code);
```

%>

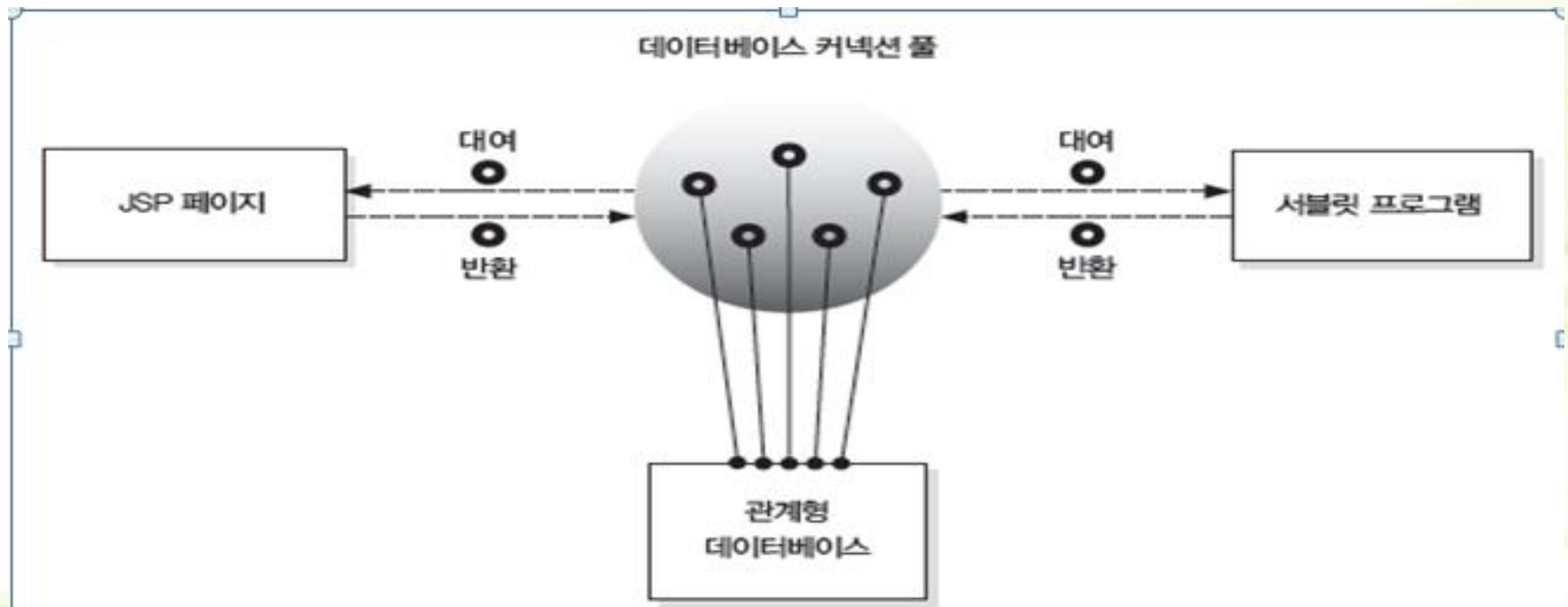
</body>

</html>

```
<%@ page language="java" contentType="text/html; charset=EUC-
KR"
    pageEncoding="EUC-KR"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><head><meta http-equiv="Content-Type" content="text/html;
charset=EUC-KR">
<title>Insert title here</title>
</head>
<body>
    <h3>수정 결과</h3>
    상품정보가 수정되었습니다.<br>
</body>
</html>
```

데이터베이스 커넥션 풀의 설치와 사용

- 데이터베이스에 동시에 접속할 수 있는 사용자 수는 한정되어 있는데, 웹 서버에는 동시에 수백, 수천의 사용자들이 접속할 수 있다.
- 웹 애플리케이션이 실행될 때마다 데이터베이스로 새로운 접속을 맺는 것은 현실적으로 불가능하므로, 데이터베이스 몇 개의 접속을 맺어서 데이터베이스 커넥션 풀(Database Connection Pool)에 저장해놓고, 필요한 웹 애플리케이션이 빌려 쓰고 반환하는 방식을 사용해야 한다.



커넥션 풀의 사용

❖ context.xml 파일에 데이터베이스 연결과 관련된 코드를 resource로 작성하고 web.xml 파일에서 불러들인 후 사용 가능

Context.xml – WebContent 폴더의 META-INF 폴더에 작성

<Context>

<Resource name="리소스 이름"

auth="권한"

type="리소스 종류" - 데이터베이스는 javax.sql.DataSource

username="계정"

password="비밀번호"

driverClassName="드라이버이름"

factory="팩토리클래스 이름"

url="자원의 위치"

maxActive="최대사용개수"

maxIdle="사용되지 않고 풀에 저장될 수 있는 최대 커넥션 개수"/>

minIdle="사용되지 않고 풀에 저장될 수 있는 최소 커넥션 개수"

whenExhaustedAction=" 커넥션 풀에서 가져올 수 없을 때의 동작으로 0이면 maxWait 만큼 기다리고 0이면 에러를 발생시키며 2이면 일시적으로 커넥션을 생성해서 사용"

maxWait="대기시간으로 1/1000초 단위"

timeBetweenEvictionRunsMills="사용되지 않는 커넥션의 추출 주기"

/>

</Context>

- ❖ web.xml에서 앞에서 작성한 context.xml의 내용 사용하기

<resource-ref>

<description>설명</description>

<res-ref-name>리소스이름</res-ref-name>

<res-type>리소스 종류</res-type>

<res-auth>권한</res-auth>

</resource-ref>

- ❖ JSP나 서블릿에서 사용

Context 변수1= new InitialContext();

DataSource 변수2 = (DataSource) 변수1.lookup("java:comp/env/리소스이름");

Connection변수 = 변수.getConnection();

context.xml 활용 : META-INF에 context.xml을 작성하여 넣음

