



# **JSP 4강 – 답변형 게시판 만들기**

---

**양 명 속**

**[[now4ever7@gmail.com](mailto:now4ever7@gmail.com)]**



# 목차

---

- 커넥션 풀
- 싱글톤 패턴
- 답변형 게시판 만들기
  - DAO, VO(DTO, Bean)
- 트랜잭션 처리



# 커넥션 풀(Connection Pool)

---



## 커넥션 풀

- 커넥션 풀(Connection Pool)의 개요
  - 데이터베이스에 연결하기 위한 커넥션 객체는 새로 만들어질 때 많은 시스템 자원이 요구됨
  - 객체는 메모리에 적재가 되는데, 메모리에 객체를 할당할 자리를 만들고, 객체가 사용할 여러 자원들에 대한 초기화 작업, 객체가 필요 없게 되면 객체를 거두어 들여야 하는 작업 등이 요구되어서 객체의 생성 작업은 많은 비용을 요구함
  - 데이터베이스 커넥션은 데이터베이스에 한 번 연결하기 위한 작업 - 매번 새로운 데이터베이스 연결에 대한 요청이 들어올 때마다 이러한 작업들을 수행해야 한다면 많은 부담이 됨



# 커넥션 풀(Connection Pool)의 개요

- 커넥션 객체를 생성하고 관리하는 방법
  - (1) service method(doGet, doPost)에서 커넥션 객체를 생성
    - 주로 데이터베이스와 연동하기 위해 사용했던 방법
    - 문제점 - 커넥션 객체의 레퍼런스 변수는 지역변수에 할당됨
      - 이 경우 요청당 한 개씩 커넥션 객체가 생성되어서 시스템의 부하가 커지고 메모리가 낭비됨
      - 커넥션 시간이 요청 시간에 포함됨
  - (2) init method 에서 커넥션 객체를 생성
    - 서비스될 때 단 한 번만 사용하는 init method에서 커넥션 객체를 생성함
    - 커넥션 풀 없이 자바빈에서 커넥션 객체를 생성하는 예제들에서 볼 수 있음
    - 커넥션 객체의 레퍼런스 변수는 전역변수에 할당됨
    - init method에서 생성하므로 커넥션 시간이 안 걸리나 하나의 커넥션을 쓰면 커넥션에 쿼리가 쌓이게 되어 응답시간이 증가함



# 커넥션 풀(Connection Pool)의 개요

## ■ (3) 커넥션 풀에서 커넥션 객체 생성 및 관리하기

- 자원을 빌려 쓰고 회수하는 방법을 씀
- 커넥션 풀에 미리 여분의 커넥션을 만들어 놓고, 사용자의 요청이 있으면 메모리에 이미 만들어져 있는 커넥션을 부여하고, 사용된 커넥션 객체는 다시 커넥션 풀로 회수됨

- 커넥션 생성의 개수를 결정할 수 있음
- 권장하는 형태

## ■ 커넥션 풀

- 끊임없이 생성되는 커넥션의 문제를 해결하기 위한 목적
- 반드시 컨테이너에 1개만 만들어지도록 패턴을 만들어야 함
- 컨테이너가 자동지원을 못해서 커넥션 풀을 작성해야 하는 경우 - 커넥션 객체를 저장하고 있는 저장소인 커넥션 풀은 벡터, 맵등을 사용해서 구현함



# 커넥션 풀의 전략

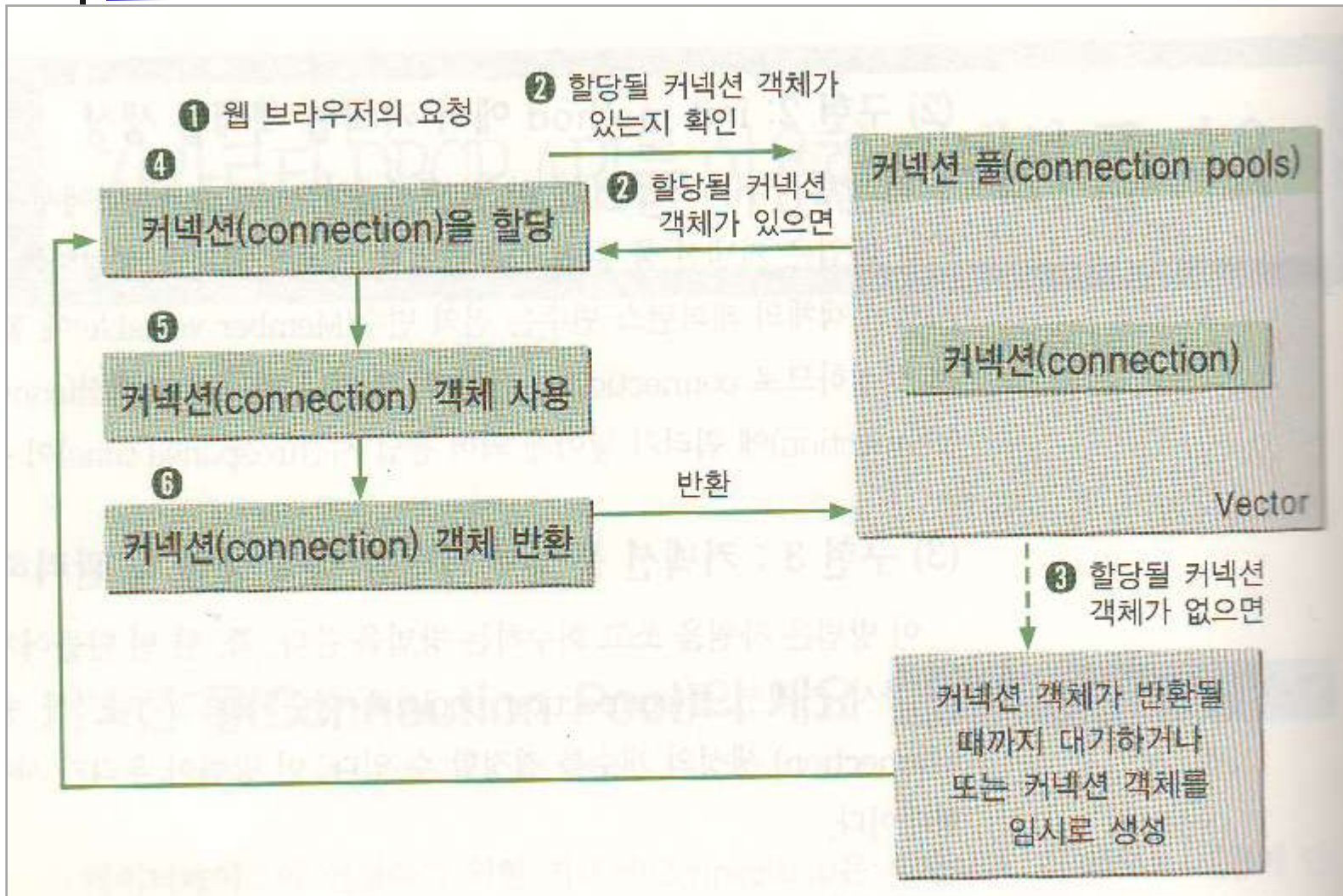
- 1) service() 메서드당 1개씩은 가지고 쓰게 함
  - service() 메서드는 요청당 1개씩 매핑됨
- 2) 커넥션의 개수를 제한함
- 3) 커넥션 객체 관리자가 다 쓰면 자원을 회수함
- 톰캣 컨테이너는 커넥션 풀을 제공함 - 이것을 사용

