

Java Web Programming 입문

(Servlet/JSP)

29일차

오늘의 키워드

- ▶ 게시판

게시판

▶ 게시판

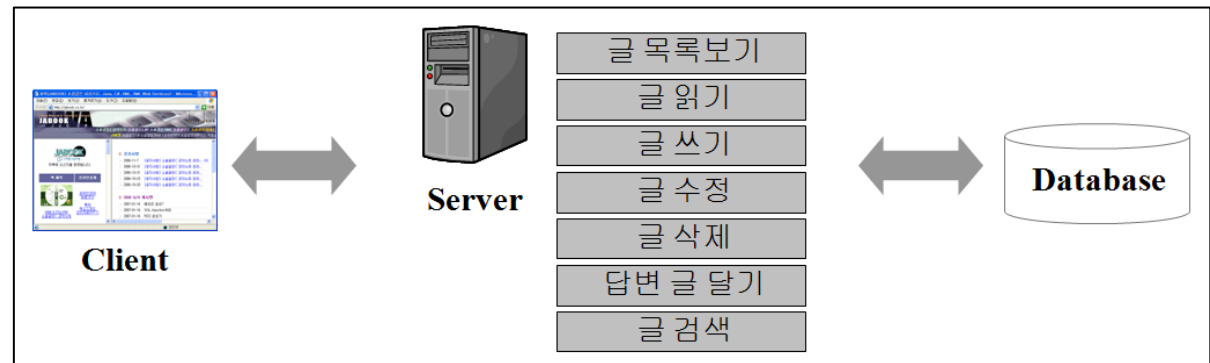
- Internet에서 목록 형태의 정보를 저장 및 표현하기 위한 방법
 - Internet에서 가장 많은 부분을 차지
 - 일반적인 게시판 뿐만 아니라
 - Blog, 쇼핑몰 상품리스트, 방명록, etc...
 - Web Site에서 운영되고 있는 대부분의 기능들은 게시판의 응용

◦ 설계

- Database에 저장된 게시판의 정보를
- 가져오고, 표현하고, 입력하는 등의 과정에 필요한 요소 파악
- 적절히 배치하여 효율적인 정보관리가 될 수 있도록 구성
- 과정
 - 기능 파악
 - 단위 설계
 - 구현

◦ 기능

- 글 목록보기
- 글 읽기
- 글 쓰기
- 글 수정
- 글 삭제
- 검색
- etc...



게시판

▶ 구성

◦ 3-tier Model

- 데이터 계층 (Data Tier)
 - System의 운영에 필요한 Data를 저장하고 내보내는 계층
- 비즈니스 로직 계층 (Business Logic Tier)
 - Program의 기능이 실제 구현된 계층
- 프리젠테이션 계층 (Presentation Tier)
 - 실행 결과에 대한 표현과 같이 Client 사용자와의 Interface를 위한 계층

◦ 3-tier Model 장점

- 동일한 Business Logic을 필요로 하는 Presentation Logic을 다양하게 구현
- Business Logic을 Component 단위로 Module화하여 Web 환경에 적용
- 서로 다른 Database Server를 운영하여도 Code의 수정 없이 확장 가능

