
JSP 화면 설계 및 구현

게시판 사이트
WeBoard

권 우 리

2020. 05. 25

목차

1. 프로젝트 개요	3
2. 프로젝트 설명	3
①. 기능 정의서	3
②. 사용 환경	4
3. 화면 설계	4
①. UI 스타일 가이드	4
②. UI 흐름도	4
③. 프로세스 정의	5
④. 화면 설계도	5
4. 데이터베이스	9
5. 기능 설계	10
①. 주요 기능 구현 및 상세 내용	10
②. 파일 구조	36
③. 개선사항	36
6. 배포 환경	37
Contact Me wewednfl@gmail.com	37

프로젝트 개요

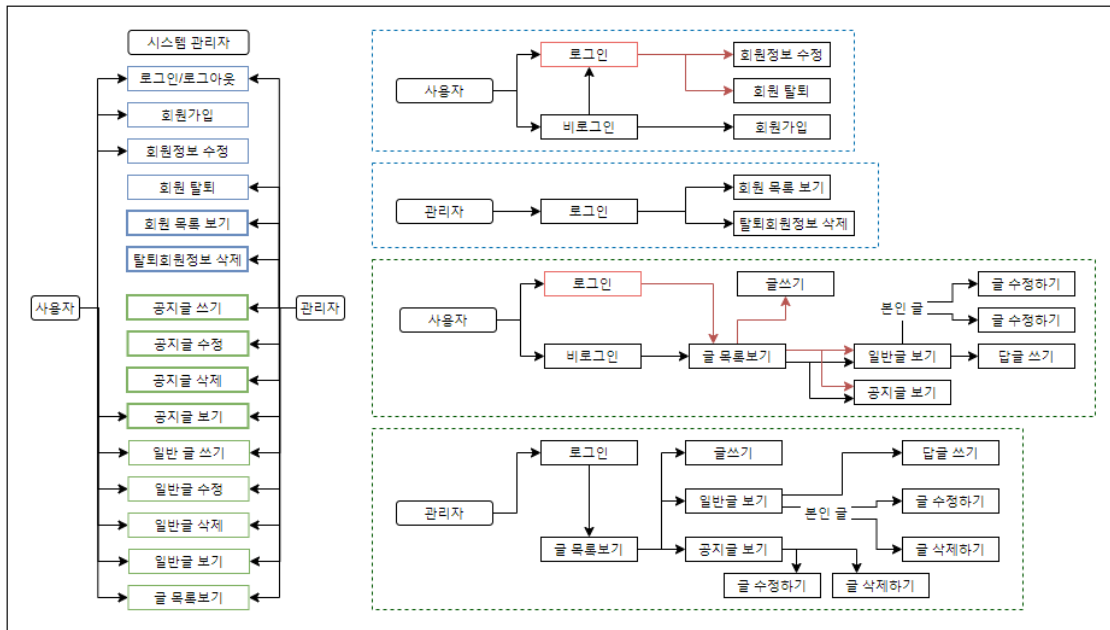
1. Jsp, Oracle, Tomcat을 활용한 회원제 게시판 사이트이다.
2. 비 로그인 시에는 게시글 목록 및 내용 보기만 가능하다
3. 글쓰기 및 기능은 로그인 시에만 제공되며, 작성된 글은 작성자만 수정 및 삭제 가능하다.
4. 탈퇴 후 한 달 동안은 회원정보가 유지되며, 탈퇴 전 삭제하지 않은 게시글은 남는다.

프로젝트 설명

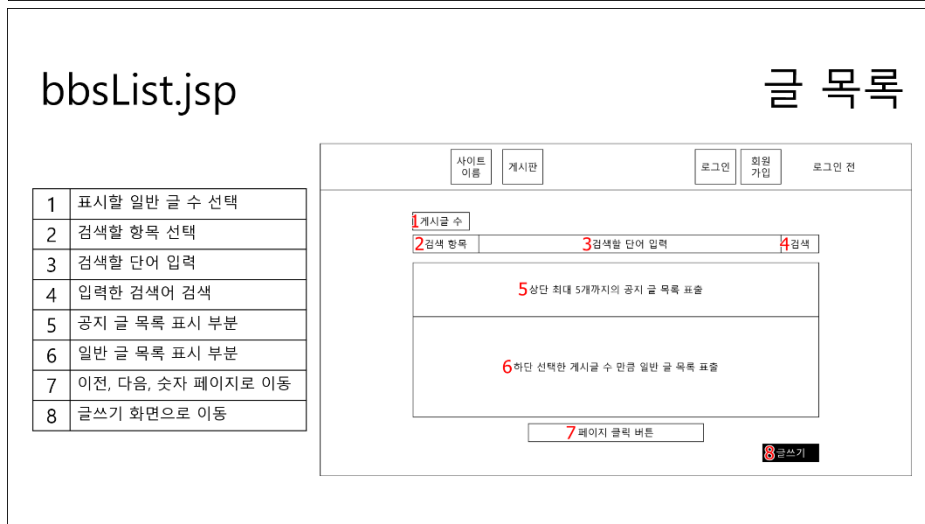
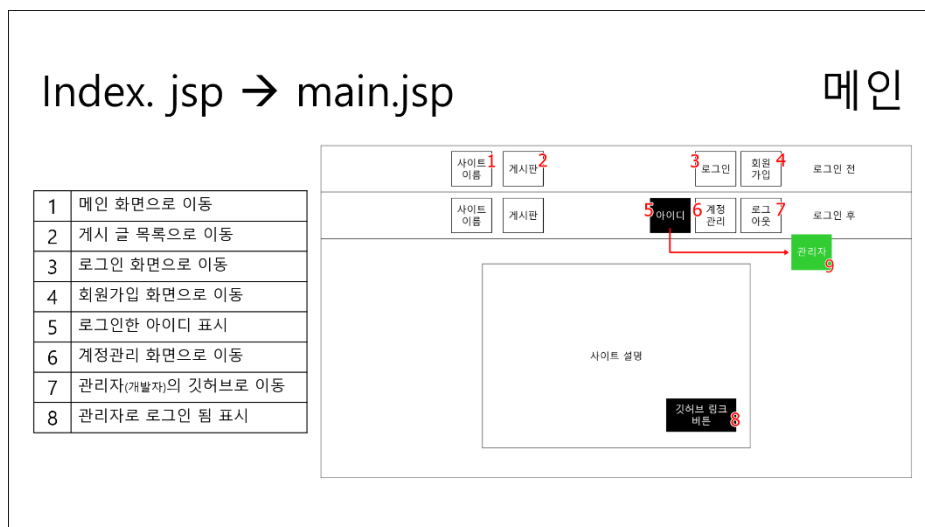
1. 기능 정의