- 커넥션 풀 사용방법

- [1] 직접 작성하여 사용

- [2] 컨테이너나, **WAS**차원에서 제공하는 커넥션 풀을 사용

# 커넥션풀의 구현방법





```
package com.mystudy.db;
import java.sql.*;
import java.util.*;
public class ConnectionPoolMgr{
    private String url,user,pwd;
    private HashMap<Connection, Boolean> hmap;
    private int increment; //증가치

    //생성자
    public ConnectionPoolMgr(){
        increment=5;//5만큼씩 증가
        hmap=new HashMap<Connection,Boolean>(10);
        Connection con=null;

        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            System.out.println("드라이버 로딩 성공!");
            url="jdbc:oracle:thin:@192.168.0.105:1521:orcl";
            user="herb";
            pwd="herb123";

            //커백션 객체를 미리 생성해 놓기 - 10개
            for(int i=0;i<10;i++){
                con=DriverManager.getConnection(url,user,pwd);
                hmap.put(con, Boolean.FALSE); //해시맵의 key에 커백션 저장
                //해시맵의 value에 true, false 저장, false - 쉬는 커백션이라는 표시
            }
            System.out.println("ConnectionPool 생성!");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
            System.out.println("Class Not Found!");
        }
    }
}
```

```

        }catch (SQLException e) {
            e.printStackTrace();
            System.out.println("sql 예외발생!");
        }
    }//생성자

    public synchronized Connection getConnection() //jsp - 요청시 Thread로 처리
        throws SQLException{
        Iterator<Connection> iterKeys=hmap.keySet().iterator();
        Connection con=null;
        while(iterKeys.hasNext() ){ //hmap에 key가 있는 동안 반복
            con=iterKeys.next();//key값
            Boolean b=hmap.get(con);//value값
            //만약 쉬고있는 컨넥션이라면 일하는 컨넥션으로 표시해주고 반환한다.

            if(b==Boolean.FALSE){
                hmap.put(con, Boolean.TRUE);//일한다고 표시
                return con; //일하러 나감
            }
        }//while

        //쉬고 있는 컨넥션이 없으면 일할 Connection을 5개 증가시킨다
        for(int i=0;i<increment;i++){
            Connection con2=DriverManager.getConnection(url,user,pwd);
            hmap.put(con2, Boolean.FALSE);
        }//for

        return getConnection();//재귀호출
    }

```

//커백션을 사용하고 난 후 다시 되돌려주는 메소드

```
public void returnConnection(Connection returnCon){
    Iterator<Connection> iterKeys=hmap.keySet().iterator();
    Connection con=null;
    while(iterKeys.hasNext() ){
        con=iterKeys.next();
        if(con==returnCon){ //con의 주소값이 일치하면
            hmap.put(con, Boolean.FALSE); //쉬는 커백션으로 표시
            break;
        }
    }
}

try{
    removeConnection(); //쉬고있는 커백션 10개를 유지해주는 메소드
}catch(SQLException e){
    e.printStackTrace();
    System.out.println("sqlerror:" + e.getMessage());
}
}
```

//Connection 10개만 유지해주는 메서드

```
public void removeConnection() throws SQLException{
    Connection con=null;
    Iterator<Connection> iterKeys=hmap.keySet().iterator();
    int count=0; //false인 커백션 개수
    while(iterKeys.hasNext() ){
        con=iterKeys.next();
        Boolean b=hmap.get(con);
        boolean b_pre=b.booleanValue();
        if(!b_pre){ //쉬고 있는 커백션 개수 세기 - false인 경우
```

```
count++;  
if(count>10){ //취고 있는 커넥션이 10개가 넘어가면  
    hmap.remove(con); //해시맵에서 삭제
```

```
    con.close();
```

```
}
```

```
}//if
```

```
}//while
```

```
}
```

```
//모든 커넥션 close하는 메서드
```

```
public void closeAll() throws SQLException{
```

```
    Iterator<Connection> iterKeys=hmap.keySet().iterator();
```

```
    Connection con=null;
```

```
    while(iterKeys.hasNext() ){
```

```
        con=iterKeys.next();
```

```
        con.close();
```

```
    }//while
```

```
}
```

```
}//class
```



# 싱글톤 패턴

---



# 싱글톤 패턴

---

- 소프트웨어 설계를 잘 할 수 있도록 만들어 줄 수 있는 방법
  - 소프트웨어 설계 방법론 - 일반화된 것
  - 사례를 통한 공유 - 너무 구체적
- 너무 일반적이지도, 또 너무 구체적이지도 않은 형태로 소프트웨어 설계를 위한 지식이나 노하우를 공유할 수 있는 방법
  - => Erich Gamma 외 3명이 쓴 [Design Patterns] 라는 책에서 제시한 디자인 패턴
  - 객체 지향 개념에 따른 설계들 중 재사용할 경우 유용한 설계들을 디자인 패턴으로 정립
  - 디자인 패턴 - 여러 가지 문제에 대한 설계 사례를 분석해서 서로 비슷한 문제를 해결하기 위한 설계들을 분류하고, **각 문제 유형별로 가장 적합한 설계를 일반화시켜 패턴으로 정립한 것**
- 싱글톤 패턴(Singleton)
  - 인스턴스를 하나만 생성해서 사용하는 패턴
  - 인스턴스의 개수를 하나로 제한하는 패턴



# 기본적인 싱글톤 패턴의 코드

- 인스턴스를 하나만 생성해서 사용하는 패턴

```
public class Singleton{  
    private static Singleton instance;  
  
    private Singleton(){} //private 생성자  
  
    public static Singleton getInstance(){  
        if(instance==null){  
            instance = new Singleton();  
        }  
        return instance;  
    }  
}
```