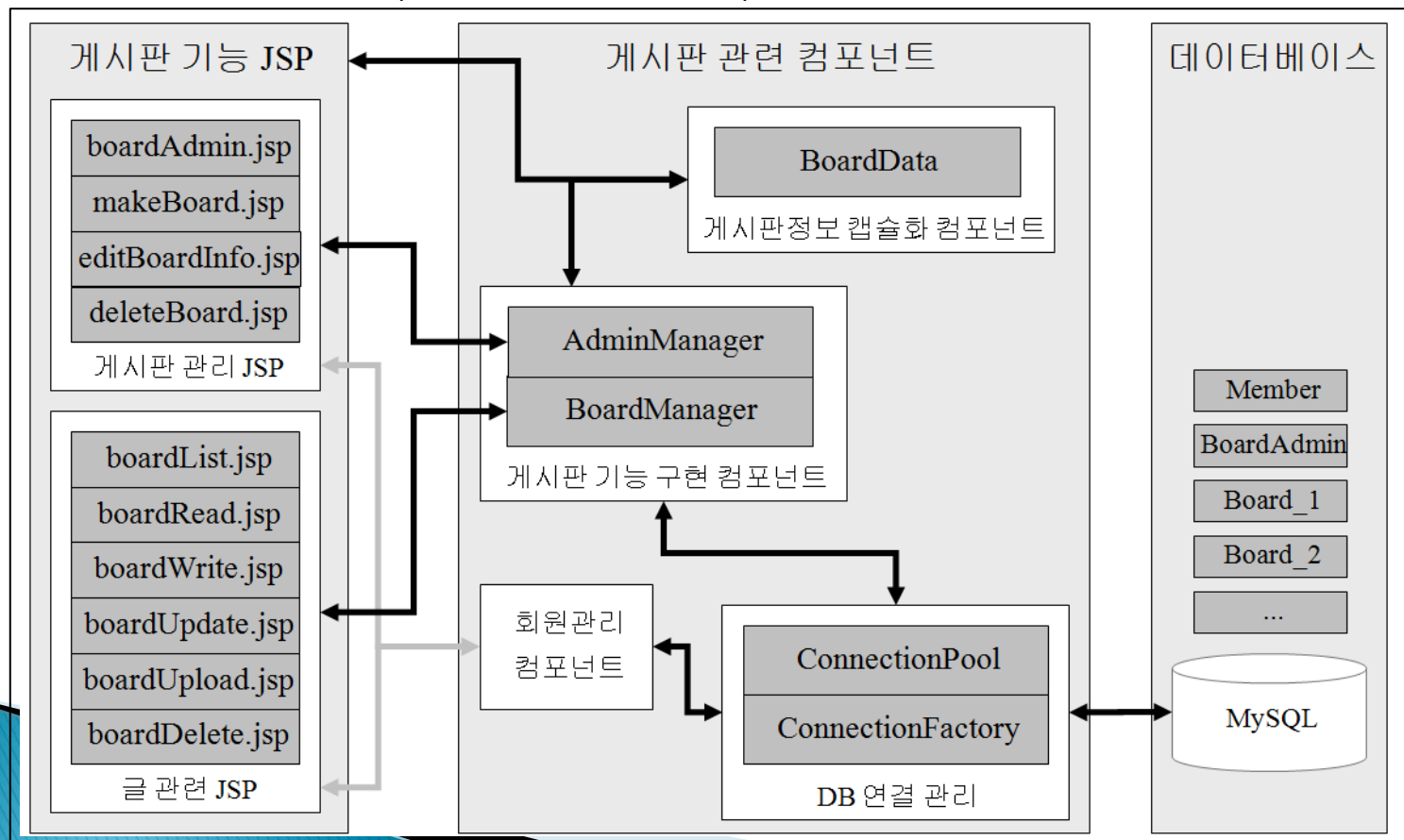
◦ 게시판의 기능 및 특징

- 멀티형 게시판
- 답변형 게시판
- 내부 Logic은 Java Beans 형태로 구현
- Database 연결을 위한 Connection Pool 사용

게시판

▶ 구성

- 게시판의 기능 및 특징
 - 멀티형 게시판
 - 답변형 게시판
 - 내부 Logic은 Java Beans 형태로 구현
 - Database 연결을 위한 Connection Pool 사용



게시판

- ▶ 게시판 관리를 위한 DB Table
 - 멀티형 게시판에서 새로운 게시판을 만든다는 것은
 - Database에 게시판에 해당하는 Table을 새로 생성
 - 게시판의 각 항목에 해당하는 Field를 Database에 만들고
 - 그 Field들의 Type 및 크기 등을 지정
 - 관리자 Table은 게시판의 Table 이름과 제목을 관리

```
CREATE TABLE BOARDADMIN (  
    BOARDNAME    VARCHAR(16) PRIMARY KEY,  
    BOARDSUBJECT VARCHAR(100)  
);
```

필드 명	데이터 타입	설명
boardName	VARCHAR(16)	게시판의 이름
boardSubject	VARCHAR(100)	게시판의 제목(설명)

게시판

▶ 게시판을 위한 DB 테이블

필드 명	데이터 타입	설명
num	INT	글 번호
name	VARCHAR(20)	글 작성자 이름
subject	VARCHAR(100)	글 제목
content	TEXT	글 내용
writeDate	DATETIME	글 작성일
password	VARCHAR(20)	글 비밀번호
count	INT	글 조회 수
ref	INT	원문 글의 참조번호
step	INT	원문 글로부터의 답변 출력 순서
depth	INT	원문 글에 대한 답변 레벨
childCount	INT	답변 글의 수

게시판

▶ 답변 글 구현을 위한 DB 구조

◦ num

- 글 번호 (작성 순서에 따라 순차적으로 할당)
- num 은 글마다 고유의 값을 가짐
 - 일차 Key

◦ ref

- 답변 글이 참조하는 원문 글의 num 값
- 1번 글에 달린 답변 글은 모두 ref 값을 1로 가짐

◦ step

- 답변 글의 출력 순서

◦ depth

- 답변 글의 깊이
- 답변 글에 대한 답변 값

◦ childCount

- 답변 글의 수

num	subject	ref	step	depth	childCount
1	JSP를 잘 하려면?	1	0	0	6
2	↳ [RE] 답변 1	1	1	1	4
3	↳ [RE][RE] 답변 1-1	1	2	2	2
6	↳ [RE][RE][RE] 답변 1-1-1	1	3	3	0
7	↳ [RE][RE][RE] 답변 1-1-2	1	4	3	0
5	↳ [RE][RE] 답변 1-2	1	5	2	0
4	↳ [RE] 답변 2	1	6	1	0