u. 회원 관리		
번호	기능	설명
W-U-1	로그인	아이디와 비밀번호를 입력 받아 회원 테이블에서 일치 여부를 확인.
W-U-2	로그아웃	버튼 클릭 시 별도 확인작업 없이 바로 로그아웃.
W-U-3	회원가입	아이디, 비밀번호, 이름, 성별, 이메일 주소 기입 후 회원가입 가능.
W-U-4	정보 수정	비밀번호 확인 후 진입 가능. 아이디 외 모든 정보 수정 가능.
W-U-5	회원 탈퇴	비밀번호 확인 후 탈퇴처리. 정보는 한 달 후 관리자에 의해 삭제.
W-U-6	회원 확인	회원정보 수정, 회원 탈퇴 전 비밀번호를 재 확인.
W-U-7	회원 목록	관리자 계정으로만 진입 가능. 회원들의 비밀번호 외 모든 정보를 출력. 아이디와 이메일 주소 검색 가능. 회원 정보 10개씩 페이지 처리.
W-U-8	회원 삭제	회원 목록 페이지에 버튼으로 위치. 탈퇴한지 한 달 이상 된 회원정보 삭제.
b. 게시 글 관리		
번호	기능	설명
W-B-1	글 작성	로그인 후 글쓰기 가능. 답글 쓰기 가능. 파일 업로드 가능. 관리자 계정으로 진입 시 최대 5개까지 공지 글 작성 가능.
W-B-2	글 조회	로그인을 하지 않아도 글 보기 가능. 하단에 답글 쓰기 버튼 표시. 로그인 후 해당 계정으로 작성한 글을 볼 경우 수정, 삭제버튼 표시. 업로드 된 파일이 있을 경우 해당 파일 명 클릭 시 다운로드 가능.
W-B-3	글 수정	동일한 계정으로 쓴 글만 수정 가능. 글 보기 화면에서 버튼클릭 후 진입
W-B-4	글 삭제	동일한 계정으로 쓴 글만 삭제 가능. 글 보기 화면에서 버튼클릭 후 진입
W-B-5	글 목록	로그인을 하지 않아도 글 목록 보기 가능. 하단에 글쓰기 버튼 표시. 공지 글은 최 상단에 굵은 글씨로 어느 페이지에서든 출력. 한 페이지당 출력할 글의 수는 사용자가 선택할 수도 있도록 함. 제목, 내용, 작성자 검색 가능. 작성자 클릭 시 작성자의 전체 글 보기와 작성자에게 이메일 보내기 가능

3. 프로세스 정의



4. 화면 설계도



bbsWriteForm.jsp // 사용자

글 작성

1	게시 글 제목 입력
2	게시 글 내용 입력
3	업로드 할 파일 선택
4	게시 글 목록으로 이동
5	1, 2 입력 초기화
6	게시글 저장

사이트 이름

게시판

아이디

계정 관리

로그 아웃

로그인 후

글쓰기 or 글 수정하기

1 글 제목

2 글 내용

파일 선택 3

찾기

4 목록

5 다시쓰기

6 저장하기

bbsWriteForm.jsp // 관리자

글 작성

1	게시 글 제목 입력
2	게시 글 내용 입력
3	업로드 할 파일 선택
4	게시 글 목록으로 이동
5	1, 2 입력 초기화
6	게시글 저장
7	저장할 글의 유형 선택

사이트 이름

게시판

관리자

계정 관리

로그 아웃

로그인 후

글쓰기 or 글 수정하기

1 글 제목

7 공지글 / 일반글 선택

2 글 내용

파일 선택 3

찾기

4 목록

5 다시쓰기

6 저장하기

bbsViewForm.jsp

글 조회

1	게시 글의 번호 (아이디) 표시
2	게시 글의 조회수 표시
3	게시 글의 제목 표시
4	게시 글의 작성자 아이디 표시
5	게시 글의 작성일 표시
6	게시 글의 내용 표시
7	게시 글에 저장된 파일 표시
8	게시 글 목록으로 이동
9	본인 글일 경우 해당 글 수정
10	본인 글일 경우 해당 글 삭제
11	해당 글의 답글로 글쓰기 이동

사이트 이름

게시판

로그인

회원 가입

로그인 전

글 내용보기

1 글 번호 2 조회수

3 글 제목

4 작성자

5 작성일

6 글 내용

7 파일

8 목록

본인 글에만 보임 9 수정 10 삭제

답글쓰기 11

usrLoginForm.jsp

로그인

1	아이디 입력
2	비밀번호 입력
3	1, 2 입력값 확인 및 로그인

[사이트 이름](#)
[게시판](#)
[로그인](#)
[회원 가입](#)
[로그인 전](#)

로그인

1 아이디
2 비밀번호
3 로그인

usrJoinForm.jsp

회원 가입 및 정보 수정

1	아이디 입력
2	비밀번호 입력
3	재확인할 비밀번호 입력
4	이름 입력
5	성별 입력
6	이메일 주소 입력
7	회원가입 입력

[사이트 이름](#)
[게시판](#)
[로그인](#)
[회원 가입](#)
[로그인 전](#)

회원가입 or 회원 정보 수정

1 아이디
2 비밀번호
3 비밀번호 확인
4 이름
5 성별
6 이메일 주소
7 회원가입

usrCheckForm.jsp

비밀번호 확인

1	아이디 입력
2	비밀번호 입력
3	1, 2 입력값 확인

[사이트 이름](#)
[게시판](#)
[아이디](#)
[계정 관리](#)
[로그 아웃](#)
[로그인 후](#)

비밀번호 확인

1 아이디
2 비밀번호
3 확인

usrMain.jsp // 사용자

계정 관리

사이트 이름

게시판

아이디

계정 관리

로그아웃

로그인 후

1 (아이디)에 로그인 아이디 표시

2 수정 전 비밀번호 확인 화면 이동

3 탈퇴 전 비밀번호 확인 화면 이동

4 로그아웃되는 버튼

1 (아이디) 님의 계정 관리

2 회원정보 수정

3 회원 탈퇴

4 로그아웃

usrMain.jsp // 관리자

계정 관리

사이트 이름

게시판

관리자

계정 관리

로그아웃

로그인 후

1 관리자 전용 화면임 표시

2 회원 목록 화면으로 이동

3 로그아웃되는 버튼

1 관리자용 계정 관리

2 회원 정보 관리

3 로그아웃

usrList.jsp

회원 목록

사이트 이름

게시판

관리자

계정 관리

로그아웃

로그인 후

1 검색할 항목 선택

2 검색할 단어 입력

3 검색

4 가입된 회원의 정보 목록(비밀번호 제외)
탈퇴한 회원 최 상단 위치. 페이지당 10개씩 보임

5 페이지 클릭 버튼

6 10일 전 탈퇴 회원 정보 삭제

논리모델링 중 성별과 이름은 필요하지 않은 필드라 판단되어 요구사항을 변경하게 됨.

W-u-j	회원가입	아이디, 비밀번호, 이름, 성별, 이메일 주소 기입 후 회원가입 가능.
W-u-j	회원가입	아이디, 비밀번호, 이메일 주소 기입 후 회원가입 가능.

데이터베이스

5. 테이블 구조

A. 회원 관리 테이블 – Usr

필드 명	데이터 타입	설명
usrId	VARCHAR(20)	PRIMARY KEY 사용자의 아이디를 저장. 아이디 중복 불가
usrPasswd	VARCHAR2(20)	NOT NULL 사용자의 비밀번호를 저장.
usrDelete	NUMBER(2)	DEFAULT 0 사용자의 탈퇴여부를 저장. 0은 사용, 1은 탈퇴.
delDate	TIMESTAMP	NOT NULL 사용자의 가입 또는 탈퇴 시점을 저장

B. 글 관리 테이블 – Bbs

필드 명	데이터 타입	설명
bbsId	NUMBER(20)	PRIMARY KEY 글의 번호를 저장
usrId	VARCHAR2(15)	NOT NULL 글의 작성자 아이디를 저장 탈퇴해도 작성한 글은 남기 때문에 외래키 지정 X
bbsTitle	VARCHAR2(75)	NOT NULL 글의 제목을 저장
bbsDate	TIMESTAMP	NOT NULL 글의 작성 날짜를 저장
readCount	NUMBER	DEFAULT 0 글의 조회수를 저장. 작성시에는 조회수 0.
ref	NUMBER(20)	NOT NULL 답 글이 있을 경우 그룹화 하기 위한 번호 저장
re_step	NUMBER(10)	NOT NULL 제목 글과 답변 글의 순서를 저장
re_level	NUMBER(10)	NOT NULL 글의 레벨을 저장
bbsContent	VARCHAR2(3000)	NOT NULL 글의 내용을 저장
notice	number(2)	NOT NULL 글의 공지 여부를 저장. 0은 일반, 1은 공지
fileName	VARCHAR2(75)	 첨부파일의 이름을 저장