# 싱글톤 패턴

```
public class Person {  
    /* Person 클래스를 메모리에 한 번만 올리게 막는다  
    [1] 생성자를 private로 => new 연산자를 사용하여, 객체 생성 불가  
    [2] get 메서드 만들기  
        - 멤버변수로 Person을 담을 수 있는 instance 변수를 만든다  
        - static으로 지정해서 한 개만 만들어지게 한다  
        - get 메서드를 만든다  
    */
```

```
    private static Person instance; //static 변수는 중복해서 생성되지 않고, 공유한다는 특성 이용  
    private Person(){ }  
    public static Person getInstance(){  
        if(instance == null)  
            instance = new Person();  
        /*[3] null이면 new로 Person 인스턴스 생성 */  
        return instance;  
    }  
}
```

```
/* Person을 사용 하는 클래스*/  
public class UsePerson {  
    public static void main(String[] args) {  
        Person p = Person.getInstance();  
        System.out.println(p);  
    }  
}
```



```
public class ConnectionPoolMgr{
    private String url,user,pwd;
    private HashMap<Connection, Boolean> hmap;
    private int increment; //증가치
```

```
//static변수
```

```
private static ConnectionPoolMgr instance; //static - 하나만 생성되도록
```

```
//생성자
```

```
private ConnectionPoolMgr(){
    increment=5;//5만큼씩 증가
    hmap=new HashMap<Connection,Boolean>(10);
    Connection con=null;
```

```
try {
```

```
    Class.forName("oracle.jdbc.driver.OracleDriver");
    System.out.println("드라이버 로딩 성공!");
    url="jdbc:oracle:thin:@yang-hp:1521:orcl";
    user="herb";
    pwd="herb";
```

```
//커백션 객체를 미리 생성해 놓기 - 10개
```

```
for(int i=0;i<10;i++){
    con=DriverManager.getConnection(url,user,pwd);
    hmap.put(con, Boolean.FALSE); //해시맵의 key에 커백션 저장
    //해시맵의 value에 true, false 저장, false - 쉬는 커백션이라는 표시
} //for
```

```
System.out.println("ConnectionPool 생성!");
```

멤버의 위치에서 **static** 을 사용해서 객체를 생성하면 이 객체는 객체간의 전역 객체가 됨  
 ⇒이 객체는 단 한번만 생성되고, 객체들 간에 공유됨  
 ⇒**ConnectionPoolMgr**클래스의 객체는 한 번만 생성되고, 이 객체에 접근하는 모든 사용자간에 공유됨

```

        }catch (ClassNotFoundException e) {
            e.printStackTrace();
            System.out.println("Class Not Found!");
        }catch (SQLException e) {
            e.printStackTrace();
            System.out.println("sql 예외발생!");
        }
    }

    }//생성자

    //객체를 생성하는 메서드
    public static ConnectionPoolMgr getInstance(){
        if(instance==null){
            instance = new ConnectionPoolMgr();
        }
        return instance;
    }
}

```

```

public class UpBoardDAO {
    ConnectionPoolMgr pool;

    Connection con;
    PreparedStatement ps;
    ResultSet rs;

    public UpBoardDAO(){
        pool= ConnectionPoolMgr.getInstance();
    }
}

```

UpBoardDAO



# ConnectionPoolMgr 사용

---

- [1] import  
`<%@ page import="com.mystudy.db.ConnectionPoolMgr" %>`
- [2] 커넥션 풀 객체를 이용하여 커넥션 객체 대여 (싱글톤 패턴)  
`ConnectionPoolMgr pool= ConnectionPoolMgr.getInstance();`  
`Connection con= pool.getConnection();`
- [3] 대여한 커넥션 객체 반납  
`pool.returnConnection(con);`



## 답변형 게시판

---



## 답변형(계층형) 게시판

- 데이터들을 계층형으로 만들기 위해서는 하나의 글에 대한 답변과 또 연관된 다른 답변들을 하나의 그룹으로 묶어주어야 함
- 게시판 테이블에 3개의 컬럼 추가

[1] 하나의 글에 연관된 모든 글들을 하나의 그룹으로 묶어주는 컬럼 : groupNo

[2] 글이 몇 번째 단계의 답변인지를 나타내는 컬럼 : step

[3] 글의 정렬 순서를 지정해주는 컬럼 : sortNo

- groupNo - 하나의 글에 연관된 모든 글들을 하나의 그룹으로 묶어주는 컬럼

- step - 글이 몇 번째 단계의 답변인지를 나타내는 컬럼

- sortNo - 글의 정렬 순서를 지정해주는 컬럼

## 자료실

번호	제목	작성자	작성일	조회수
16	☐ 수정하자 <small>NEW</small>	가나다	2015-04-19	10
19	🔗 Re: 수정합니다 <small>NEW</small>	이기수	2015-04-19	2
17	🔗 Re: 수정하자 <small>NEW</small>	이길순	2015-04-19	3
18	🔗 Re: Re: 수정하자 <small>NEW</small>	홍길도	2015-04-19	4
12	☐ 안녕 <small>NEW</small>	김길동	2015-04-19	15

1 [2] [3]

제목 ▼

검색

글쓰기



# 답변형 게시판 테이블

```
create table reboard
```

```
(  
    no                number      primary key,      --번호  
    name              varchar2(20) not null,        --이름  
    pwd               varchar2(20) not null,        --비밀번호  
    title             varchar2(100) not null,        --제목  
    email             varchar2(80) null,            --이메일  
    regdate           date         default sysdate,  --작성일  
    readcount         number      default 0 null,    --조회수  
    content           clob         null,            --내용  
    groupNo           number      default 0,         --그룹번호  
    step              number      default 0,         --글의 단계  
    sortNo            number      default 0,         --글의 정렬순서  
    delFlag           char         default 'N'       --삭제 Flag  
);
```

```
create sequence reboard_seq
```

```
increment by 1
```

```
start with 1
```

```
nocache;
```

# 답변형 게시판

- 하나의 글에 대한 모든 답변들은 같은 GroupNo를 가지게 하여 DB에서 불러올때 GroupNo가 동일한 데이터들을 한 번에 가져온다

	Group	step	sort
새로운 글(120)	120	0	0
답변1	120	1	1

	Group	step	sort
새로운 글(120)	120	0	0
답변1	120	1	1
답변1의 답변	120	2	2
답변2	120	1	2 -> 3



# 답변형 게시판

	Group	step	sort
새로운 글(120)	120	0	0
답변1	120	1	1
답변1의 답변	120	2	2
답변2	120	1	3
답변2의 답변	120	2	▶ 4

	Group	step	sort
새로운 글(120)	120	0	0
답변1	120	1	1
답변1의 답변	120	2	2
3단계 답변	120	3	▶ 3
답변2	120	1	3 -> 4
답변2의 답변	120	2	4 -> 5