```
<Context>
  <Resource name="jdbc/OracleDB"
    auth="Container"
    type="javax.sql.DataSource"
    username="scott"
    password="tiger"
    driverClassName="oracle.jdbc.driver.OracleDriver"
    factory="org.apache.tomcat.dbcp.dbcp.BasicDataSourceFactory"
    url="jdbc:oracle:thin:@127.0.0.1:1521:ORCL"
    maxActive="500"
    maxIdle="100"/>
</Context>
```

%. 과거에는 web.xml에 다음을 추가했으나 현재는 생략해도 됨

```
<resource-ref>
  <description>Connection</description>
  <res-ref-name>jdbc/OracleDB</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
</resource-ref>
```

```
<%@ page language="java" contentType="text/html; charset=EUC-KR"%>
<%@ page import="java.sql.*"%>
<%@ page import="javax.sql.*" %>
<%@ page import="javax.naming.*" %>
<%
    Connection conn = null;

    try {
        Context init = new InitialContext();
        DataSource ds = (DataSource)
            init.lookup("java:comp/env/jdbc/OracleDB");
        conn = ds.getConnection();

        out.println("<h3>연결되었습니다.</h3>");
    }catch(Exception e){
        out.println("<h3>연결에 실패하였습니다.</h3>");
        e.printStackTrace();
    }

%>
```

MetaData

```
<%@ page language="java" contentType="text/html; charset=EUC-KR"%>
<%@ page import="java.sql.*"%>
<%@ page import="javax.sql.*" %>
<%@ page import="javax.naming.*" %>
<%
    Connection conn = null;
    String sql="SELECT * FROM student";
    try { Context init = new InitialContext();
        DataSource ds = (DataSource) init.lookup("java:comp/env/jdbc/OracleDB");
        conn = ds.getConnection();
        PreparedStatement pstmt = conn.prepareStatement(sql);
        ResultSet rs=pstmt.executeQuery();
        ResultSetMetaData rsmd = rs.getMetaData();
        out.println("컬럼 수 : "+rsmd.getColumnCount()+"<br>");
        for(int i=1;i<=rsmd.getColumnCount();i++){
            out.println(i+"번째 컬럼의 이름 : " + rsmd.getColumnName(i)+"<br>");
            out.println(i+"번째 컬럼의 타입 이름 : "
                +rsmd.getColumnTypeName(i)+"<br>");
        }
    }catch(Exception e){
        e.printStackTrace();
    }
%>
```

Cursor