게시판

▶ 답변 글의 구조 자세히 살펴보기

1	num	subject	ref	step	depth	childCount	
	1	JSP를 잘 하려면?	1	0	0	0	← 원문 작성
2	1	JSP를 잘 하려면?	1	0	0	1	
	2	[RE] 답변 1	1	1	1	0	← 1번의 답변
3	1	JSP를 잘 하려면?	1	0	0	2	
	2	[RE] 답변 1	1	1	1	1	
	3	[RE][RE] 답변 1-1	1	2	2	0	← 2번의 답변
4	1	JSP를 잘 하려면?	1	0	0	3	
	2	[RE] 답변 1	1	1	1	1	
	3	[RE][RE] 답변 1-1	1	2	2	0	
	4	[RE] 답변 2	1	3	1	0	← 1번의 답변
5	1	JSP를 잘 하려면?	1	0	0	4	
	2	[RE] 답변 1	1	1	1	2	
	3	[RE][RE] 답변 1-1	1	2	2	0	
	5	[RE][RE] 답변 1-2	1	3	2	0	← 2번의 답변
	4	[RE] 답변 2	1	4	1	0	

게시판

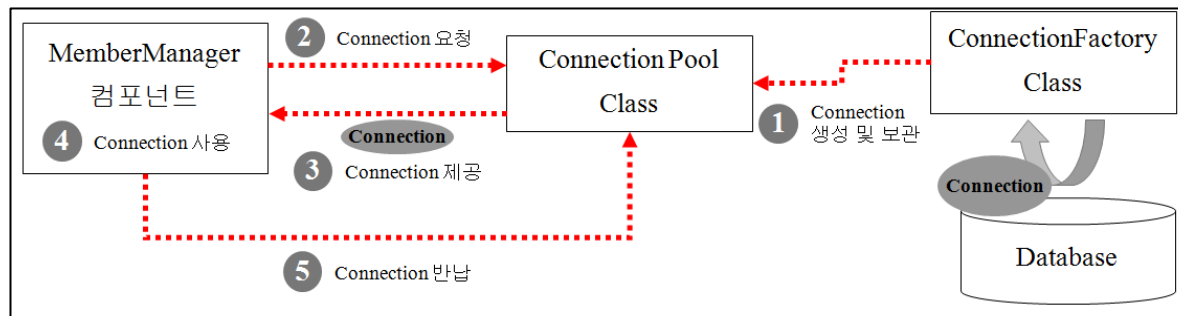
▶ Database 관련 Component

◦ ConnectionFactory Class

- Database 접속을 위한 Connection 생성을 전담하는 Class
- Database 연결에 신경 쓰지 않고
 - Application 자체에만 집중할 수 있는 장점

◦ ConnectionPool Class

- Connection을 생성해서 보관하는 Class
- Connection에 대한 요청이 들어오면
 - 보관 중인 Connection 중 하나를 넘겨주고 일이 끝나면 회수
- 처리 속도 향상과 자원 공유, Connection Object 제어 등의 장점



게시판

▶ 게시판 데이터 캡슐화 Component

◦ BoardData Class

- 게시판 Database로부터 얻어오는 모든 Column에 대응하는 Member Field를 가짐
- Table Column Data를 캡슐화(encapsulation)하여 Object로 관리
- Data를 얻고 설정하기 위한 setter, getter Method 제공
 - Database로부터 값들은 얻어와 Class Field에 저장
 - Object를 통해 Data를 넘겨줌
- 반복적인 Query문을 줄이고 Data 관리와 사용을 용이하게 함

게시판

- ▶ 게시판 데이터 캡슐화 Component
 - BoardData.java

```
import java.io.*;
import java.sql.*;
```

```
public class BoardData implements Serializable {
    private int num;
    private String name;
    private String subject;
    private String content;
    private Date date;
    private String password;
    private int count;
    private int ref;
    private int step;
    private int depth;
    private int childCount;
```

```
    public void setNum(int num) {
        this.num = num;
    }
    public void setName(String name) {
        this.name = name;
    }
    public void setSubject(String subject) {
        this.subject = subject;
    }
    public void setContent(String content) {
        this.content = content;
    }
    public void setDate(Date date) {
        this.date = date;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public void setCount(int count) {
        this.count = count;
    }
    public void setRef(int ref) {
        this.ref = ref;
    }
    public void setStep(int step) {
        this.step = step;
    }
    public void setDepth(int depth) {
        this.depth = depth;
    }
    public void setChildCount(int childCount) {
        this.childCount = childCount;
    }
}
```

```
    public int getNum() {
        return num;
    }
    public String getName() {
        return name;
    }
    public String getSubject() {
        return subject;
    }
    public String getContent() {
        return content;
    }
    public Date getDate() {
        return date;
    }
    public String getPassword() {
        return password;
    }
    public int getCount() {
        return count;
    }
    public int getRef() {
        return ref;
    }
    public int getStep() {
        return step;
    }
    public int getDepth() {
        return depth;
    }
    public int getChildCount() {
        return childCount;
    }
}
```

