JSP 2.3 & Servlet 3.1

DB 연동

- 김근형 -

DB 연동

- DB는 기존 Java의 JDBC를 연동하는 것과 동일하다.
- 단지 차이점이라면 해당 환경이 JSP 환경이라는 점 하나만 다르다.
- 해당 JDBC의 드라이브 클래스를 얻기 위해 맨 처음 ClassFor를 사용한다
- 각 DB에 따라서 jdbc driver와 url만 달라질 뿐 큰 차이는 존재하지 않는다.

DB 연동

- 오라클(jdbcTest.jsp)

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

    String driver="oracle.jdbc.driver.OracleDriver";
    String url="jdbc:oracle:thin:@localhost:1521:XE";

    Boolean connect=false;

    try{
        Class.forName(driver);
        conn=DriverManager.getConnection(url,"java","java");

        connect=true;

        conn.close();
    }catch(Exception e){
        connect=false;
        e.printStackTrace();
    }
%>
```

```jsp
<html>
<head>
<title>JDBC 연동 테스트 예제</title>
</head>
<body>
<h3>
<%if(connect==true){ %>
    연결되었습니다.
<%}else{ %>
    연결에 실패하였습니다.
<%} %>
</h3>
</body>
</html>
```

DB 연동

- Mysql(mySQLJdbcTest.jsp)

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

    String driver="com.mysql.jdbc.Driver";
    String url="jdbc:mysql://localhost:3306/testDB";

    Boolean connect=false;

    try{
        Class.forName(driver);
        conn=DriverManager.getConnection(url,"java","java");

        connect=true;

        conn.close();
    }catch(Exception e){
        connect=false;
        e.printStackTrace();
    }
%>
```

```
<html>
<head>
<title>JDBC 연동 테스트 예제</title>
</head>
<body>
<h3>
<%if(connect==true){ %>
    연결되었습니다.
<%}else{ %>
    연결에 실패하였습니다.
<%} %>
</h3>
</body>
</html>
```

DB 연동

- mariaDB(mariaDBJdbcTest.jsp)

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

    String driver="com.mysql.jdbc.Driver";
    String url="jdbc:mysql://localhost:3306/testDB";

    Boolean connect=false;

    try{
        Class.forName(driver);
        conn=DriverManager.getConnection(url,"java","java");

        connect=true;

        conn.close();
    }catch(Exception e){
        connect=false;
        e.printStackTrace();
    }
%>
```

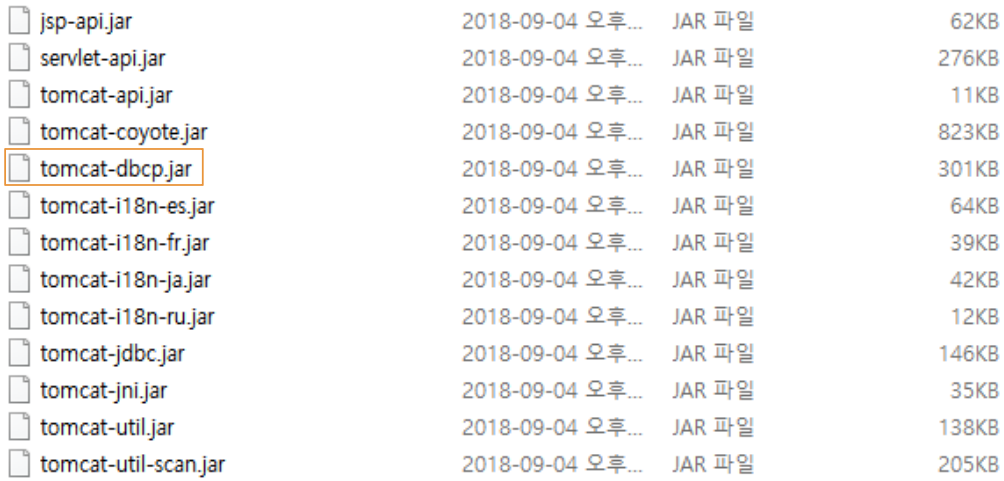
```jsp
<html>
<head>
<title>JDBC 연동 테스트 예제</title>
</head>
<body>
<h3>
<%if(connect==true){ %>
    연결되었습니다.
<%}else{ %>
    연결에 실패하였습니다.
<%} %>
</h3>
</body>
</html>
```

JNDI(Java Naming and Directory Interface)

- JDBC를 연동하기 위해서는 드라이버를 로드하고 JDBC URL로 접속하여 Connection 객체를 얻어오는 단계를 거쳐야 한다.
- Connection Pool 이란 데이터베이스와 연결된 Connection 객체를 미리 생성하여 풀(Pool) 속에 저장해두고 필요할 때마다 이 풀에 접근하여 Connection 객체를 사용하고, 작업이 끝나면 다시 반환하는 것을 말한다.
- 이렇게 되면 커넥션을 생성하는데 드는 연결 시간이 단축되며 다른 사용자가 사용하지 않는 커넥션을 재사용 할 수 있기 때문에 사용자가 접속할 때마다 계속해서 커넥션을 생성할 필요가 없다.

JNDI(Java Naming and Directory Interface)

- JNDI란 명명 서비스 및 디렉토리 서비스에 접근하기 위한 API를 말한다.
- 특정 자원에 접근하기 위한 이름으로 사용한다.
- 톰캣에는 톰캣 설치 경로의 lib 디렉토리에 CP(Connection Pool) 기능을 제공하기 위해 DBCP API를 지정한다.



JNDI(Java Naming and Directory Interface)

- META-INF 디렉토리 밑에 context.xml 파일을 생성해서 서버에 공유할 리소스를 정의한다.
- META-INF/context.xml(Oracle)