# 답변형 게시판

## ■ 규칙

- 새 글이 들어올 때 GroupNo는 자신의 고유값(no)으로 세팅, step은 0, SortNo도 0
- 답변인 글의 GroupNo 는 자신의 상위 글의 GroupNo 와 같은 값을 가짐, step은 자신의 상위글의 step에 1을 더한 값, SortNo 은 자신의 상위 글의 SortNo 에 1을 더한 값
- 단, SortNo 은 자신이 들어가야 할 위치(SortNo)보다 높은 SortNo 의 글이 이미 존재할 경우, SortNo 가 높은 글들은 모두 +1을 해주어야 함

답변으로 들어오는 글은 상위 글의 GroupNo, step, SortNo 값을 참조하여 GroupNo 는 같은 값으로,  
step은 +1 이 된 값으로,  
SortNo 값은 현재 자신이 차지할 SortNo 값(+1 된 값)을 넣어주기 전에 그 SortNo 값보다 큰 SortNo 값은 모두 +1 을 해 준 뒤 자신의 값을 갖게 한다

Update reboard set SortNo = SortNo + 1  
Where GroupNo = 120 and SortNo > 2

← → ↻ localhost:9090/mystudy/reboard/detail.jsp?no=3

## 글 상세보기

제목 안녕하세요

작성자 김길동

등록일 2015-04-19 16:40:14.0

조회수 2

첨부파일  설문.txt(1KB) 다운 : 0

하나  
둘  
셋

```
<div class="center">
  <a href="edit.jsp?no=<%=no%>">수정 </a> |
  <a href="delete.jsp?no=<%=no%>">삭제 </a> |
  <a href="reply.jsp?no=<%=no%>">답변 </a> |
  <a href="list.jsp">목록 </a>
</div>
```

수정 | 삭제 | **답변** | 목록

← → ↻ localhost:9090/mystudy/reboard/reply.jsp?no=16

## 답변하기

제목 Re: 수정하자

작성자

비밀번호

이메일

내용

답변

글목록



# 답변하기 – reply.jsp

---

## ■ 글쓰기(답변하기)

- 새로운 글을 올릴 때는 write.jsp, 어떤 글의 답변을 할 경우에는 reply.jsp 사용
- 새 글
  - 글이 저장될 때 GroupNo는 no로 저장, step, SortNo은 0으로 세팅
- 답변하기
  - 원본글의 no로 GroupNo, step, SortNo를 조회해와서
  - DB에 글을 저장할 때 GroupNo 는 그대로,
  - step와 SortNo는 +1 을 해서 값을 넣어줌
  - detail.jsp에서 넘어온 no값을 가지고 원래 글의 제목과 내용을 DB에서 조회해와서 미리 보여줌.
  - 제목에는 Re :



# 오라클-트랜잭션

---



# 트랜잭션(Transaction) 이란?

- 은행에서 계좌 입,출금(송금)과 같은 개념
  - 은행에서는 송금 자체를 하나의 트랜잭션(거래)으로 보고 A통장에서 출금한 돈이 B통장에 정확히 입금이 확인되면 그 때 거래를 성사시킴(commit)
  - 네트워크 장애로 인해 출금만 발생하고 입금이 되지 않았을 경우에는 이를 모두 취소(rollback)하게 됨
- 복불복 개념 - 모두 처리(Commit)하거나 모두 취소(Rollback)
- 반드시 트랜잭션 처리해야 함
- Savepoint를 설정하여, 설정한 savepoint까지 트랜잭션 관리 가능



# Transaction 관리하기

---

## ■ Transaction

- 논리적인 작업 단위
- 여러 가지 DML 작업들을 하나의 단위로 묶어 둔 것
- 해당 트랜잭션 내에 있는 모든 DML이 성공해야 해당 트랜잭션이 성공하는 것이고 만약 1개의 DML이라도 실패하면 전체가 실패하게 됨
- 모든 트랜잭션은 크기가 다를 수 있음
- 트랜잭션의 시작은 DML이고, 완료하려면 TCL, DCL, DDL이 입력되면 됨

## ■ TCL

- commit – 트랜잭션 내의 작업의 결과를 확정하는 명령어
- Rollback – 트랜잭션 내의 모든 명령어들을 취소하는 명령어





# COMMIT

---

- 메모리 상에서 변경된 내용을 데이터 파일에 반영.
- COMMIT 을 수행하지 않고, 세션이 끝나면 반영 안됨.

- [ 구문 ]

**COMMIT [WORK] [TO SAVEPOINT savepoint\_name];**

- Insert, update, delete 문을 실행하더라도 바로 데이터가 변경되지 않음
- 실제 데이터의 변경은 데이터 파일에 변경사항이 반영될 때 발생하게 됨
- 그 전에는 변경된 데이터들은 오직 오라클 메모리 상에만 존재함
- 최종적으로 데이터 파일에 적용하는 시점 => **COMMIT** 문장을 실행했을 때



## ROLLBACK

---

- 메모리 상에서 변경된 내용을 데이터 파일에 반영하지 않고 종료.
  - 변경된 데이터들을 변경 전 상태로 되돌리는 역할
  - 작업한 내용을 되돌리는 ‘취소’의 개념
- [ 구문 ]  
ROLLBACK [WORK] [TO SAVEPOINT savepoint\_name];
- COMMIT, ROLLBACK 모두 TRANSACTION 처리를 위해 존재.



# jsp에서 트랜잭션 처리

- 트랜잭션(Transaction) – All or Nothing으로 표현
  - 하나의 작업으로 처리되어야 하는 일이 모두 성공적으로 끝나면 commit 되고, 하나라도 문제가 발생하면 rollback 되어서 작업을 수행하기 전 단계로 모든 과정이 회수됨
  - 트랜잭션은 프로그램의 신뢰도를 보장하게 됨
- jsp 에서도 트랜잭션 처리에 대한 메서드를 제공함
  - 트랜잭션을 위한 메서드 : commit(), rollback()
    - JDBC API의 **Connection 객체**는 commit(), rollback() 메서드를 제공
    - commit() – 트랜잭션의 commit을 수행
    - rollback() – 트랜잭션의 rollback을 수행



# 트랜잭션 처리

---

- `setAutoCommit()` 메서드
  - 기본적으로 Connection 객체에 `setAutoCommit(boolean autoCommit)` 메서드가 있음, 기본값이 `true`
- 기본적으로 jsp는 Autocommit
  - 쿼리문이 오토커밋에 의해 자동으로 수행되었던 것
- 트랜잭션을 처리할 때는 오토커밋에 의해 자동으로 `commit`을 사용하면 안됨
  - jsp의 오토커밋이 자동으로 작동되지 못하게 해야 함
  - 오토커밋이 자동으로 작동되지 못하게 하려면 `setAutoCommit(false)` 로 지정해야 함