게시판

▶ 게시판 관리 Component

- 관리자 Login 처리, 새 게시판 생성 수정, 삭제
- 게시판 Table 관리를 위한 Connection Pool 사용
- Query 수행, Query로부터 String/int 등의 값을 가져오기
- 멀티게시판 생성
 - Database Connection, 게시판 Table 생성
- 멀티 게시판 제어
 - Login, 게시판 수정 및 삭제

게시판

- ▶ 게시판 관리 Component
 - AdminManagerSkeletonCode.java

```
import java.util.*; import java.sql.*;
public class AdminManager {
    private static ConnectionPool connectionPool = null;
    private static AdminManager adminManager = null;
    private String id = "testID";
    static { // Connection을 초기화하는 connectionPool을 생성 }
    public AdminManager(){} }
    public static AdminManager getInstance() {
        // AdminManager 객체를 반환하는 메소드
    }
    public boolean existBoard(String boardName) {
        // 해당하는 이름의 게시판이 존재하는지 여부를 검사하기 위한 메소드
    }
    public void makeBoard(String boardName, String boardSubject) {
        // 새로운 게시판을 만들기 위한 메소드
    }
    public boolean isAdmin(String id) {
        // 관리자로 로그인 했는지의 여부를 파악하기 위한 메소드
    }
    public Hashtable getBoardList() throws Exception {
        // 게시판의 리스트를 해쉬테이블로 반환하는 메소드
    }
    public void updateBoard(String boardName, String boardSubject) {
        // 게시판의 제목을 수정하는 메소드
    }
    public void deleteBoard(String boardName) {
        // 게시판 삭제 메소드
    }
    public void adminExecuteUpdate(String sql) {
        // 리턴 값이 없는 SQL 실행 메소드
    }
    public Vector adminExecuteQuery(String sql) {
        // 게시판의 전체 내용을 Vector로 반환하는 메소드
    }
    public int adminExecuteQueryNum(String sql) {
        // 질의문의 결과가 하나의 int형일 경우를 위한 메소드
    }
    public String adminExecuteQueryString(String sql) {
        // 질의문의 결과가 하나의 String형일 경우를 위한 메소드
    }
}
```

게시판

▶ 게시판 관리 Component

◦ SQL 실행을 위한 공통 Method

- return 값이 없는 SQL 실행 Method
 - UPDATE, DELETE 등의 SQL 문 실행 시 사용

```
public void adminExecuteUpdate(String sql) {  
    Connection conn = null;  
    try {  
        conn = connectionPool.getConnection();  
        Statement stmt = conn.createStatement();  
        stmt.executeUpdate(sql);  
        stmt.close();  
    } catch(Exception e) {  
        e.printStackTrace();  
    } finally {  
        connectionPool.releaseConnection(conn);  
    }  
}
```

• return 값이 int 형 하나인 SQL 실행 Method

- 게시판 글의 전체 개수나 검색된 글의 수 등 숫자 값 하나 return 시 사용

```
public int adminExecuteQueryNum(String sql) {  
    ...  
    while(rs.next()) {  
        num = rs.getInt(1);  
    }  
    return num;  
}
```

게시판

▶ 게시판 관리 Component

◦ SQL 실행을 위한 공통 Method

- return 값이 String 형 하나인 SQL 실행 Method

```
public int adminExecuteQueryString(String sql) {  
    ...  
    while(rs.next()) {  
        str = rs.getString(1);  
    }  
    return str;  
}
```

- 게시물의 전체 내용을 Vector로 반환하는 SQL 실행 Method

```
public int adminExecuteQuery(String sql) {  
    Vector v = new Vector();  
    ...  
    while(rs.next()) {  
        BoardData data = new BoardData();  
        data.setNum(rs.getInt(1));  
        data.setName(rs.getString(2));  
        ...  
        v.addElement(data);  
    }  
    return v;  
}
```


게시판

▶ 게시판 관리 Component

◦ 관리자 인증 Method

- Login 하려는 사용자가
 - 게시판 생성 및 수정, 삭제 권한을 가진 사용자인지를 판별하는 method

```
public boolean chkAdmin(String id, String pass) {  
    if( this.id.equals(id) && this.password.equals(pass) ) {  
        return true;  
    } else {  
        return false;  
    }  
}
```

- 관리자에 대한 사항을 Database 또는 Property 파일로 만들어서 사용하거나 회원관리 구현 시 권한에 대한 Field를 반영하는 방법도 가능
- 동일한 이름의 게시판 존재 유무 확인

```
public boolean existBoard(String boardName) {  
    String sql = "select count(boardName) from BoardAdmin";  
    sql += " where boardName='" + boardName + "'";  
    int num = this.adminExecuteQueryNum(sql);  
    if(num == 0){  
        return false;  
    } else {  
        return true;  
    }  
}
```