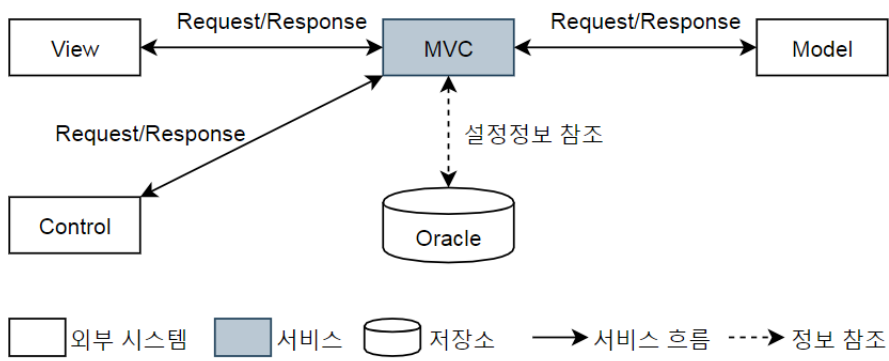
6. ER-Diagram

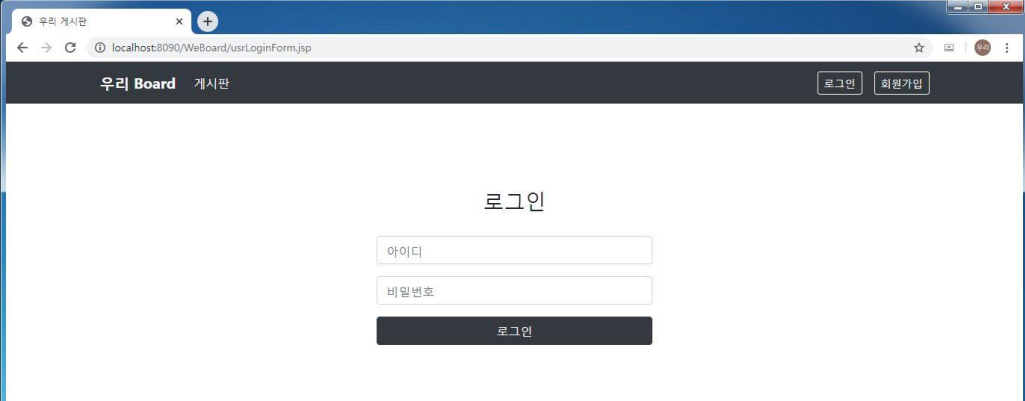
SCOTT.BBS	
P * BBSID	NUMBER (20)
* USRID	VARCHAR2 (15 BYTE)
* BBSTITLE	VARCHAR2 (75 BYTE)
* BBSDATE	TIMESTAMP
READCOUNT	NUMBER
* REF	NUMBER (20)
* RE_STEP	NUMBER (10)
* RE_LEVEL	NUMBER (10)
* BBSCONTENT	VARCHAR2 (3000 BYTE)
NOTICE	NUMBER (2)
FILENAME	VARCHAR2 (75 BYTE)
BBS_PK (BBSID)	


SCOTT.USR	
P * USRID	VARCHAR2 (20 BYTE)
USRPASSWD	VARCHAR2 (20 BYTE)
USRNAME	VARCHAR2 (20 BYTE)
USRGENDER	VARCHAR2 (20 BYTE)
USREMAIL	VARCHAR2 (20 BYTE)
USRDELETE	NUMBER (38)
DELDATE	TIMESTAMP
USR_PK (USRID)	

기능 설계 및 상세 내용

1. 인터페이스 정의서

환경	개발환경	서비스그룹	화면 처리
서비스 명	MVC		
주요 매커니즘	 <p> 외부 시스템 서비스 저장소 → 서비스 흐름 --- 정보 참조 </p> <p>MVC 서비스는 개발프레임워크 설정 정보에서 MVC 서비스에 대한 설정 정보를 참조하여 Model, View, Control 간의 연결을 가운데서 조정하는 역할을 수행한다.</p>		
환경	개발환경	서비스그룹	화면 처리
W-U-1	로그인		
기능	<p>비 로그인 시에만 진입이 가능하다.</p> <p>Form에서 아이디와 비밀번호를 입력한 뒤 로그인 버튼 또는 엔터키를 누르면 Pro에서 DB의 Usr테이블에 접속하여 일치 여부를 확인한다.</p> <p>일치할 경우 세션에 아이디 정보를 저장한다.</p> <p>일치하는 아이디 또는 비밀번호가 없을 경우, 탈퇴한 계정일 경우, DB 연결에 실패할 경우 팝업으로 알린다.</p>		
usrLogin Form.jsp	<pre> if(session.getAttribute("usrId")!=null) { usrId = (String) session.getAttribute("usrId"); } if(usrId!=null) { PrintWriter script = response.getWriter(); script.println("<script>"); script.println("alert('이미 로그인 되어 있습니다.');""); script.println("location.href='index.jsp'"); script.println("</script>"); } </pre> <p>세션에 저장된 값이 없을 경우에만 로그인 창에 진입이 가능하므로 위와 같이 저장된 값이 null이 아닐 경우 usrId에 저장된 값을 대입하고, 이미 로그인 되어 있다는 팝업을 띄운 후 index(첫 화면)로 이동한다.</p> <p>그 외에는 화면을 띄워 정보를 입력할 수 있도록 한다.</p>		
usrDAO. java	<pre> //로그인 public int login(String usrId, String usrPasswd) { Connection conn = null; </pre>		

	<pre> PreparedStatement pstmt = null; ResultSet rs = null; String sql = "select usrDelete from Usr where usrId = ? and usrPasswd = ?"; try { conn = getConnection(); pstmt = conn.prepareStatement(sql); pstmt.setString(1, usrId); pstmt.setString(2, usrPasswd); rs = pstmt.executeQuery(); if (rs.next()) { if (rs.getInt("usrDelete")==0) { return 1; //로그인 성공 } else if (rs.getInt("usrDelete")==1) { return 2; //탈퇴계정 } } else return 0; //로그인 실패 } catch (Exception e) { e.printStackTrace(); } finally { closeDBResources(rs, pstmt, conn); } return -1; //DB오류 } </pre> <p>입력한 아이디와 비밀번호 매개변수로 받아와서 동시에 일치하는 값이 있는지 select 문으로 확인한다.</p> <p>일치하는 값이 있으면 탈퇴여부를 확인 후 사용계정이면 0, 탈퇴계정이면 2, 일치하는 값이 없으면 0을 반환. 그 외 DB오류일 경우 -1을 반환한다.</p>
usrLigin Pro.jsp	<pre> UsrDAO usrDAO = UsrDAO.getInstance(); int result = usrDAO.login(usr.getUsrId(), usr.getUsrPasswd()); if (result==1){ session.setAttribute("usrId", usr.getUsrId()); PrintWriter script = response.getWriter(); script.println("<script>"); script.println("location.href='index.jsp'"); script.println("</script>"); } </pre> <p>java코드에서 가져온 결과값을 result에 대입한다.</p> <p>값이1일 경우 세션에 아이디를 저장한 후 첫 화면으로 이동한다.</p> <p>그 외에는 각각의 값에 따른 팝업 안내 후 다시 입력할 수 있도록 이전 화면으로 돌아간다.</p>
화면 구현	