```
<%@ page language="java" contentType="text/html; charset=EUC-KR"%>
<%@ page import="java.sql.*"%>
<%@ page import="javax.sql.*" %>
<%@ page import="javax.naming.*" %>
<%
    Connection conn = null;
    String sql="SELECT * FROM student";
    try {
        Context init = new InitialContext();
        DataSource ds = (DataSource) init.lookup("java:comp/env/jdbc/OracleDB");
        conn = ds.getConnection();
        PreparedStatement pstmt=
            conn.prepareStatement(sql,ResultSet.TYPE_SCROLL_SENSITIVE,
                ResultSet.CONCUR_UPDATABLE);
        ResultSet rs=pstmt.executeQuery();
        rs.last();
        out.println(rs.getInt(1)+", "+rs.getString(2)+"<br>");
        rs.first();
        out.println(rs.getInt(1)+", "+rs.getString(2)+"<br>");
        rs.absolute(3);
        out.println(rs.getInt(1)+", "+rs.getString(2)+"<br>");
    }catch(Exception e){ out.println("<h3>데이터 가져오기에 실패하였습니다.</h3>");
        e.printStackTrace();
    }
%>
```

트랜잭션 관리

- 트랜잭션은 데이터베이스에서 한꺼번에 처리되는 작업의 단위
- 트랜잭션은 시작되서 commit되기 전에 에러가 발생하면 작업내용이 데이터베이스에 반영되지 않고 취소(rollback)됩니다.
- 자바의 JDBC는 기본적으로 autoCommit이 설정되어 있어서 하나의 작업을 수행하면 즉시 데이터베이스에 반영되도록 설정되어 있습니다.
- Connection 객체의 setAutoCommit 메서드를 이용해서 이를 해제하거나 설정할 수 있습니다.
- 또한 commit()과 rollback()을 이용해서 트랜잭션을 완료하거나 롤백 시킬 수 있습니다.

Transaction

```
<%@ page language="java" contentType="text/html; charset=EUC-KR"%>
<%@ page import="java.sql.*"%><%@ page import="javax.sql.*" %>
<%@ page import="javax.naming.*" %>
<%
    Connection conn = null;          PreparedStatement pstmt = null;
    ResultSet rs = null;
    String sql="INSERT INTO student (num, name) VALUES (12,'홍길동')";
    String sql2="SELECT * FROM student WHERE num=11";
    try {
        Context init = new InitialContext();
        DataSource ds = (DataSource) init.lookup("java:comp/env/jdbc/OracleDB");
        conn = ds.getConnection();      conn.setAutoCommit(false);
        pstmt=conn.prepareStatement(sql);   pstmt.executeUpdate();
        pstmt.close();
        pstmt=conn.prepareStatement(sql2);
        rs=pstmt.executeQuery();
        if(!rs.next()){
            conn.rollback();
            out.println("<h3>데이터 삽입에 문제가 발생하여 롤백하였습니다.</h3>");
        }else{ conn.commit();
            out.println("<h3>데이터 삽입이 모두 완료되었습니다.</h3>");    }
        pstmt.close();
        conn.setAutoCommit(true);
    }catch(Exception e){ out.println("<h3>데이터 삽입에 실패하였습니다.</h3>");
        e.printStackTrace();    }
%>
```


Procedure의 사용 과정에서 **Compile** 단계까지 실행을 한 후에 java의 **CallableStatement** 객체를 이용하여 받아와서 처리하는 것

```
Connection con=DriverManager.getConnection(url,id,password);
cstmt = con.prepareCall("{call plus( ?, ?, ?)}");
//bind변수 채우기
//in parameter 값 채우기 : setString(bind변수번호, 입력값)
cstmt.setInt(1, 10); //첫 번째 bind 변수의 ? = 10
cstmt.setInt(2, 20); //두 번째 bind 변수의 ? = 20

//out parameter 값 채우기 : 직접 실행 시 out 변수를 넣는parameter
cstmt.registerOutParameter(3, Types.NUMERIC);
//DB의 데이터형이 varchar2이거나, text의 문자열이면 VARCHAR로 TYPE을 설정한다.
// ( Types.VARCHAR)
//DB의 데이터형이 number, int 등의 정수이면 NUMERIC으로 설정한다. (Types.NUMERIC)

//쿼리 실행 시execute()로 처리한다
cstmt.execute();

//out 변수에 들어간 값 받기 : getString/getInt(out변수의 bind번호)
int result = cstmt.getInt(3);

System.out.println("프로시저에서 처리한 결과"+result);
```


callableStatement

callableStatement는 SQL의 스토어드 프로시저를 실행시키기 위해 사용되는 인터페이스
스토어드프로시저란 query문을 하나의 파일형태로 만들거나 데이터베이스에 저장해 놓고 함수
처럼 호출해서 사용 하는 것이다 이것을 이용하면 연속되는 query문에 대해서 매우 빠른 성능을
보인다

장점 : 자바코드에 쿼리문이 들어가지 않아 보다 간결해진다

단점 : 저장 프로시저에서 선언한 순서대로 파라미터를 넘겨야 한다

```
/*  
create or replace function employ_ename (e_empno in emp.empno%TYPE) return  
varchar2  
is  
e_ename varchar2(10) := '홍길동';  
begin  
select ename into e_ename from emp  
where empno = e_empno;  
return e_ename;  
exception  
when no_data_found or too_many_rows then  
return e_ename;  
end;  
/
```