게시판

▶ 게시판 관리 Component

- 새로운 게시판 생성
 - 게시판의 이름, 제목정보를 삽입한 후 새로운 게시판 Table을 생성

```
public void makeBoard(String boardName, String boardSubject) {  
    // BoardAdmin 테이블 업데이트      this.adminExecuteUpdate(insertBoardSQL);  
    // 새로운 게시판 테이블 생성        this.adminExecuteUpdate(makeBoardSQL);  
}
```

- 게시판 목록 반환
 - BoardAdmin Table로부터
 - 전체 Field 내용을 가져오는 역할 (Hashtable 형태)
 - 게시판 이름은 HashTable Key, 게시판 제목은 HashTable 값

```
public Hashtable getBoardList() throws Exception {  
    ResultSet rs = stmt.executeQuery("select * from BoardAdmin");  
    while(rs.next()) {  
        ht.put(boardName, boardSubject);  
    }  
    return ht;  
}
```

게시판

▶ 게시판 관리 Component

- 게시판 정보 수정 및 삭제
 - 게시판 정보 수정 Method

```
public void updateBoard(String boardName, String boardSubject) {  
    String sql = "update BoardAdmin set boardSubject='" + boardSubject;  
    sql += "' where boardName='" + boardName + "'";  
    this.adminExecuteUpdate(sql);  
}
```

- 게시판 삭제 메소드

```
public void deleteBoard(String boardName) {  
    String deleteRecordSQL = "delete from BoardAdmin";  
    deleteRecordSQL += "where boardName =" + boardName + "'";  
    String dropBoardSQL = "Drop Table " + boardName;  
    this.adminExecuteUpdate(deleteRecordSQL);  
    this.adminExecuteUpdate(dropBoardSQL);  
}
```

게시판

- ▶ 게시판 관리 Component
 - AdminManager.java (시작)

```
package org.jabook.noveljsp.board;
import java.util.*;
import java.sql.*;
import org.jabook.noveljsp.sql.*;
public class AdminManager {
    private static ConnectionPool connectionPool = null;
    private static AdminManager adminManager = null;
    private String id = "testId";
    private String password = "testPass";
    static {
        try{
            connectionPool = connectionPool.getConnectionPool();
            adminManager = adminManager.getInstance();
        } catch(Exception e){
            System.out.println("Error01:static block error!!");
        }
    }

    public AdminManager(){ }
    public static AdminManager getInstance() {
        if (adminManager == null) {
            adminManager = new AdminManager();
        }
        return adminManager;
    }
}
```

게시판

- ▶ 게시판 관리 Component
 - AdminManager.java (계속)

```
// 해당하는 이름의 게시판이 존재하는지 여부를 검사하기 위한 메소드
public boolean existBoard(String boardName) {
    String sql = "select count(boardName) from BoardAdmin where boardName='" + boardName + "'";
    int num = this.adminExecuteQueryNum(sql);
    if(num == 0){
        return false;
    } else {
        return true;
    }
}

// 새로운 게시판을 만들기 위한 메소드
public void makeBoard(String boardName, String boardSubject) {
    String insertBoardSQL = "insert BoardAdmin values('" + boardName + "', '" + boardSubject + "')";
    String makeBoardSQL = "create table " + boardName + " (" ;
    makeBoardSQL += "num int NOT NULL PRIMARY KEY,";
    makeBoardSQL += "name varchar(20) NOT NULL,";
    makeBoardSQL += "subject varchar(100) NOT NULL,";
    makeBoardSQL += "content text NULL,";
    makeBoardSQL += "writeDate datetime,";
    makeBoardSQL += "password varchar(20) NOT NULL,";
    makeBoardSQL += "count int NOT NULL,";
    makeBoardSQL += "ref int NOT NULL,";
    makeBoardSQL += "step int NOT NULL,";
    makeBoardSQL += "depth int NOT NULL,";
    makeBoardSQL += "childCount int NOT NULL";
    makeBoardSQL += ")";
    this.adminExecuteUpdate(insertBoardSQL);
    this.adminExecuteUpdate(makeBoardSQL);
}
```

게시판

▶ 게시판 관리 Component

◦ AdminManager.java (계속)

```
// 관리자로 로그인 했는지의 여부를 파악하기 위한 메소드
public boolean chkAdmin(String id, String pass) {
    if(this.id.equals(id) && this.password.equals(pass)) {
        return true;
    }else{
        return false;
    }
}

public boolean isAdmin(String id) {
    if(id.equals(this.id)){
        return true;
    }else{
        return false;
    }
}

// 게시판의 리스트를 해쉬테이블로 반환하는 메소드
public Hashtable getBoardList() throws Exception {
    Hashtable ht = new Hashtable();
    Connection conn = null;
    try{
        conn = connectionPool.getConnection();
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery("select * from BoardAdmin");
        while(rs.next()) {
            String boardName = rs.getString("boardName");
            String boardSubject = rs.getString("boardSubject");
            ht.put(boardName, boardSubject);
        }
        rs.close();
        stmt.close();
    }catch(Exception e){
        e.printStackTrace();
    } finally {
        connectionPool.releaseConnection(conn);
    }
    return ht;
}
```