# 여러 작업을 하나의 트랜잭션으로 묶어서 처리하는 jsp 예제

---

```
try{
    ...
    conn.setAutoCommit(false); //오토커밋이 자동으로 작동되지 못하게 막자

    pstmt.executeUpdate("update ...");
    pstmt.executeUpdate("update ...");
    pstmt.executeUpdate("update ...");
    ...
    conn.commit();
    ...
}catch(SQLException ex){
    if(conn!=null) conn.rollback();
}
conn.setAutoCommit(true);
```



# ReboardVO

```
package com.mystudy.reboard.model;
import java.sql.*;

public class ReBoardVO {
    //멤버변수
    private int no;
    private String name;
    private String pwd;
    private String title;
    private String email;
    private Timestamp regdate;
    private int readcount;
    private String content;

    //답변형 게시판, 자료실에 추가
    private int groupNo;
    private int step;
    private int sortNo;
    private String delFlag;
```

.....

```
public class ReBoardDAO {
    private ConnectionPoolMgr pool;

    //생성자
    public ReBoardDAO(){
        pool = ConnectionPoolMgr.getInstance();
    }

    public int updateReBoard(ReBoardVO vo) throws SQLException{
        Connection con=null;
        PreparedStatement ps=null;
        int n=0;
        try{
            con=pool.getConnection();
            String sql="update reboard set title=?, name=?, email=?, content=?"
                        + " where no=? and pwd=?";

            ps=con.prepareStatement(sql);
            ps.setString(1, vo.getTitle());
            ps.setString(2, vo.getName());
            ps.setString(3, vo.getEmail());
            ps.setString(4, vo.getContent());
            ps.setInt(5, vo.getNo());
            ps.setString(6, vo.getPwd());

            n=ps.executeUpdate();
            System.out.println("게시판 글수정!, n=" + n + ", 입력값 vo:" + vo);
        }finally{
            pool.dbClose(ps, con);
        }
        return n;
    }
}
```

```

public List<ReBoardVO> selectAll(String category,String keyword) throws SQLException{
    Connection con=null;
    PreparedStatement ps=null;
    ResultSet rs=null;
    List<ReBoardVO> alist = new ArrayList<ReBoardVO>();
    try{
        con=pool.getConnection();
        String sql="select no,name, pwd, title,email,regdate, readcount, "
            + "content,groupno, step, sortno, delflag from reboard ";
        //검색의 경우
        if(keyword!=null && !keyword.isEmpty()){
            sql += " where "+category+ " like '%" +keyword+"%";
        }
        sql += " order by groupno desc, sortno";

        ps=con.prepareStatement(sql);

        rs = ps.executeQuery();
        while(rs.next()){
            int no=rs.getInt("no");
            String title=rs.getString("title");
            String name=rs.getString("name");
            String email=rs.getString("email");
            String content=rs.getString("content");
            String pwd=rs.getString("pwd");
            int readcount =rs.getInt("readcount");
            Timestamp regdate=rs.getTimestamp("regdate");

            //답변형 게시판에 추가
            int groupNo=rs.getInt("groupno");
            int step=rs.getInt("step");
            int sortNo=rs.getInt("sortno");
            String delFlag=rs.getString("delflag");

```



```

        ReBoardVO vo = new ReBoardVO();
        vo.setContent(content);
        vo.setEmail(email);
        vo.setName(name);
        vo.setNo(no);
        vo.setPwd(pwd);
        vo.setReadcount(readcount);
        vo.setRegdate(regdate);
        vo.setTitle(title);
        vo.setGroupNo(groupNo);
        vo.setStep(step);
        vo.setSortNo(sortNo);
        vo.setDelFlag(delFlag);

        alist.add(vo);
    }
    System.out.println("게시판 글목록 조회!, alist.size():"
        + alist.size()+", 입력값 category="+category
        +", keyword=" + keyword);
}finally{
    pool.dbClose(rs, ps, con);
}

return alist;
}

```

```

public ReBoardVO selectByNo(int no) throws SQLException{
    Connection con=null;
    PreparedStatement ps=null;
    ResultSet rs=null;

    ReBoardVO vo = new ReBoardVO();
    try{
        con=pool.getConnection();
        String sql="select * from reboard where no=?";
        ps = con.prepareStatement(sql);
        ps.setInt(1, no);

        rs=ps.executeQuery();
        if(rs.next()){
            String title=rs.getString("title");
            String name=rs.getString("name");
            String email=rs.getString("email");
            String content=rs.getString("content");
            String pwd=rs.getString("pwd");
            int readcount =rs.getInt("readcount");
            Timestamp regdate=rs.getTimestamp("regdate");

            //답변형 게시판에 추가
            int groupNo=rs.getInt("groupno");
            int step=rs.getInt("step");
            int sortNo=rs.getInt("sortno");
            String delFlag=rs.getString("delflag");

            vo.setContent(content);
            vo.setEmail(email);
            vo.setName(name);
            vo.setNo(no);
        }
    }
}

```

```

        vo.setPwd(pwd);
        vo.setReadcount(readcount);
        vo.setRegdate(regdate);
        vo.setTitle(title);
        vo.setGroupNo(groupNo);
        vo.setStep(step);
        vo.setSortNo(sortNo);
        vo.setDelFlag(delFlag);
    }
    System.out.println("게시판 상세보기 조회!, vo:"+vo+ ", 입력값 : no=" + no);
}finally{
    pool.dbClose(rs, ps, con);
}
return vo;
}

```

```

public int insertReBoard(ReBoardVO vo) throws SQLException{
    Connection con=null;
    PreparedStatement ps=null;
    int n=0;
    try{
        //[1] 드라이버 로딩, [2] db연결 -con
        con=pool.getConnection();

        //[3] sql 문장 처리 -ps
        String sql="insert into reboard (no, name, pwd, title, email,"
            + " content, groupNo)"
            + " values(reboard_seq.nextval, ?,?,?,?, ?, "
            + "reboard_seq.nextval)";
        ps=con.prepareStatement(sql);
        ps.setString(1, vo.getName());
        ps.setString(2, vo.getPwd());
        ps.setString(3, vo.getTitle());
        ps.setString(4, vo.getEmail());
        ps.setString(5, vo.getContent());

        //[4] 실행
        n=ps.executeUpdate();
        System.out.println("게시판 글쓰기!, n=" + n + ", 입력값 vo:" + vo);
    }finally{
        //[5] 자원 해제
        pool.dbClose(ps, con);
    }
    return n;
}

