JSP 4-1

JSP 3-4

🛠️ JDBC 고급 사용법: executeUpdate(), PreparedStatement, DAO/DTO

🔄 executeUpdate() 메소드

> DB 테이블의 내용을 **변경(insert, update, delete)** 할 때 사용

- DML 문장 실행 시 사용
- 영향을 받은 레코드 수(int) 반환

```
int rows = stmt.executeUpdate("DELETE FROM member WHERE id = 'user1'");
System.out.println(rows + "개의 행이 삭제되었습니다.");
```

🧩 PreparedStatement

> `Statement`의 하위 인터페이스로, **미리 SQL 문장을 준비(컴파일)** 해두고 변수만 바꿔서 실행 가능

특징

- SQL 문장이 **미리 컴파일**되어 실행 속도 향상
- SQL 삽입 공격 방지에 효과적 (보안성 ↑)
- `?`로 자리 표시자 사용 후 `setXxx()` 메서드로 값 대입

```
String sql = "INSERT INTO member(id, pw) VALUES (?, ?)";
PreparedStatement pstmt = conn.prepareStatement(sql);
pstmt.setString(1, "admin");
pstmt.setString(2, "1234");
int result = pstmt.executeUpdate();
```

```jsp
<%!
String id, pw, name, email, address;

Connection conn;
PreparedStatement pstmt;

String driver = "oracle.jdbc.OracleDriver";
String url="jdbc:oracle:thin:@localhost:1521:xe";
String dbID = '[redacted]';
String dbPW = [redacted]
%>

<%
request.setCharacterEncoding("UTF-8");
id = request.getParameter("id");
pw = request.getParameter("pw");
name = request.getParameter("name");
email = request.getParameter("email");
address = request.getParameter("address");
```

```jsp
try
{
    String query = "INSERT INTO KGMEMBER (ID, PW, NAME, EMAIL, ADDRESS) VALUES (?, ?, ?, ?, ?)";


    Class.forName(driver);
    conn = DriverManager.getConnection(url, dbID, dbPW);
    pstmt = conn.prepareStatement(query);

    //값 넣어주기
    //PreparedStatement의 setter 메소드로
    //첫 번째 인자로는 ?에 대한 인덱스
    //두 번째 인자로는 값을 넣어준다
    //?에 대한 인덱스는 0부터가 아닌 1부터 시작된다
    pstmt.setString(1, id);
    pstmt.setString(2, pw);
    pstmt.setString(3, name);
    pstmt.setString(4, email);
    pstmt.setString(5, address);

    //insert, delete, update문 사용할 때 사용
    //변경된 레코드의 수 반환
    int result = pstmt.executeUpdate();

    if(result == 1)
    {

        System.out.println("회원가입이 되었습니다.");
        response.sendRedirect("memberList.jsp");
    }
    else
        System.out.println("회원가입 실패");
```

🗂️ DAO와 DTO

✅ DAO (Data Access Object)

> DB에 직접 접근해서 **데이터 추가/수정/삭제 등 처리하는 클래스**

- DB 연결, SQL 실행 등 데이터 작업 전담
- 비즈니스 로직과 분리하여 구조화

✅ DTO (Data Transfer Object)

> 데이터 이동을 위한 **순수 자바 객체**

- 계층 간 데이터 전달에 사용
- 필드 + getter/setter 구성
- 자바빈(JavaBean) 형태로 사용

```
public class MemberDTO {
    private String id;
    private String name;
    // getter / setter
}
```

```java
package com.jsp.member;

import java.sql.Timestamp;

public class MemberDTO {
//DTO: Data Transfer Object
//계층간 데이터 교환을 위한 자바 빈즈

private String id;
private String pw;
private String name;
private String email;
private String address;
private Timestamp regDate;

public MemberDTO() {

}


public String getId() {
    return id;
}
public void setId(String id) {
    this.id = id;
}
public String getPw() {
    return pw;
}
public void setPw(String pw) {
    this.pw = pw;
}
public String getName() {
    return name;
}
public void setName(String name) {
    this.name = name;
}
public String getEmail() {
    return email;
}
public void setEmail(String email) {
    this.email = email;
}
public String getAddress() {
    return address;
}
public void setAddress(String address) {
    this.address = address;
}
public Timestamp getRegDate() {
    return regDate;
}
public void setRegDate(Timestamp regDate) {
    this.regDate = regDate;
}



}
```

```java
package com.jsp.member;

import java.sql.*;

public class MemberDAO
{
//  DAO: Data Access Object
//  데이터베이스에 접근하여 CRUD를 처리하는 객체

//  싱글톤: 단 하나의 객체만 생성하는 것
//  싱글톤을 만들려면 클래스 외부에서 new 연산자를 호출할 수 없도록 해야된다
//  외부에서 생성자 호출을 막기위해 private을 붙여준다
//  외부에서 객체를 얻는 방법은 getinstance() 메소드를 호출하는 방법이다
//  getInstance() 메소드는 단 하나의 객체만 리턴한다

//  싱글톤 쓰는 이유
//  고정된 메모리 영역을 얻으면서 한번의 new로 인스턴스 사용하기 때문에
//  메모리 낭비를 방지할 수 있다
//  싱글톤으로 만드렁진 클래스의 인스턴스는 전역 인스턴스이기 때문에
//  다른 클래스의 인스턴스들이 데이터를 공유하기 쉽다
//  DECP(DataBase Connection Pool)처럼 공통된 객체를
//  여러 개 생성해서 사용해야하는 상황에서 많이 사용한다


    private Connection conn;
    private PreparedStatement pstmt;
    private ResultSet rs;

    private String driver = "oracle.jdbc.OracleDriver";
    private String url="jdbc:oracle:thin:@localhost:1521:xe";
    private String dbID = "[redacted]";
    private String dbPW = "[redacted]";
```

```java
    private static MemberDAO instance = new MemberDAO();

    public static MemberDAO getInstance()
    {
        return instance;
    }

    //회원관리
    //회원가입 > 로그인 / 로그아웃 > 회원정보 열기 >회원 정보 수정 > 회원 삭제


    public int memberInsert(MemberDTO dto)
    {
        int result = 0;
        String query ="INSERT INTO KGMEMBER (ID, PW, NAME, EMAIL, ADDRESS) VALUES(?, ?, ?, ?, ?)";


        try
        {
            Class.forName(driver);
            conn = DriverManager.getConnection(url, dbID, dbPW);
            pstmt = conn.prepareStatement(query);

            pstmt.setString(1, dto.getId());
            pstmt.setString(2, dto.getPw());
            pstmt.setString(3, dto.getName());
            pstmt.setString(4, dto.getEmail());
            pstmt.setString(5, dto.getAddress());

            result = pstmt.executeUpdate();
```

```java
            pstmt.setString(1, dto.getId());
            pstmt.setString(2, dto.getPw());
            pstmt.setString(3, dto.getName());
            pstmt.setString(4, dto.getEmail());
            pstmt.setString(5, dto.getAddress());

            result = pstmt.executeUpdate();
```

```
66
67
68          }
69          catch(Exception e)
70          {
71              e.printStackTrace();
72          }
73          finally
74          {
75              try
76              {
77                  if(pstmt != null)
78                      pstmt.close();
79                  if(conn != null)
80                      conn.close();
81              }
82              catch(Exception e2) {}
83          }
84
85          return result;
86      }
87
```