게시판

- ▶ 게시판 관리 Component
 - AdminManager.java (계속)

```
// 게시판의 제목을 수정하는 메소드
public void updateBoard(String boardName, String boardSubject) {
    String sql = "update BoardAdmin set boardSubject='" + boardSubject + "' where boardName='" + boardName + "'";
    this.adminExecuteUpdate(sql);
}

// 게시판 삭제 메소드
public void deleteBoard(String boardName) {
    String deleteRecordsSQL = "delete from BoardAdmin where boardName ='" + boardName + "'";
    String dropBoardSQL = "Drop Table " + boardName;
    this.adminExecuteUpdate(deleteRecordsSQL);
    this.adminExecuteUpdate(dropBoardSQL);
}

// 리턴 값이 없는 SQL 실행 메소드
public void adminExecuteUpdate(String sql) {
    Connection conn = null;
    try {
        conn = connectionPool.getConnection();
        Statement stmt = conn.createStatement();
        stmt.executeUpdate(sql);
        stmt.close();
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        connectionPool.releaseConnection(conn);
    }
}
```

게시판

▶ 게시판 관리 Component

◦ AdminManager.java (끝)

```
// 게시판의 전체 내용을 Vector로 반환하는 메소드
public Vector adminExecuteQuery(String sql) {
    Vector v = new Vector();
    Connection conn = null;
    try{
        conn = connectionPool.getConnection();
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(sql);
        while(rs.next()) {
            BoardData data = new BoardData();
            data.setNum(rs.getInt(1));
            data.setName(rs.getString(2));
            data.setSubject(rs.getString(3));
            data.setContent(rs.getString(4));
            data.setDate(rs.getDate(5));
            data.setPassword(rs.getString(6));
            data.setCount(rs.getInt(7));
            data.setRef(rs.getInt(8));
            data.setStep(rs.getInt(9));
            data.setDepth(rs.getInt(10));
            data.setChildCount(rs.getInt(11));
            v.addElement(data);
        }
        rs.close();
        stmt.close();
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        connectionPool.releaseConnection(conn);
    }
    return v;
}
```

```
// 질의문의 결과가 하나의 int형일 경우를 위한 메소드
public int adminExecuteQueryNum(String sql) {
    int num = 0;
    Connection conn = null;
    try{
        conn = connectionPool.getConnection();
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(sql);
        while(rs.next()) {
            num = rs.getInt(1);
        }
        rs.close();
        stmt.close();
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        connectionPool.releaseConnection(conn);
    }
    return num;
}
```

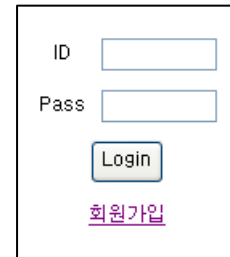
```
// 질의문의 결과가 하나의 String형일 경우를 위한 메소드
public String adminExecuteQueryString(String sql) {
    String str = null;
    Connection conn = null;
    try{
        conn = connectionPool.getConnection();
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(sql);
        while(rs.next()) {
            str = rs.getString(1);
        }
        rs.close();
        stmt.close();
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        connectionPool.releaseConnection(conn);
    }
    return str;
}
```


게시판

▶ 관리자 확인

◦ 관리자

- 전체 게시판 관리 기능
- ID, Password 의 철저한 관리 필요
- 관리자 Login 시 게시판 목록 열람
- 게시판을 만들거나, 제목 수정, 삭제 가능



A login form with two input fields labeled 'ID' and 'Pass', a 'Login' button, and a link labeled '회원가입' (Sign Up) below the button.

```
AdminManager adminManager = AdminManager.getInstance();  
if ( session.getAttribute("id") != null  
    && adminManager.isAdmin( (String) session.getAttribute("id")) )  
    // 관리자 페이지 출력  
else  
    // 로그인 페이지 출력
```

- isAdmin() Method를 이용하여
 - 현재 저장된 id session의 값이 관리자의 ID와 동일한지 비교
 - 동일하다면 게시판의 관리 Page 호출, 아니면 Login Page 호출

◦ 관리자 Login을 위한 ID/Password 검사

```
if(adminManager.chkAdmin(adminID, adminPass)) {  
    // ID 값으로 세션을 설정  
} else {  
    // 로그인 페이지 출력  
}
```