```
<Context>
  <Resource name="jdbc/OracleDB"
   auth="Container"
   type="javax.sql.DataSource"
   username="java"
   password="java"
   driverClassName="oracle.jdbc.driver.OracleDriver"
   factory="org.apache.tomcat.dbcp.dbcp2.BasicDataSourceFactory"
   url="jdbc:oracle:thin:@127.0.0.1:1521:XE"
   maxActive="500"
   maxIdle="100"/>
</Context>
```

JNDI(Java Naming and Directory Interface)

- META-INF 디렉토리 밑에 context.xml 파일을 생성해서 서버에 공유할 리소스를 정의한다.
- META-INF/context.xml(mariaDB)

```
<Resource name="jdbc/mariaDB"
          auth="Container"
          driverClassName="org.mariadb.jdbc.Driver"
          username="root"
          password="1234"
          factory="org.apache.tomcat.dbcp.dbcp2.BasicDataSourceFactory"
          type="javax.sql.DataSource"
          url="jdbc:mariadb://localhost:3306/board"
          maxActive="500"
          maxIdle="100"/>
```

JNDI(Java Naming and Directory Interface)

- dbcpAPITest.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"%>
<%@ 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/mariaDB");
        conn = ds.getConnection();

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

JNDI(Java Naming and Directory Interface)

- web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSc
  <display-name>Chapter13</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
  <resource-ref>
      <description>DB Connection</description>
      <res-ref-name>jdbc/mariaDB</res-ref-name>
      <res-type>javax.sql.DataSource</res-type>
      <res-auth>Container</res-auth>
  </resource-ref>
</web-app>
```

Statement test

- statementTest.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"%>
<%@ page import="java.sql.*"%>
<%@ page import="javax.sql.*" %>
<%@ page import="javax.naming.*" %>
<%
    Connection conn = null;
    String sql="INSERT INTO student (num,name) VALUES (7,'홍길동')";
    Statement stmt = null;
    try {
        Context init = new InitialContext();
        DataSource ds = (DataSource) init.lookup("java:comp/env/jdbc/mariaDB");
        conn = ds.getConnection();
        stmt=conn.createStatement();

        int result=stmt.executeUpdate(sql);
        if(result!=0){
            out.println("<h3>레코드가 등록되었습니다.</h3>");
        }
    }catch(Exception e){
        out.println("<h3>레코드 등록에 실패하였습니다.</h3>");
        e.printStackTrace();
    }
    finally{
        try{
            stmt.close();
            conn.close();
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }
%>
```

PreparedStatement

- preparedStatementTest.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"%>
<%@ page import="java.sql.*"%>
<%@ page import="javax.sql.*" %>
<%@ page import="javax.naming.*" %>
<%
    Connection conn = null;
    String sql="INSERT INTO student (num,name) VALUES (?,'홍길동')";
    PreparedStatement pstmt = null;
    try {
        Context init = new InitialContext();
        DataSource ds = (DataSource) init.lookup("java:comp/env/jdbc/mariaDB");
        conn = ds.getConnection();
        pstmt=conn.prepareStatement(sql);

        for(int i=8;i<=11;i++){
            pstmt.setInt(1,i);
            if(pstmt.executeUpdate()!=0){
                out.println("<h3>"+i+"번 레코드를 등록하였습니다.</h3>");
            }
        }
    }catch(Exception e){
        out.println("<h3>레코드 등록에 실패하였습니다.</h3>");
        e.printStackTrace();
    }
    finally{
        try{
            pstmt.close();
            conn.close();
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }
%>
```

ResultSet

- resultSetTest.jsp

```jsp
<%@ page import="javax.naming.*" %>
<%
    Connection conn = null;
    String sql="SELECT * FROM student";
    PreparedStatement pstmt= null;
    ResultSet rs=null;

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

        pstmt=conn.prepareStatement(sql);
        rs=pstmt.executeQuery();

        while(rs.next()){
            out.println("<h3>"+rs.getInt(1)+","+rs.getString(2)+"</h3>");
        }

    }catch(Exception e){
        out.println("<h3>데이터 가져오기에 실패하였습니다.</h3>");
        e.printStackTrace();
    }finally{
        try{
            rs.close();
            pstmt.close();
            conn.close();
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }
%>
```

ResultSetMetaData

- ResultSet으로 얻어온 레코드들의 메타 정보에 해당하는 칼럼의 정보들을 제공한다.
- ResultSetMetaData 객체를 사용하게 되면 컬럼 수나 각 컬럼 이름, 컬럼 타입 등의 정보를 쉽게 알아낼 수 있다.