환경	개발환경	서비스그룹	화면 처리
W-U-2	로그아웃		
기능	<p>로그인 시에만 진입 가능하다.</p> <p>별도의 폼이나 확인절차 없이 버튼 클릭 시 바로 세션에 저장된 정보를 삭제한다.</p>		
usrLogout Pro.jsp	<pre>session.removeAttribute("usrId"); PrintWriter script = response.getWriter(); script.println("<script>"); script.println("alert('로그아웃 되었습니다.');""); script.println("location.href='mainVr1.jsp'"); script.println("</script>");</pre> <p>로그아웃은 별도의 입력 값 확인 절차가 필요하지 않기 때문에 로그아웃 버튼 클릭 시 Pro로 바로 이동하여 저장되어 있는 세션을 삭제하고 팝업으로 알린다.</p>		
화면 구현			

환경	개발환경	서비스그룹	화면 처리
W-U-3	회원가입		
기능	<p>비 로그인 시 진입 가능하다.</p> <p>폼에서 모든 값을 입력한 후 회원가입 버튼 또는 엔터키를 누르면 Pro에서 DB의 Ustr 테이블에 접속하여 아이디의 중복 여부를 확인. 중복이 없을 경우 회원정보를 저장한다.</p> <p>입력하는 값에 빈 칸이 없어야 하며, 비밀번호와 재확인 비밀번호가 같아야 한다.</p> <p>만약 중복되는 아이디 값이 있을 경우 팝업으로 알린다.</p>		
usrJoin Form.jsp	<pre>if(session.getAttribute("usrId")!=null) { usrId = (String) session.getAttribute("usrId"); } if(usrId!=null) { PrintWriter script = response.getWriter(); script.println("<script>"); script.println("alert('이미 로그인 되어 있습니다.');""); script.println("location.href='index.jsp'"); script.println("</script>"); }</pre> <p>세션에 저장된 값이 없을 경우에만 회원가입 창에 진입이 가능하므로 위와 같이 저장된 값이 null이 아닐 경우 usrId에 저장된 값을 대입하고, 이미 로그인 되어 있다는 팝</p>		

	<p>업을 띄운 후 index(첫 화면)로 이동한다.</p> <p>그 외에는 화면을 띄워 정보를 입력할 수 있도록 한다.</p>
usrDAO. java	<pre>//회원가입 public int join(String usrId, String usrPasswd, String usrEmail) { Connection conn = null; PreparedStatement pstmt = null; ResultSet rs = null; String sql = "select usrEmail from Usr where usrId = ?"; try { conn = getConnection(); pstmt = conn.prepareStatement(sql); pstmt.setString(1, usrId); rs = pstmt.executeQuery(); if (rs.next()) { return 0; //아이디 중복 오류 } else { closeDBResources(pstmt); sql = "insert into Usr (usrId, usrPasswd, usrEmail, delDate) " + "values (?, ?, ?, ?)"; pstmt = conn.prepareStatement(sql); pstmt.setString(1, usrId); pstmt.setString(2, usrPasswd); pstmt.setString(3, usrEmail); pstmt.setTimestamp(4, new Timestamp(System.currentTimeMillis())); return pstmt.executeUpdate(); //회원가입 요청 } } catch (Exception e) { e.printStackTrace(); } finally { closeDBResources(rs, pstmt, conn); } return -1; //DB오류 }</pre> <p>입력한 정보들을 매개변수로 받아와서 아이디 중복여부 먼저 확인한 뒤 중복되는 아이디가 있을 경우 0을 반환, 없을 경우 입력한 회원정보들을 저장. DB오류일 경우 -1을 반환한다.</p>
usrJoin Pro.jsp	<pre>UsrDAO usrDAO = UsrDAO.getInstance(); int result = usrDAO.join(usr.getUsrId(), usr.getUsrPasswd(), usr.getUsrEmail());</pre> <p>Form에서 입력되지 않은 항목이 있거나 두 번 입력한 비밀번호가 서로 일치하지 않을 경우 이전화면으로 돌아간다.</p> <p>그 외의 경우 위 코드와 같이 usrDAO에서 가져온 결과값을 result에 대입한다.</p> <p>DB에 입력한 값이 저장되었을 경우 세션에 아이디를 저장한 후 첫 화면으로 이동한다. 그 외에는 각각의 값에 따른 팝업 안내 후 다시 입력할 수 있도록 이전 화면으로 돌아간다.</p>
화면 구현	(다음 페이지)

환경	개발환경	서비스그룹	화면 처리
W-U-4	회원 비밀번호 확인		
기능	<p>회원정보 수정 또는 회원 탈퇴 전에 사용자 확인을 위해 거쳐가는 화면이다.</p> <p>Form으로 이동시 로그인 되어있는 아이디 값은 입력창에 자동으로 입력된다.</p> <p>비밀번호 입력 후 확인 버튼 또는 엔터키를 누르면 Pro에서 DB의 Usr테이블에 접속하여 아이디에 대한 비밀번호가 올바른지 확인 후 수정 화면으로 이동 또는 탈퇴 처리한다.</p>		
usrCheck Form.jsp	<pre> if (session.getAttribute("usrId")!=null){ usrId = (String) session.getAttribute("usrId"); } else { PrintWriter script = response.getWriter(); script.println("<script>"); script.println("alert('로그인 후 이용 가능합니다.');""); script.println("location.href='usrLoginForm.jsp'"); script.println("</script>"); } </pre> <p>회원정보 수정과 탈퇴 모두 로그인 상태에서 가능하므로 저장된 세션의 정보 확인 후 null일 경우 로그인 화면으로 이동한다.</p> <p>그 외에는 usrId에 저장된 값을 대입한다.</p> <pre>String where = request.getParameter("where");</pre> <p>이전 화면에서 클릭한 버튼의 값을 가져와 그에 맞는 비밀번호 확인 화면을 띄운다.</p>		

usrDAO. java	<pre>//사용자 비밀번호 확인 public int checkPwd(String usrId, String usrPasswd) { Connection conn = null; PreparedStatement pstmt = null; ResultSet rs = null; String SQL = "select usrEmail from Usr where usrId = ? and usrPasswd = ?"; try { conn = getConnection(); pstmt = conn.prepareStatement(SQL); pstmt.setString(1, usrId); pstmt.setString(2, usrPasswd); rs = pstmt.executeQuery(); if (rs.next()) { return 1; //확인됨 } else return 0; //확인 안됨 } catch (Exception e) { e.printStackTrace(); } finally { closeDBResources(rs, pstmt, conn); } return -1; //DB오류 }</pre> <p>아이디와 비밀번호를 매개변수로 받아와서 일치하는 정보가 있는지 확인한다. 있으면 1, 없으면 0, DB오류일 경우 -1을 반환한다.</p>
usrCheck Pro.jsp	<pre>String usrId = request.getParameter("usrId"); String usrPasswd = request.getParameter("usrPasswd"); String where = request.getParameter("where"); int check = usrDAO.checkPwd(usrId, usrPasswd);</pre> <p>아이디, 비밀번호, 클릭한 버튼의 값(where)을 가져온 후 usrDAO에서 가져온 결과값을 check에 대입한다.</p> <p>Check의 값이 1이면 where의 값에 따라 아래와 같이 탈퇴처리. 또는 회원정보 수정 화면으로 이동한다.</p> <pre>if (where.equals("del")){ //탈퇴버튼 거쳐옴 int result = usrDAO.delete(usrId); PrintWriter script = response.getWriter(); script.println("<script>"); script.println("alert('탈퇴요청이 접수되었습니다.');""); script.println("location.href='usrLogoutPro.jsp'"); script.println("</script>"); } else if (where.equals("up")){ //수정버튼 거쳐옴 PrintWriter script = response.getWriter(); script.println("<script>"); script.println("location.href='usrUpdateForm.jsp'"); script.println("</script>"); }</pre> <p>그 외에는 각각의 값에 따른 팝업 안내 후 다시 입력할 수 있도록 이전 화면으로 돌아간다.</p>
화면 구현	회원 정보 수정 클릭 시 (다음 페이지)

우리 게시판

weewewe

계정관리

로그아웃

수정 전 비밀번호 확인

weewewe

비밀번호

확인하기

회원 탈퇴 클릭 시

우리 게시판

weewewe

계정관리

로그아웃

탈퇴 전 비밀번호 확인

weewewe

비밀번호

확인하기

탈퇴하더라도 게시글은 유지됩니다.