```
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR" import="java.sql.*,javax.sql.*,javax.naming.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><head><meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>Insert title here</title></head><body>
<%
    Context init = new InitialContext();
    DataSource ds = (DataSource)init.lookupLink("java:comp/env/jdbc/OracleDB");
    Connection conn = ds.getConnection();
    String call = "{? = call employ_ename(?) }";
    CallableStatement c = conn.prepareCall(call);
    // IN parameter설정
    c.setInt(2, 7566);
    // Out parameter의 Type설정
    c.registerOutParameter(1, java.sql.Types.VARCHAR );
    // CallableStatement실행
    c.execute();
    // Out parameter의 값을 얻고, 출력한다.
    out.println("result : " + c.getString(1));
    conn.close();
%>
</body>
</html>
```

```
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR" import="ch10.TestCallable"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><head><meta http-equiv="Content-Type" content="text/html;
charset=EUC-KR">
<title>Insert title here</title>
</head>
<body>
<%
    TestCallable tc = new TestCallable();
    tc.setEmpno(7369);
    String result = tc.call();
%>
결과 : <%=result %>
</body>
</html>
```

```
package ch10;
import java.sql.*; import javax.naming.*; import javax.sql.*;
public class TestCallable {
    private int empno;
    public void setEmpno(int empno) { this.empno = empno; }
    public String call() throws Exception {
        Context init = new InitialContext();
        DataSource ds = (DataSource)init.lookupLink("java:comp/env/jdbc/OracleDB");
        Connection conn = ds.getConnection();
        String call = "{? = call employ_ename(?) }";
        CallableStatement c = conn.prepareCall(call);
        // IN parameter설정
        c.setInt(2, empno);
        // Out parameter의 Type설정
        c.registerOutParameter(1, java.sql.Types.VARCHAR );
        // CallableStatement실행
        c.execute();
        // Out parameter의 값을 얻고, 출력한다.
        String result = c.getString(1);
        conn.close();
        return result;
    }
}
```

Blob 데이터 jsp로 입력

```
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR" import="java.sql.*,javax.sql.*,javax.naming.*,java.io.*" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><head><meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>Insert title here</title></head>
<body>
<%PreparedStatement pstmt = null;
    try{ Context cont = new InitialContext();
        DataSource ds = (DataSource)cont.lookup("java:comp/env/jdbc/OracleDB");
        Connection conn = ds.getConnection();
        File file = new File("C:/Users/Public/Pictures/Sample Pictures/kk.jpg");
        InputStream is = new FileInputStream(file);
        int fileSize = (int)file.length();
        String sql= "INSERT INTO TEST VALUES('bb','1234', ? )";
        PreparedStatement psmt = conn.prepareStatement(sql);
        psmt.setBinaryStream(1, is, fileSize);
        int count =psmt.executeUpdate();
        is.close(); psmt.close(); conn.close();
        out.println("입력 성공");
    }catch(Exception e) {out.println("에러 " + e.getMessage());}
%>
</body>
</html>
```

Blob 데이터 jsp로 보기

blob_view.jsp

```
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>Insert title here</title>
</head>
<body>

</body>
</html>
```

blob_out.jsp

```
<body>
<%PreparedStatement pstmt = null;      ResultSet rs = null;
   String id = request.getParameter("id"); InputStream in = null;
   BufferedOutputStream bos = null;
   try{ Context cont = new InitialContext();
       DataSource ds = (DataSource)cont.lookup("java:comp/env/jdbc/OracleDB");
       Connection conn = ds.getConnection();
       String sql= "select photo from test where id=?;
       PreparedStatement psmt = conn.prepareStatement(sql);
       pstmt.setString(1,id);
       rs=psmt.executeQuery();
       if(rs.next()){
           Blob blob=rs.getBlob(1);
           out.clear();
           out=pageContext.pushBody();
           bos = new BufferedOutputStream(response.getOutputStream());
           in = blob.getBinaryStream();   int length = (int)blob.length();
           byte[] buffer=new byte[length];
           while((length=in.read(buffer)) !=-1){   bos.write(buffer);   }
       }
       bos.close(); in.close(); psmt.close(); conn.close();
   }catch(Exception e) {out.println("에러 " + e.getMessage());}
%>
</body></html>
```


- ❖ jsp에서 그림 등 다운로드를 위하여 OutputStream을 받아서 처리할 때 jsp에서 servlet 으로 변환할 때 내부적으로 out객체가 생성된다
- ❖ 그래서 response.getOutputStream() 호출할 때 생기는 out과 jsp에서 servlet으로 변환할 때 생성되는 out와 충돌된다
- ❖ 예방하기 위하여
out.clear();
out=pageContext.pushBody();을
- ❖ response.getOutputStream() 위에 넣는다