```

```
public void deleteReboard(int no, int step, int groupNo) throws SQLException{
```

```
//no에 해당하는 삭제 처리
```

```
Connection con=null;
```

```
CallableStatement cs=null;
```

```
try{
```

```
    //[1][2]con
```

```
    con=pool.getConnection();
```

```
    //[3] cs
```

```
    String sql="{call deleteReboard(?, ?, ?)}";
```

```
    cs = con.prepareCall(sql);
```

```
    cs.setInt(1, no);
```

```
    cs.setInt(2, step);
```

```
    cs.setInt(3, groupNo);
```

```
    //[4] exec
```

```
    cs.execute();
```

```
    System.out.println("글 삭제!, 입력값 no="+no
```

```
        +", groupno="+groupNo+", step="+step);
```

```
    }finally{
```

```
        pool.dbClose(cs, con);
```

```
    }
```

```
}
```

**exec deleteReboard(5, 1, 5)**

```

public int reply(ReBoardVO vo) throws SQLException{    //답변달기
    Connection con=null;
    PreparedStatement ps=null;
    int n=0;
    try{
        con=pool.getConnection();
        con.setAutoCommit(false);//자동커밋이 안되게 막는다
        //===트랜잭션 시작===

        //insert하기 전에 sortNo를 위한 자리확보-update
        String sql="update reboard set sortno=sortno+1"
            + " where groupno=? and sortno>?";
        ps=con.prepareStatement(sql);
        ps.setInt(1, vo.getGroupNo());
        ps.setInt(2, vo.getSortNo());
        n= ps.executeUpdate();

        //insert
        sql="insert into reboard (no, name, pwd, title, email,"
            + " content, groupNo, step, sortNo)"
            + " values(reboard_seq.nextval, ?,?,?,?,?,?,?)";
        ps=con.prepareStatement(sql);
        ps.setString(1, vo.getName());
        ps.setString(2, vo.getPwd());
        ps.setString(3, vo.getTitle());
        ps.setString(4, vo.getEmail());
        ps.setString(5, vo.getContent());
        //답변형 게시판관련 번호들
        ps.setInt(6, vo.getGroupNo());
        ps.setInt(7, vo.getStep()+1);
        ps.setInt(8, vo.getSortNo()+1);
    }
}

```

```

n=ps.executeUpdate();
System.out.println("답변형 게시판 글쓰기 성공!, n=" + n
                    + ", 입력값 vo:" + vo);

    con.commit();
    //-----트랜잭션 성공, 종료-----
}catch(SQLException e){
    System.out.println("답변형 게시판 글쓰기 실패!");
    e.printStackTrace();
    try {
        con.rollback(); //트랜잭션 실패
    } catch (SQLException e1) {
        e1.printStackTrace();
    }
}finally{
    try {
        con.setAutoCommit(true);
    } catch (SQLException e) {
        e.printStackTrace();
    }

    pool.dbClose(ps, con);
}
return n;
}

```

```

public int updateReadCount(int no) throws SQLException{
    //no에 해당하는 조회수 증가시키기
    Connection con=null;
    PreparedStatement ps=null;
    int n=0;
    try{
        //[1][2] con
        con=pool.getConnection();

        //[3] ps
        String sql="update reboard set readcount=readcount+1 "
                    +" where no=?";

        ps=con.prepareStatement(sql);
        ps.setInt(1, no);

        //[4] 실행
        n= ps.executeUpdate();
        System.out.println("조회수 증가 , n="+n);
    }finally{
        pool.dbClose(ps, con);
    }

    return n;
}

```



```

public boolean checkPwd(int no, String pwd) throws SQLException{
    //비밀번호 일치여부 체크하기
    Connection con=null;
    PreparedStatement ps=null;
    ResultSet rs=null;
    boolean result=false;
    try{
        con=pool.getConnection();
        String sql="select pwd from reboard where no=?";
        ps=con.prepareStatement(sql);
        ps.setInt(1, no);

        rs = ps.executeQuery();
        if(rs.next()){
            String dbPwd = rs.getString("pwd");
            if(dbPwd.equals(pwd)){
                result=true; //비밀번호가 일치하는 경우
            }
        }
        System.out.println("비밀번호 조회, result="+result
            +", 입력값 no="+no+", pwd="+pwd);
    }finally{
        pool.dbClose(rs, ps, con);
    }
    return result;
}

```

```
public class Utility {  
    public static String cutString(String title, int len){  
        String result="";  
        if(title.length()>len){  
            result=title.substring(0, len) + "...";  
        }else{  
            result=title;  
        }  
  
        return result;  
    }  
  
    public static String displayRe(int step){  
        //답변글인 경우 단계별로 이미지 보여주기  
        String result="";  
        if(step>0){  
            int imgWidth=step*7;  
            result="<img src='../images/blank.gif' border='0' "  
                + " height='7' width='"+imgWidth+ "'>";  
            result+="<img src='../images/re.gif' border='0'>";  
        }  
  
        return result;  
    }  
}
```

```

public static String displayRe2(int step){
    //답변글인 경우 단계별로 이미지 보여주기
    String result="";
    if(step>0){
        for (int i = 0; i < step*2; i++) {
            result+="&nbsp;";
        }

        result+="<img src='../images/re.gif' border='0'>";
    }

    return result;
}

public static String displayNew(Date regdate){
    //24시간 이내의 글인 경우 new이미지 보여주기
    Date today = new Date();
    long gap= (today.getTime() - regdate.getTime())/1000;//초
    gap = gap/(60*60); //시간
    String result="";
    if(gap<24){
        result="<img src='../images/new.gif' border='0'>";
    }

    return result;
}
}

```

reply.jsp

```
<div class="divForm">
<form name="frmReply" method="post" action="reply_ok.jsp">
    <input type="hidden" name="no" value="<%=no%>">
    <!-- 답변형 게시판에 필요한 컬럼 추가 -->
    <input type="hidden" name="groupNo" value="<%=vo.getGroupNo()%>">
    <input type="hidden" name="step" value="<%=vo.getStep()%>">
    <input type="hidden" name="sortNo" value="<%=vo.getSortNo()%>">

    <fieldset>
        <legend>답변하기</legend>
    <div class="firstDiv">
        <label for="title">제목</label>
        <input type="text" id="title" name="title" value="Re: <%=vo.getTitle()%>" />
    </div>
```

reply\_ok.jsp

```
<%
    request.setCharacterEncoding("utf-8");

    String title = request.getParameter("title");
    String name = request.getParameter("name");
    String pwd = request.getParameter("pwd");
    String email = request.getParameter("email");
    String content = request.getParameter("content");

    //답변형에 필요한 no들- hidden 필드
    String groupNo = request.getParameter("groupNo");
    String step = request.getParameter("step");
    String sortNo = request.getParameter("sortNo");
```

```
ReBoardVO vo = new ReBoardVO();
vo.setContent(content);
vo.setEmail(email);
vo.setName(name);
vo.setPwd(pwd);
vo.setTitle(title);
vo.setGroupNo(Integer.parseInt(groupNo));
vo.setStep(Integer.parseInt(step));
vo.setSortNo(Integer.parseInt(sortNo));
```

```
ReBoardDAO dao = new ReBoardDAO();
try{
    int n = dao.reply(vo);
    if(n>0){    %>
        <script>
            alert("답변달기 성공!");
            location.href="list.jsp";
        </script>
    <%        }else{    %>
        <script>
            alert("답변달기 실패!");
            history.back();
        </script>
    <%        }//if
}catch(SQLException e){
    e.printStackTrace();
}
```