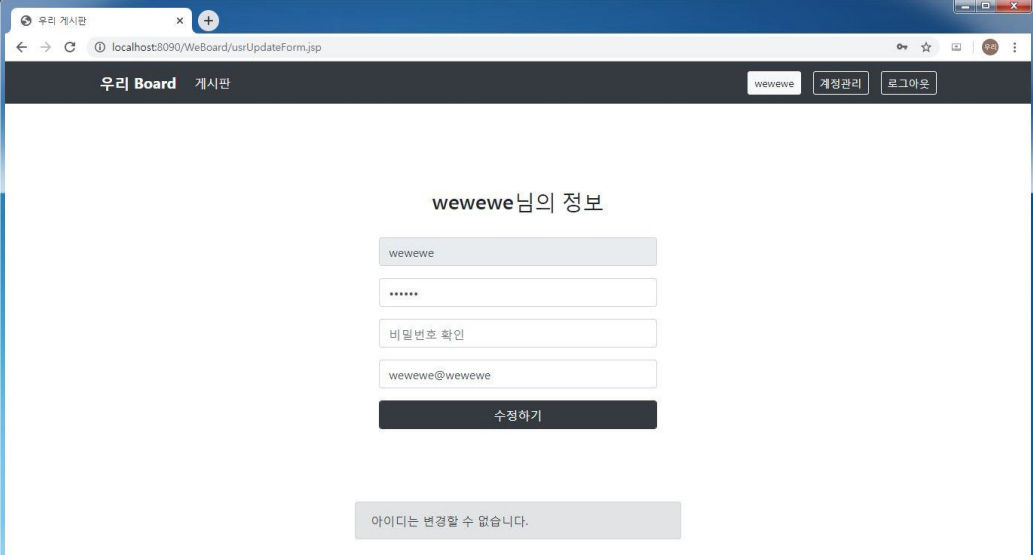
원치 않으시면 탈퇴 전에 삭제해 주세요.

탈퇴요청된 계정은 30일 뒤 최종 삭제됩니다.

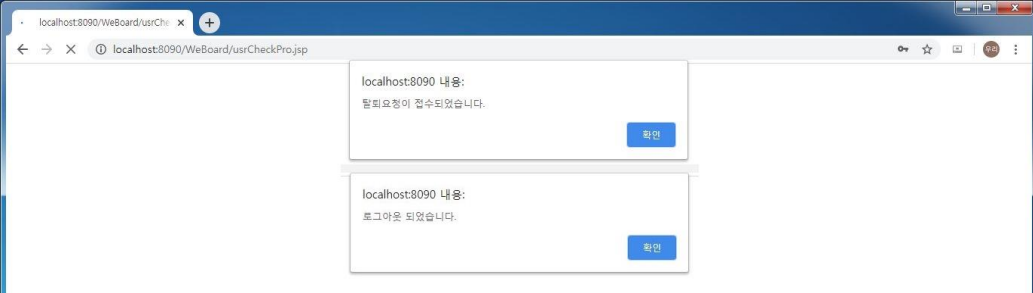
환경	개발환경	서비스그룹	화면 처리
W-U-5	회원정보 수정		
기능	<p>로그인 후 usrCheckPro.jsp를 거쳐서 진입 가능하다.</p> <p>회원정보 수정 시 아이디는 수정 불가하다.</p> <p>각 입력 화면에 기존에 저장된 비밀번호를 제외한 회원의 정보가 자동으로 입력된다.</p> <p>모든 정보가 입력된 후 정보 수정 버튼 또는 엔터키를 누르면 Pro에서 DB의 Usr테이블에 접속하여 아이디에 대한 각 정보들을 저장한다.</p> <p>입력하는 값에 빈 칸이 없어야 하며, 비밀번호와 재확인 비밀번호가 같아야 한다.</p>		
usrUpdate Form.jsp	<pre> if (session.getAttribute("usrId")!=null){ usrId = (String) session.getAttribute("usrId"); } else { PrintWriter script = response.getWriter(); script.println("<script>"); script.println("alert('로그인 후 수정 가능합니다.');""); script.println("location.href='usrLoginForm.jsp'""); script.println("</script>"); } </pre>		

	<p>계정정보 수정은 로그인 후 수정이 가능하다.</p> <p>저장된 세션이 있는지 확인 후 없을 경우 팝업으로 알린 후 로그인 폼으로 이동한다.</p> <p>그 외의 경우 화면을 띄워 정보를 입력할 수 있도록 한다.</p> <p>화면을 띄울 때 기존 정보를 불러와야 하므로 아래의 코드를 이용한다.</p> <pre> UsrDAO usrDAO = UsrDAO.getInstance(); Usr usr = usrDAO.getUsr(usrId); </pre>
usrDAO. java	<pre> //회원정보 읽기 public Usr getUsr(String usrId) { Connection conn = null; PreparedStatement pstmt = null; ResultSet rs = null; String sql = "select * from Usr where usrId = ?"; try { conn = getConnection(); pstmt = conn.prepareStatement(sql); pstmt.setString(1, usrId); rs = pstmt.executeQuery(); if (rs.next()) { Usr usr = new Usr(); usr.setUsrId(rs.getString("usrId")); usr.setUsrPasswd(rs.getString("usrPasswd")); usr.setUsrEmail(rs.getString("usrEmail")); return usr; } } catch (Exception e) { e.printStackTrace(); } finally { closeDBResources(rs, pstmt, conn); } return null; } </pre> <p>Form에 회원의 기존에 저장되어 있던 정보를 불러오는 메소드이다.</p> <p>세션에 저장된 아이디를 매개변수로 받아 그 아이디에 대한 다른 정보들을 저장한다.</p> <p>DB 오류일 경우 null값을 반환한다.</p> <pre> //회원정보 수정하기 public int update(Usr usr) { Connection conn = null; PreparedStatement pstmt = null; String sql = "update Usr set usrPasswd = ?, usrEmail = ? where usrId = ?"; try { conn = getConnection(); pstmt = conn.prepareStatement(sql); pstmt.setString(1, usr.getUsrPasswd()); pstmt.setString(2, usr.getUsrEmail()); pstmt.setString(3, usr.getUsrId()); return pstmt.executeUpdate(); } catch (Exception e) { e.printStackTrace(); } finally { closeDBResources(pstmt, conn); } return -1; //DB오류 } </pre> <p>입력된 정보들을 받아와서 각각의 값을 수정하는 메소드. DB오류일 경우 -1을 반환한다.</p>

usrUpdate Pro.jsp	<pre>String usrPasswd = request.getParameter("usrPasswd"); String checkedPwd = request.getParameter("checkedPwd"); String usrName = request.getParameter("usrName"); String usrGender = request.getParameter("usrGender"); String usrEmail = request.getParameter("usrEmail");</pre> <p>위와 같이 Form에서 입력한 정보를 먼저 받아온다.</p> <p>받아온 정보 중 입력되지 않은 항목이 있거나 두 번 입력한 비밀번호가 서로 일치하지 않을 경우 이전화면으로 돌아간다.</p> <p>그 외의 경우 아래와 같이 usrDAO에서 가져온 결과값을 result에 대입한다.</p> <pre>usr.setUsrId(usrId); usr.setUsrPasswd(usrPasswd); usr.setUsrEmail(usrEmail); int result = usrDAO.update(usr);</pre> <p>DB에 입력한 값이 저장되었을 경우 세션에 아이디를 저장한 후 첫 화면으로 이동한다.</p> <p>그 외에는 각각의 값에 따른 팝업 안내 후 다시 입력할 수 있도록 이전 화면으로 돌아간다.</p>
----------------------	--