게시판

- ▶ 관리자 확인
 - boardAdmin_AdminLogin.jsp

```
<%@page contentType="text/html; charset=euc-kr"%>
<%@page import="java.util.*"%>
<html><head><title>Admin Page</title></head>
<body>
    <%
        AdminManager adminManager = AdminManager.getInstance();
        if ( session.getAttribute("id") != null
            && adminManager.isAdmin((String)session.getAttribute("id"))) {    %>
            <h2>관리자로 로그인되어 있습니다.</h2>
            <a href="logout.jsp">로그 아웃</a>

    <%
        } else {
            String adminID = request.getParameter("adminid");
            String adminPass = request.getParameter("adminpass");
            if(adminManager.chkAdmin(adminID, adminPass)) {
                session.setAttribute("id", adminID);
                response.sendRedirect("boardAdmin_AdminLogin.jsp");

            } else {    %>
                <form action="boardAdmin_AdminLogin.jsp" method="post">
                    ID : <input type="text" name="adminid" /><br>
                    PASS : <input type="password" name="adminpass" /><br>
                    <input type="submit" value="로그인" />
                </form>

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

게시판

- ▶ Logout
 - logout.jsp

```
<%@page contentType="text/html; charset=euc-kr"%>
<html>
  <head>
    <title>Logout Page</title>
  </head>
  <body>
    <%
      session.invalidate();
    %>
    로그아웃 되었습니다.
  </body>
</html>
```

게시판

▶ 게시판 생성

```
<form action="makeBoard.jsp" method="post" name="create">  
  <input type="text" name="boardName" size="20">  
  <textarea name="boardSubject" cols="35" rows="3"></textarea>  
  <input type="submit" value="게시판 생성">  
</form>
```

- 게시판 이름과 게시판 제목 입력
- 두 개의 Text 상자를 가진 입력 양식 작성

* 게시판 생성

게시판 이름	게시판 설명
<input type="text" value="freeboard"/>	<input type="text" value="누구나 쓸 수 있는 자유 게시판"/>

게시판

- ▶ 게시판 생성
 - boardAdmin_CreateBoard.jsp

```
<%@page contentType="text/html; charset=euc-kr"%>
<%@page import="java.util.*"%>
<html>
  <head>
    <title>Admin Page</title>
  </head>
  <body>
    <%
      AdminManager adminManager = AdminManager.getInstance();
      if ( session.getAttribute("id") != null
          && adminManager.isAdmin( (String)session.getAttribute("id")) ) {

    %>
        <b>* 게시판 생성</b>
        <form action="makeBoard.jsp" method="post" name="create">
          게시판 이름 : <input type="text" name="boardName" size="20"><br>
          게시판 설명 : <textarea name="boardSubject" cols="35" rows="3"></textarea><br>
          <input type="submit" value="게시판 생성">
        </form>

    %>
      } else {
        out.print("관리자 권한이 없습니다.");
      }
    %>
  </body>
</html>
```

게시판

▶ 게시판 생성 처리 JSP

- 이전 Page 로부터 전달된 게시판 이름, 제목을 추출
 - Parameter 값 추출

```
String boardName = request.getParameter("boardName");  
String boardSubject = request.getParameter("boardSubject");
```

- 새로운 게시판 생성

```
if(!adminManager.existBoard(boardName)) {  
    adminManager.makeBoard(boardName, boardSubject);  
} else {  
    out.print("동일한 이름의 게시판이 존재합니다.");  
}
```

게시판

- ▶ 게시판 생성 처리 JSP
 - makeBoard.jsp

```
<%@page contentType="text/html; charset=euc-kr"%>
<%@page import="org.jabook.noveljsp.board.*"%>
<html>
  <head>
    <title>Make New Board</title>
  </head>
  <body>
    <%
      String boardName = request.getParameter("boardName");
      String boardSubject = request.getParameter("boardSubject");
      AdminManager adminManager = AdminManager.getInstance();
      if(boardName.equals("")) {
        %>
          게시판 이름은 필수 항목입니다.<br>
        <%
          } else {
            if(!adminManager.existBoard(boardName)) {
              adminManager.makeBoard(boardName, boardSubject);
            } else {
              %>
                동일한 이름의 게시판이 존재합니다.<br>
                다른 이름을 입력하세요.<br>
              <%
            }
          }
        %>
      </body>
</html>
```

게시판

▶ 게시판 목록 출력

- 게시판 목록을 출력해주는 getBoardList() Method는
 - Hashtable 반환

```
AdminManager adminManager = AdminManager.getInstance();  
Hashtable ht = adminManager.getBoardList();  
Enumeration e = ht.keys();  
while(e.hasMoreElements()) {  
    String key = (String)e.nextElement();  
    String value = (String)ht.get(key);  
}
```

* 게시판 목록

freeboard	누구나 쓸 수 있는 자유 게시판	보기 수정 삭제
-----------	-------------------	--

게시판

▶ 게시판 목록 출력

◦ boardAdmin_BoardList.jsp

```
<%@page contentType="text/html; charset=euc-kr"%>
<%@page import="java.util.*"%>
<html><head><title>Admin Page</title></head>
<body>
    <%
        AdminManager adminManager = AdminManager.getInstance();
        if ( session.getAttribute("id") != null
            && adminManager.isAdmin((String)session.getAttribute("id"))) {
    %>

        <b>* 게시판 목록</b>

    <%

        Hashtable ht = adminManager.getBoardList();
        Enumeration e = ht.keys();
        while(e.hasMoreElements()) {
            String key = (String)e.nextElement();
            String value = (String)ht.get(key);