메소드	설명
getColumnCount()	ResultSet에 저장되어 있는 테이블의 컬럼의 수를 반환한다
getColumnLabel(int column)	해당 번호의 컬럼의 라벨(title)을 반환한다
getColumnName(int column)	해당 번호의 컬럼의 이름을 반환한다
getColumnType(int column)	해당 번호의 데이터 타입을 int 형으로 반환한다
getColumnTypeName(int column)	해당 번호의 칼럼의 데이터 타입을 String 형으로 반환한다.

ResultSetMetaData

- resultSetMetaDataTest.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"%>
<%@ page import="java.sql.*"%>
<%@ page import="javax.sql.*" %>
<%@ page import="javax.naming.*" %>
<%
    Connection conn = null;
    String sql = "SELECT * FROM student";
    PreparedStatement pstmt = null;
    ResultSet rs = null;
    ResultSetMetaData rsmd = null;

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

        pstmt = conn.prepareStatement(sql);
        rs = pstmt.executeQuery();
        rsmd = rs.getMetaData();

        out.println("칼럼 수 : "+rsmd.getColumnCount()+"<br>");
        for(int i=1;i<=rsmd.getColumnCount();i++){
            out.println(i+"번째 칼럼의 이름 : " + rsmd.getColumnName(i)+" : ");
            out.println(i+"번째 칼럼의 타입 이름 : " +rsmd.getColumnTypeName(i)+"<br>");
        }
```

ResultSet의 커서 자유롭게 움직이기

- ResultSet 관련 메서드

메소드	설명
Absolute(int rowNum)	지정된 위치로 커서를 이동한다
beforeFirst()	커서를 처음 레코드 이전 위치로 이동한다. 커서를 실제 레코드가 아닌 ResultSet 객체의 처음 부분으로 이동한다.
afterLast()	커서를 마지막 레코드 이후 위치로 이동한다. 커서를 실제 레코드가 아닌 ResultSet 객체의 끝 부분으로 이동한다.
First()	처음 레코드가 존재하는 행으로 이동한다.
Last()	마지막 레코드가 존재하는 행으로 이동한다.
Next()	다음 레코드 행으로 이동한다
Previous()	이전 레코드 행으로 이동한다

ResultSet의 커서 자유롭게 움직이기

- resultSetCursorTest.java

```
<%
    Connection conn = null;
    String sql = "SELECT * FROM student";
    PreparedStatement pstmt = null;
    ResultSet rs = null;

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

        pstmt = conn.prepareStatement(sql,ResultSet.TYPE_SCROLL_SENSITIVE,
                                          ResultSet.CONCUR_UPDATABLE);
        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>");
```

Transaction

- Transaction 예제 (transactionTest.java)

```
Connection conn = null;
PreparedStatement pstmt = null;
ResultSet rs = null;
String sql="INSERT INTO student (num, name) VALUES (13,'홍길동')";
String sql2="SELECT * FROM student WHERE num=12";
try {
    Context init = new InitialContext();
    DataSource ds = (DataSource) init.lookup("java:comp/env/jdbc/mariaDB");
    conn = ds.getConnection();

    conn.setAutoCommit(false);

    pstmt=conn.prepareStatement(sql);
    pstmt.executeUpdate();

    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>");
    }

    conn.setAutoCommit(true);
}catch(Exception e){
    out.println("<h3>데이터 삽입에 실패하였습니다.</h3>");
    e.printStackTrace();
```

회원 관리 시스템

- sql

```
create table java.MEMBER (
    ID int,
    PASSWORD varchar(20),
    NAME varchar(15),
    AGE int,
    GENDER varchar(5),
    EMAIL varchar(30),
    unique key (ID)
) engine=InnoDB character set = utf8;
```

회원 관리 시스템

- 파일 목록

웹 페이지	설명
loginForm.jsp	로그인 폼 페이지(로그인 정보를 입력하는 페이지)
joinForm.jsp	회원 가입 폼 페이지(회원가입 정보를 입력하는 페이지)
loginProcess.jsp	로그인을 실제로 처리하는 페이지
joinProcess.jsp	회원가입을 실제로 처리하는 페이지
main.jsp	메인페이지
member_list.jsp	회원 목록을 확인하는 페이지
member_info.jsp	특정 회원의 정보를 보여주는 페이지
member_delete.jsp	특정 회원의 정보를 삭제하는 페이지