화면 구현	
-------	---

환경	개발환경	서비스그룹	화면 처리
W-U-6	회원 탈퇴		
기능	<p>별도의 jsp파일 없이 usrCheckPro.jsp에서 usrDAO의 메소드만 호출한다.</p> <p>탈퇴시점을 기준으로 한 달 후에 정보가 삭제되므로 탈퇴여부와 탈퇴시점을 함께 저장해야 한다.</p> <p>탈퇴 후 usrLogoutPro.jsp로 이동하여 로그아웃 처리 된다.</p>		

usrDAO java	<pre>//회원 탈퇴하기 public int delete(String usrId) { Connection conn = null; PreparedStatement pstmt = null; String sql = "update Usr set usrDelete = 1, delDate = ? where usrId = ?"; try { conn = getConnection(); pstmt = conn.prepareStatement(sql); pstmt.setTimestamp(1, new Timestamp(System.currentTimeMillis())); pstmt.setString(2, usrId); return pstmt.executeUpdate(); //계정 삭제 요청 } catch (Exception e) { e.printStackTrace(); } finally { closeDBResources(pstmt, conn); } return -1; //DB오류 }</pre> <p>아이디를 매개변수로 받아서 그 아이디의 탈퇴여부 필드(usrDelete)의 값을 1로 수정하고 탈퇴 및 가입 날짜에 현재 시각을 저장한다.</p>		
화면 구현			
환경	개발환경	서비스그룹	화면 처리
W-U-7	회원 목록 조회		
기능	<p>관리자 계정으로만 진입이 가능하다.</p> <p>한 페이지 당 10명의 회원이 표시 되도록 페이징 처리하며, 탈퇴회원이 가장 먼저 보이도록 함. 그 다음은 가입 또는 탈퇴일의 내림차순으로 정렬한다.</p> <p>탈퇴 여부는 숫자 대신 사용중 회원, 탈퇴 회원으로 표시한다.</p> <p>아이디와 이메일로 회원 검색이 가능하다.</p> <p>저장된 회원이 없을 경우 해당 내용을 안내한다.</p>		
usrList jsp	<pre>if(session.getAttribute("usrId")!=null) { usrId = (String) session.getAttribute("usrId"); } if(usrId==null !usrId.equals("admin")) { session.setAttribute("usrId", usr.getUsrId()); PrintWriter script = response.getWriter(); script.println("<script>"); script.println("alert('관리자 계정으로 로그인 후 확인 가능합니다.');""); script.println("location.href='index.jsp'"); script.println("</script>"); } else {</pre> <p>관리자 계정으로만 진입할 수 있기 때문에 로그인하지 않았거나 관리자가 아닌 사용자가 진입할 경우 팝업으로 알린 후 메인 화면으로 이동한다.</p>		

```
String pageNum = request.getParameter("pageNum");
if (pageNum == null) {
    pageNum = "1";
}
```

선택된 페이지 번호가 없을 경우 기본값인 1페이지로 설정한다.

```
int currentPage = Integer.parseInt(pageNum);
int startRow = (currentPage - 1) * pageSize + 1;
int endRow = currentPage * pageSize;
int totalUsr = 0;
List<Usr> usrList = null;
```

그 외 페이지 처리하는데 필요한 변수들을 지정. 회원 목록을 담은 리스트를 선언한다.

```
String option = request.getParameter("option");
String searchWord = request.getParameter("searchWord");
```

이전 페이지에서 받아온 검색 항목과 검색어 값을 각각의 변수에 할당.

```
if (option!=null && searchWord!=null){
    totalUsr = usrDAO.totalSearch(option, searchWord, startRow, endRow);
    if (totalUsr > 0) {
        usrList = usrDAO.searchUsr(option, searchWord, startRow, endRow);
    }
}
else {
    totalUsr = usrDAO.getTotalUsr();
    if (totalUsr > 0) {
        usrList = usrDAO.getUsrs(startRow, endRow);
    }
}
```

만약 이전페이지에서 받아온 검색 항목과 검색어 값이 있을 경우 그 값에 대한 총 회원 수와 리스트를 각각의 변수에 할당하며, 그렇지 않을 경우 전체 회원 수와 리스트를 할당한다.

회원이 없는 경우에는 아래와 같이 회원이 없음을 표 내용으로 알린다.

```
if (totalUsr == 0) { %>
    <tr>
        <td colspan="6">가입 회원이 없습니다.</td>
    </tr>
```

리스트에 저장된 회원이 없을 경우 회원정보 대신 회원이 없다는 문구를 출력한다.

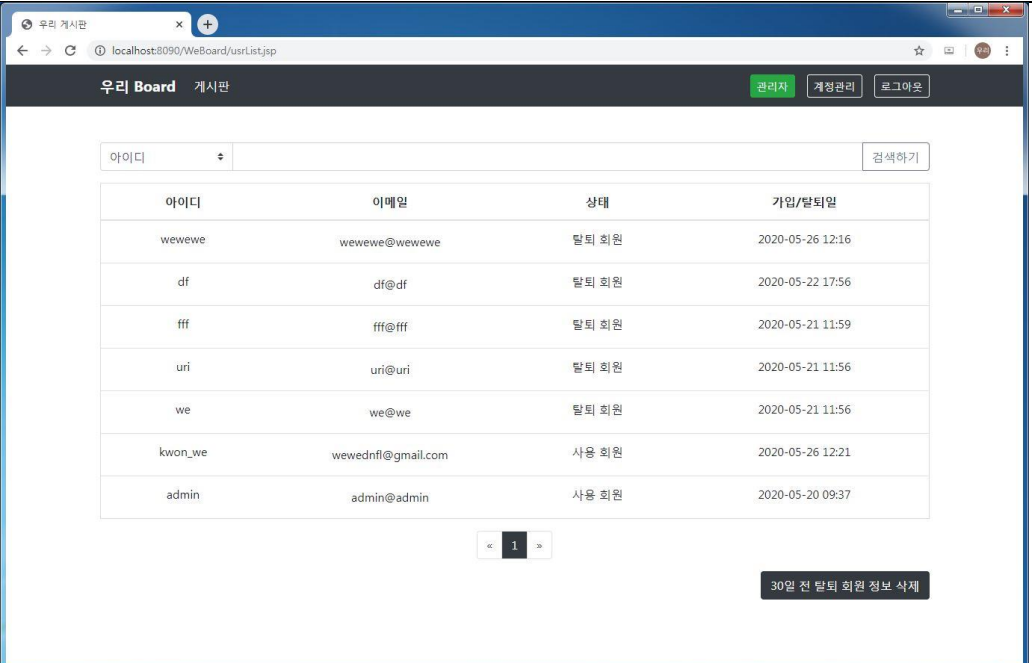
```
if (totalUsr > 0) {
    int pageCount = totalUsr / pageSize + (totalUsr % pageSize == 0 ? 0 : 1);
    int startPage = 1;

    if(currentPage % 10 != 0)
        startPage = (int)(currentPage/10)*10 + 1;
    else
        startPage = ((int)(currentPage/10)-1)*10 + 1;

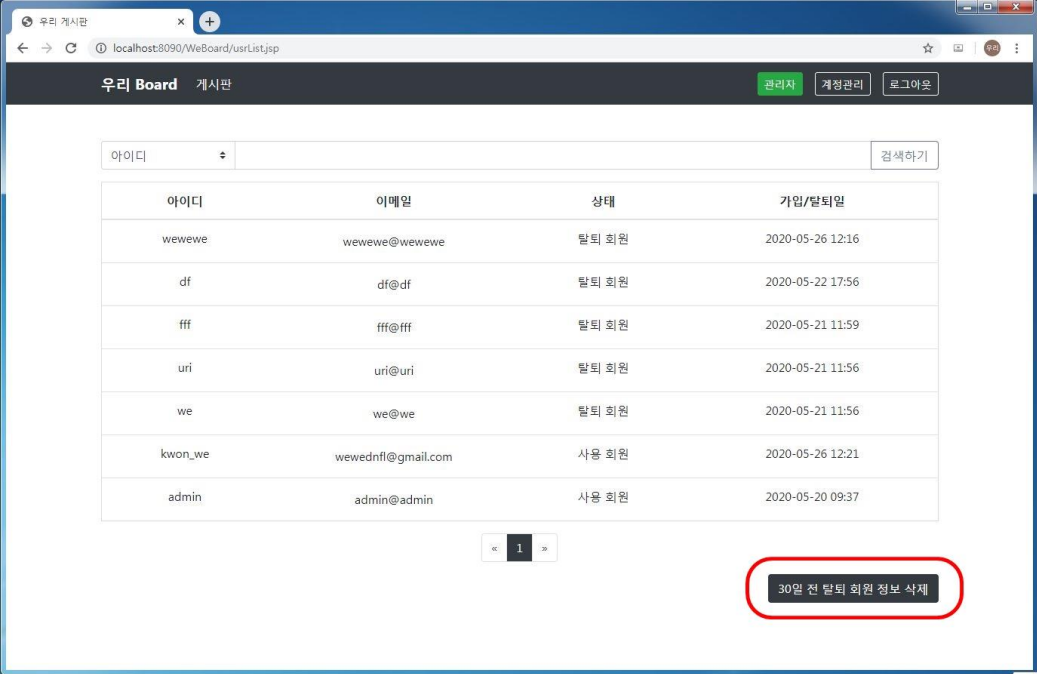
    int pageBlock = 10;
    int endPage = startPage + pageBlock - 1;
    if (endPage > pageCount) endPage = pageCount;
```