    %>

            <form action="editBoardInfo.jsp" method="post" name="edit">
                <input type="text" name="boardName" size="20" value="<%=key%>"
                    readonly="readonly"><br>
                <textarea name="boardSubject" cols="35" rows="3"><%=value%></textarea><br>
                <a href="boardList.jsp?boardName=<%=key%>" target="_BLANK">보기</a> |
                <a href="javascript:document.edit.submit()">수정</a> |
                <a href="deleteBoard.jsp?boardName=<%=key%>">삭제</a>

            </form>

    <%
        }
    } else {
        out.print("관리자 권한이 없습니다.");
    }
    %>
</body></html>
```

게시판

▶ 게시판 수정 및 삭제

◦ 게시판 제목 수정

```
String boardName = request.getParameter("boardName");  
String boardSubject = request.getParameter("boardSubject");  
adminManager.updateBoard(boardName, boardSubject);
```

◦ editBoardInfo.jsp

```
<%@page contentType="text/html; charset=euc-kr"%>  
<html><head><title>Edit Board</title></head>  
<body>  
    <%  
        String boardName = request.getParameter("boardName");  
        String boardSubject = request.getParameter("boardSubject");  
        AdminManager adminManager = AdminManager.getInstance();  
        adminManager.updateBoard(boardName, boardSubject);  
    %>  
    <%=boardName%> 게시판이 수정되었습니다.  
</body></html>
```

게시판

- ▶ 게시판 수정 및 삭제
 - 게시판 삭제 JSP

```
String boardName = request.getParameter("boardName");  
adminManager.deleteBoard(boardName);
```

- deleteBoard.jsp

```
<%@page contentType="text/html; charset=euc-kr"%>  
<html><head><title>Delete Board</title></head><body>  
    <%  
        String boardName = request.getParameter("boardName");  
        AdminManager adminManager = AdminManager.getInstance();  
        adminManager.deleteBoard(boardName);  
    %>  
    <%=boardName%> 게시판이 삭제되었습니다.  
</body></html>
```

* 게시판 목록	
<input type="text" value="freeboard"/>	누구나 쓸 수 있는 자유 게시판
	보기 수정 삭제

게시판

▶ boardAdmin.jsp (시작)

```
<%@page contentType="text/html; charset=euc-kr"%>
<%@page import="java.util.*"%>
<html>
  <head>
    <title>Admin Page</title>
  </head>
  <body>
    <%
      AdminManager adminManager = AdminManager.getInstance();
      if ( session.getAttribute("id") != null
          && adminManager.isAdmin( (String)session.getAttribute("id"))) ) {

        <h3>게시판 관리</h3>
        <a href="logout.jsp">로그 아웃</a>
        <p><b>* 게시판 목록</b></p>

        Hashtable ht = adminManager.getBoardList();
        Enumeration e = ht.keys();
        while(e.hasMoreElements()) {
          String key = (String)e.nextElement();
          String value = (String)ht.get(key);

          <form action="editBoardInfo.jsp" method="post" name="edit">
            <input type="text" name="boardName" size="20" value="<%=key%>"
              readonly="readonly">
            <a href="boardList.jsp?boardName=<%=key%>" target="_BLANK">보기</a> |
            <a href="javascript:document.edit.submit()">수정</a> |
            <a href="deleteBoard.jsp?boardName=<%=key%>">삭제</a><br>
            <textarea name="boardSubject" cols="35" rows="3"><%=value%></textarea><br>
          </form>

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

게시판 관리

* 게시판 목록

freeboard	누구나 쓸 수 있는 자유 게시판	보기 수정 삭제
-----------	-------------------	--

* 게시판 생성

게시판 이름	게시판 설명	<input type="button" value="게시판 생성"/>
<input type="text"/>	<input type="text"/>	

게시판

▶ boardAdmin.jsp (끝)

```
<p><b>* 게시판 생성</b></p>
<form action="makeBoard.jsp" method="post" name="create">
    게시판 이름 : <input type="text" name="boardName" size="20"><br>
    게시판 설명 : <textarea name="boardSubject" cols="35" rows="3"></textarea><br>
    <input type="submit" value="게시판 생성">
</form>

<%
} else {
    String adminID = request.getParameter("adminid");
    String adminPass = request.getParameter("adminpass");
    if(adminManager.chkAdmin(adminID, adminPass)) {
        session.setAttribute("id", adminID);
        response.sendRedirect("boardAdmin.jsp");
    } else {
        <form action="boardAdmin.jsp" method="post">
            ID : <input type="text" name="adminid" /><br>
            PASS : <input type="password" name="adminpass" /><br>
            <input type="submit" value="로그인" />
        </form>
    }
}

%>

</body>
</html>
```

오늘 숙제

- ▶ 집에서 해보자!
- ▶ 예제파일 분석
 - ▶ 드디어 분석 Season!