%>



# 계층적으로 리스트 보여주기

---

```
select * from Reboard  
order by groupNo desc, sortNo asc
```

## 자료실

번호	제목	작성자	작성일	조회수
16	📎 수정하자 <b>NEW</b>	가나다	2015-04-19	11
19	🔗 Re: 수정합니다 <b>NEW</b>	이기수	2015-04-19	2
17	🔗 Re: 수정하자 <b>NEW</b>	이길순	2015-04-19	3
18	🔗 Re: Re: 수정하자 <b>NEW</b>	홍길도	2015-04-19	4
12	📎 안녕 <b>NEW</b>	김길동	2015-04-19	15

[1](#) [\[2\]](#) [\[3\]](#)

제목 ▼

검색

글쓰기

# list.jsp

## 페이징 처리

```
<%
int currentPage=1; //현재 페이지

//페이지 링크를 눌렀을때 처리
if(request.getParameter("currentPage")!=null){
    currentPage= Integer.parseInt(request.getParameter("currentPage"));
}

//int totalRecord=17; //총 레코드(행) 수
int pageSize=5; //한 페이지에 보여줄 레코드(행) 수
int totalPages=(int)Math.ceil((float)totalRecord/pageSize); //총 페이지 수
int blockSize=10; //한 블록에 보여줄 페이지 수

//[1]시작 [10]끝
int firstPage=currentPage-((currentPage-1)%blockSize); //[1], [11], [21],[31]
int lastPage = firstPage + (blockSize-1); //[10], [20],[30],[40]...

//페이지당 ArrayList에서의 시작 index => 0, 5, 10, 15, 20...
int curPos = (currentPage-1)*pageSize;

//페이지당 글 리스트 시작 번호
int num = totalRecord-(curPos); %>
```

list.jsp?currentPage=2

currentPage-(currentPage-1)  
currentPage-((currentPage-1)%10)

[1][2][3][4][5][6][7][8][9][10]

0 1 2 3 4 5 6 7 8 9

[11][12][13][14][15][16][17][18][19][20]

10 11 12 13 14 15 16 17 18 19

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

17

16

15

14

13

12

11

10

9

8

7

6

5

4

3

2

1



```

<%
    request.setCharacterEncoding("utf-8");

    String category = request.getParameter("category");
    String keyword = request.getParameter("keyword");
    if(keyword==null) keyword="";

    ReBoardDAO dao = new ReBoardDAO();
    List<ReBoardVO> alist =null;
    try{
        alist = dao.selectAll(category, keyword);
    }catch(SQLException e){
        e.printStackTrace();
    }

    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");

    //페이징 처리
    int pageSize=5;
    int blockSize=10;
    PagingVO pageVO = new PagingVO(request, alist, pageSize, blockSize);
%>
<!DOCTYPE HTML><html lang="ko"><head><title>자료실 글 목록 - 허브몰</title>
<meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="../css/mainstyle.css" />
<link rel="stylesheet" type="text/css" href="../css/clear.css" />
<link rel="stylesheet" type="text/css" href="../css/formLayout.css" />
<link rel="stylesheet" type="text/css" href="../css/mystyle.css" />
<style type="text/css">
body{padding:5px;    margin:5px; }
</style>
<script type="text/javascript" src="../jquery/jquery-1.11.2.min.js"></script>

```

```

<script type="text/javascript">
    $(document).ready(function(){
        $(".divList .box2 tbody tr").hover(function(){
            $(this).css("background","#C7E9F3");
        }, function(){
            $(this).css("background","");
        });
    });
</script></head>
<body><h2>자료실</h2>
<%
    if(keyword!=null && !keyword.isEmpty()){ %>
        검색어 : <%=keyword %>, <%=alist.size() %>건 검색되었습니다.

<%}//if %>
<div class="divList">
<table class="box2" summary="기본 게시판에 관한 표로써, 번호, 제목, 작성자, 작성일, 조회수에 대한 정보를 제공">
    <caption>기본 게시판</caption>
    <colgroup> <col style="width:10%;" /><col style="width:50%;" /><col style="width:15%;" />
        <col style="width:15%;" /><col style="width:10%;" /></colgroup>
    <thead>
        <tr><th scope="col">번호</th><th scope="col">제목</th> <th scope="col">작성자</th>
            <th scope="col">작성일</th> <th scope="col">조회수</th> </tr>
    </thead>
    <tbody>
        <!--게시판 내용 반복문 시작 -->
        <% if(alist.isEmpty()){%>
            <tr>
                <td colspan="5" style="text-align:center">해당 글이 존재하지 않습니다.</td>
            </tr>
        <%}else{
            int curPos= pageVO.getCurPos();
            int num=pageVO.getNum();
            for(int i=1;i<=pageSize;i++){
                if(num<1) break;
                ReBoardVO vo = alist.get(curPos++);
                num--; %>

```

```

<tr style="text-align:center">
    <td><%=vo.getNo() %></td>
    <td style="text-align:left">
        <!-- 답변이 있는 원본글이 삭제된 경우 처리-->
        <%= if(vo.getDelFlag().equals("Y")){%>
            <span style="color:silver">삭제된 글입니다</span>
        <%= }else{%>
            <!-- 답변글인 경우, 단계별로 이미지 보여주기 -->
            <%=Utility.displayRe2(vo.getStep()) %>
            <a href="countUpdate.jsp?no=<%=vo.getNo()%>">
            <!-- 제목이 긴 경우 일부만 보여주자 -->
                <%=Utility.cutString(vo.getTitle(), 30) %></a>
            <!-- 24시간 이내의 글인 경우 new 이미지 보여주기 -->
            <%=Utility.displayNew(vo.getRegdate()) %>
        <%= }//if %>
    </td>
    <td>
        <%= String email = vo.getEmail();
        if(email!=null && !email.isEmpty()){%>
            <a href="mailto:<%=vo.getEmail()%>">
                <%=vo.getName() %>
            </a>
        <%=}else{ %>
            <%=vo.getName() %>
        <%=}//if %>
    </td>
    <td><%=sdf.format(vo.getRegdate()) %></td>
    <td><%=vo.getReadcount() %></td>
</tr>
<%= }//for
}//if %> <!--반복처리 끝 -->
</tbody></table> </div>