위와 같이 페이지에 나타낼 회원 수(10)와 전체 회원 수를 이용하여 페이지 처리를 위한 계산을 한 후 한번에 최대 10개의 페이지까지 보이도록 하였다.

<div data-bbox="204 237 300 322" data-label="Text"> usrDAO java </div>	<div data-bbox="363 237 968 835" data-label="Text"> <pre>//전체 회원 수 추출 public int getTotalUsr() throws Exception { Connection conn = null; PreparedStatement pstmt = null; ResultSet rs = null; String sql = "select count(*) from Usr"; try { conn = getConnection(); pstmt = conn.prepareStatement(sql); rs = pstmt.executeQuery(); if (rs.next()) { return rs.getInt(1); } } catch (Exception ex) { ex.printStackTrace(); } finally { closeDBResources(rs, pstmt, conn); } return -1; }</pre> </div> <div data-bbox="363 862 1246 893" data-label="Text"> <p>Usr테이블의 전체 레코드의 수를 반환하며, DB오류일 경우 -1을 반환한다.</p> </div> <div data-bbox="363 916 1394 1821" data-label="Text"> <pre>//전체 회원 목록 보기 public List<Usr> getUsrs(int start, int end) { Connection conn = null; PreparedStatement pstmt = null; ResultSet rs = null; List<Usr> usrList = null; try { conn = getConnection(); String sql = "select * from (select rowNum rnum, B.* from " + "(select * from Usr order by usrDelete asc, delDate desc) B) " + "where rnum >= ? and rnum <= ?"; pstmt = conn.prepareStatement(sql); pstmt.setInt(1, start); pstmt.setInt(2, end); rs = pstmt.executeQuery(); if (rs.next()) { usrList = new ArrayList<Usr>(end); do { Usr usr = new Usr(); usr.setUsrId(rs.getString("usrId")); usr.setUsrEmail(rs.getString("usrEmail")); usr.setUsrDelete(rs.getInt("usrDelete")); usr.setDelDate(rs.getTimestamp("delDate")); usrList.add(usr); } while (rs.next()); } } catch (Exception e) { e.printStackTrace(); } finally { closeDBResources(rs, pstmt, conn); } return usrList; }</pre> </div> <div data-bbox="363 1843 1378 2029" data-label="Text"> <p>전 회원의 정보를 rowNum을 활용하여 usrDelete의 오름차순, delDate의 내림차순으로 정렬한 뒤, usrList에서 계산한 첫 번호와 끝 번호만 받아서 리스트에 저장한다. 만약, 검색을 했다면, 검색 항목을 switch문으로 구분하고 위의 내용의 조건문에 검색어 앞 뒤로 검색어를 포함하는 값을 구하기 위한 % 기호를 붙여, 같은 값을 찾는 내</p> </div>
--	---

	용을 추가한다. (검색에 대한 소스코드 생략)
화면 구현	

환경	개발환경	서비스그룹	화면 처리
W-U-8	회원 정보 삭제		
기능	관리자 계정으로만 진입이 가능하다. 별도의 Form 없이 usrList.jsp 하단에 위치한 버튼을 클릭 시 탈퇴 회원 중 탈퇴한지 한 달 이상 된 회원의 정보만 삭제한다.		
usrDAO java	<pre>//탈퇴한 지 30일이 지난 회원정보 삭제 public int deleteFinal(String usrId) { Connection conn = null; PreparedStatement pstmt = null; String sql = "delete from Usr where usrDelete = 1 and " + "trunc(MONTHS_BETWEEN(SYSDATE, deldate)) >= 1"; try { conn = getConnection(); pstmt = conn.prepareStatement(sql); if (usrId.equals("admin")) { return pstmt.executeUpdate(); } else return 0; //사용자 계정이 아님 } catch (Exception e) { e.printStackTrace(); } finally { closeDBResources(pstmt, conn); } return -1; //DB오류 }</pre> <p>로그인한 아이디를 매개변수로 받아서 그 아이디가 관리자 계정이 맞으면 탈퇴 계정 중 탈퇴한 지 한 달이 지난 회원들의 정보만 삭제한다.</p> <p>삭제에 성공하면 0을 반환하고 실패하면 -1을 반환한다.</p>		