```

```
<div class="divPage">
    <!-- 페이지 번호 추가 -->
    <%
        if(pageVO.getFirstPage()>1){
            <a href="list.jsp?currentPage=<%=pageVO.getFirstPage()-
1%>&category=<%=category%>&keyword=<%=keyword%>">
                

            </a>

        <%
            }//if %>

        <%
            //[1][2][3][4]...[10]
            //[11][12][13][14]...[20]
            for(int i=pageVO.getFirstPage(); i<=pageVO.getLastPage();i++){
                if(i > pageVO.getTotalPage()) break;
                //현재페이지인 경우 링크걸지 말고, 색상 지정
                if(i==pageVO.getCurrentPage()){%>
                    <span style="color:blue;font-weight:bold"><%=i %></span>

                <%
                    }else{%>
                        <a
href="list.jsp?currentPage=<%=i%>&category=<%=category%>&keyword=<%=keyword%>">
                            [<%=i %>]
                        </a>

                    <%
                        }//if
                    }//for
                %>
                <%if(pageVO.getLastPage()<pageVO.getTotalPage()){ %>
                    <a
href="list.jsp?currentPage=<%=pageVO.getLastPage()+1%>&category=<%=category%>&keyword=<%=keyword%>"
>
                        

                    </a>
                <%}//if %>
            <!-- 페이지 번호 끝 -->
        </div>
```

[1][2][3][4][5][6][7][8][9][10] ▶

◀ [11][12][13][14][15][16][17][18][19][20] ▶

◀ [21][22][23][24][25][26][27][28][29][30] ▶

◀ [31][32][33][34][35][36][37]

```
<div class="divSearch">
    <form name="frmSearch" method="post" action='list.jsp'>
        <select name="category">
            <option value="title" >제목</option>
            <option value="content">내용</option>
            <option value="name">작성자</option>
        </select>
        <input type="text" name="keyword" title="검색어 입력" value="<%=keyword%>">
            <input type="submit" value="검색">
        </form>
    </div>

<div class="divBtn">
    <a href='write.jsp' >글쓰기</a>
</div>

</body>
</html>
```







# 삭제하기-Flag처리

---

- reboard table의 DelFlag 컬럼 처리
  - 1. **답변이 있는 원본글인 경우에는** 레코드를 삭제하지 말고 DelFlag = “Y” 로 update 한다
  - 2. list 페이지에서, DelFlag를 조회해 와서 값이 “Y”이면 제목에 링크 걸지 말고, font color를 회색으로 보여준다.

# 삭제하기-Flag처리

## 자료실

번호	제목	작성자	작성일	조회수
16	삭제된 글입니다	가나다	2015-04-19	12
19	 Re: 수정합니다 NEW	이기수	2015-04-19	3
20	 Re: Re: 수정합니다 NEW	김기수	2015-04-20	0
17	 Re: 수정하자 NEW	이길순	2015-04-19	3
12	 안녕 NEW	김길동	2015-04-19	15

[1](#) [\[2\]](#) [\[3\]](#)

제목 ▼

검색

글쓰기



## detail.jsp

```
<div class="center">
  <a href="edit.jsp?no=<%=no%>">수정</a> |
  <a
href="delete.jsp?no=<%=no%>&step=<%=bean.getStep()%>&groupno=<%=bean.getGroupNo()%>">
삭제</a> |
  <a href="reply.jsp?no=<%=no%>">답변</a> |
  <a href="list.jsp">목록</a>
</div>
```

## delete.jsp

```
<div class="divForm">
  <form name="frmDelete" method="post" action="delete_ok.jsp" >
    <input type="hidden" name="no" value="<%=no%>">
    <input type="hidden" name="step" value="<%=step%>">
    <input type="hidden" name="groupNo" value="<%=groupNo%>">
  </form>
</div>
```



## delete\_ok.jsp

```
<%
    request.setCharacterEncoding("utf-8");
    int no=Integer.parseInt(request.getParameter("no"));
    String pwd=request.getParameter("pwd");
    int step=Integer.parseInt(request.getParameter("step"));
    int groupNo=Integer.parseInt(request.getParameter("groupno"));

    ReBoardDAO dao = new ReBoardDAO();
    int result=0;
    try{
        if(dao.checkPwd(no, pwd)){
            result = dao.deleteByNo(no, step, groupNo);
            if(result>0){
                out.print("<script>");
                out.print("alert('삭제 성공!!');");
                out.print("location.href='list.jsp';");
                out.print("</script>");
            }else{
                out.print("<script>");
                out.print("alert('삭제 실패!!');");
                out.print("history.back()");
                out.print("</script>");
            }
        }else{
            out.print("<script>");
            out.print("alert('비밀번호가 일치하지 않습니다!!');");
            out.print("history.back()");
            out.print("</script>");
        }
    }catch(SQLException e){
        e.printStackTrace();
    }
}%>
```

## deleteReboard 저장 프로시저

```
create or replace procedure deleteReboard
(m_no number,
m_step number,
m_groupno number)
is
cnt number;
begin
  --원본글인 경우
  if m_step = 0 then
    --답변글이 존재하는지 체크
    select count(*) into cnt from reboard
    where groupNo = m_groupno;

    --답변글이 존재하는 경우
    if cnt > 1 then
      update reboard set DelFlag = 'Y'
      where no = m_no;
    else --답변글이 없는 경우
      delete reboard where no=m_no;
    end if;
  else --답변글 자체인 경우
    delete reboard where no=m_no;
  end if;

  commit;
```

```
EXCEPTION WHEN OTHERS THEN
  raise_application_error(-20001, '글 삭제 실패!');
  ROLLBACK;
END;
```

답변이 있는 원본글인 경우에는  
레코드를 삭제하지 말고 DelFlag = “Y” 로 update

**exec deleteReboard(5, 1, 5)**

# 조회수 증가-countUpdate.jsp

```
<%
String no = request.getParameter("no");
if(no==null || no.isEmpty()){
%>
    <script>
        alert("잘못된 url 주소입니다!");
        location.href="list.jsp";
    </script>
<%
    return;
} //if

ReBoardDAO dao = new ReBoardDAO();
try{
    int n = dao.updateReadCount(Integer.parseInt(no));
    if(n>0){
        response.sendRedirect("detail.jsp?no="+no);
    }else{
        <script>
            alert("조회수 증가 실패!!");
            history.back();
        </script>
    } //if
} catch(SQLException e){
    e.printStackTrace();
}%>
```