Usr DelFinal Pro.jsp	<pre> if(session.getAttribute("usrId")!=null) { usrId = (String) session.getAttribute("usrId"); } if(usrId==null !usrId.equals("admin")) { session.setAttribute("usrId", usr.getUsrId()); PrintWriter script = response.getWriter(); script.println("<script>"); script.println("alert('관리자 계정으로 로그인 후 확인 가능합니다.~')"); script.println("location.href='index.jsp'"); script.println("</script>"); } else { </pre> <p>관리자 계정으로만 진입할 수 있기 때문에 로그인하지 않았거나 관리자가 아닌 사용자가 진입할 경우 팝업으로 알린 후 메인 화면으로 이동한다.</p> <pre> int result = usrDAO.deleteFinal(usrId); </pre> <p>java코드에서 가져온 결과값을 result에 대입하고 그 값이 -1일 경우, DB오류에 대한 팝업을 하고 그 외에는 삭제되었다는 팝업을 띄운다.</p>		
화면 구현			
환경	개발환경	서비스그룹	화면 처리
W-B-1	게시 글 작성		
기능	<p>로그인 후에만 진입이 가능하다.</p> <p>작성 항목은 제목, 내용, 업로드할 파일이며 그 중 제목과 내용은 필수 입력 항목이다.</p> <p>글 저장하면 자동으로 작성자에 로그인한 아이디가 입력된다.</p>		

bbsWrite Form.jsp	<pre> if (session.getAttribute("usrId")!=null){ usrId = (String) session.getAttribute("usrId"); } if(usrId==null) { PrintWriter script = response.getWriter(); script.println("<script>"); script.println("alert('로그인 후 작성 가능합니다.')"); script.println("location.href='usrLoginForm.jsp'"); script.println("</script>"); } </pre> <p>세션에 저장된 값이 있을 경우에만 글을 작성할 수 있으므로 위와 같이 저장된 값이 null일 경우 팝업으로 안내 후 로그인 화면으로 이동한다.</p> <pre> <input type="hidden" name="bbsId" value="<%=bbsId%>"> <input type="hidden" name="usrId" value="<%=usrId%>"> <input type="hidden" name="ref" value="<%=ref%>"> <input type="hidden" name="re_step" value="<%=re_step%>"> <input type="hidden" name="re_level" value="<%=re_level%>"> </pre> <p>위의 값들은 Bbs 테이블에 값을 저장할 때는 필요하지만 사용자가 지정할 필요 없이 자동 부여되는 값이므로 숨김 처리 한다.</p> <pre> if (usrId.equals("admin")){ %> <tr> <td colspan="2"> <select name="option" class="form-control"> if (bbsDAO.totalBbs(1)<5) { %> <option value="notice" class="form-control" selected="selected">공지글</option> }else{ %> <option value="notice" class="form-control" disabled>공지글 // 최대 5개 까지만 작성 가능함 } %> <option value="normal" class="form-control">일반글</option> </select> </pre> <p>공지 글은 관리자 계정으로 진입 시 최대 5개 까지 작성 가능하므로 위와 같이 관리자 계정을 거른 후 그 외 일반 사용자일 경우 option에 normal 값만 넣어 tr태그 채 숨김 처리 한다.</p>
bbsDAO java	<pre> sql = "select max(bbsId) from Bbs"; pstmt = conn.prepareStatement(sql); rs = pstmt.executeQuery(); if(rs.next()) { number = rs.getInt(1) + 1; } else { number = 1; } </pre> <p>글 번호를 매기기 위해 글 번호의 최대값 + 1을 반환하고 작성된 글이 없을 때는 1을 반환한다.</p>


```

closeDBResources(rs, pstmt);
if(bbsId!=0) { // 댓글
    sql = "update Bbs set re_step=re_step+1 where ref=? and re_step>?";
    pstmt = conn.prepareStatement(sql);
    pstmt.setInt(1, ref);
    pstmt.setInt(2, re_step);
    pstmt.executeUpdate();
    re_step = re_step + 1;
    re_level = re_level + 1;
} else { // 원본글쓰기 - 댓글x
    ref = number;
    re_step = 0;
    re_level = 0;
}

```

작성하려는 글의 답 글 여부를 확인하여 글의 순서와 단계를 설정한다.

위 사항이 모두 완료 되면 받아온 입력된 정보를 Bbs테이블에 저장한다.

```

sql = "insert into Bbs (bbsId, usrId, bbsTitle, bbsDate, ref, re_step, "
      + "re_level, bbsContent, notice, fileName) values (?,?,?,?,?,?,?,?);";
pstmt = conn.prepareStatement(sql);
pstmt.setInt(1, number);
pstmt.setString(2, bbs.getUsrId());
pstmt.setString(3, bbs.getBbsTitle());
pstmt.setTimestamp(4, bbs.getBbsDate());
pstmt.setInt(5, ref);
pstmt.setInt(6, re_step);
pstmt.setInt(7, re_level);
pstmt.setString(8, bbs.getBbsContent());
pstmt.setInt(9, bbs.getNotice());
pstmt.setString(10, bbs.getFileName());
pstmt.executeUpdate();

```

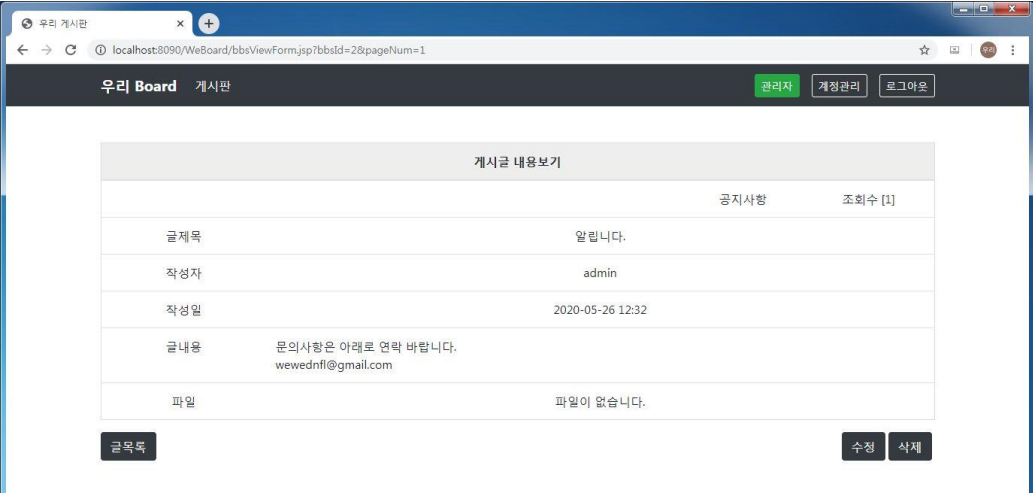
화면 구현

일반 사용자가 글 작성 시

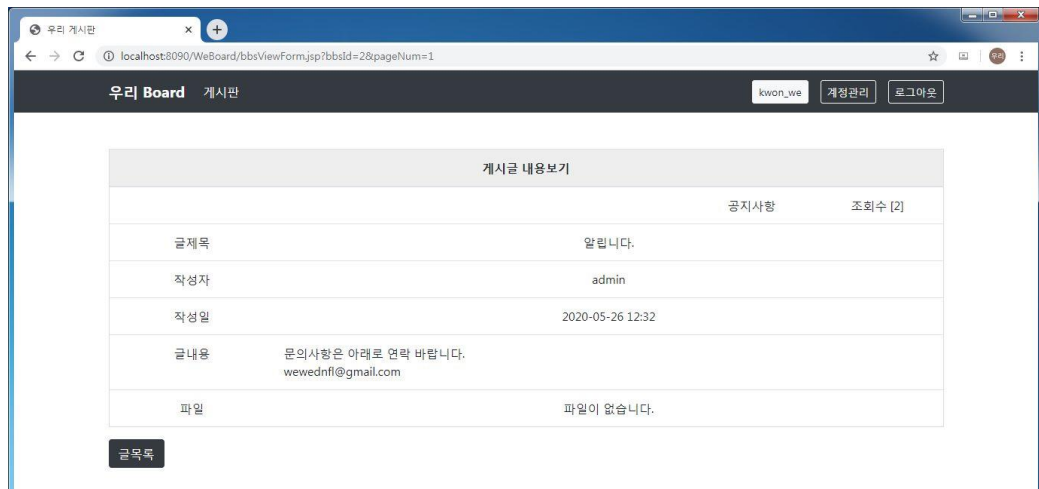
관리자가 글 작성 시 (다음 페이지)

관리자가 글 작성할 때 공지 글이 5개일 경우

환경	개발환경	서비스그룹	화면 처리
W-B-2	게시 글 조회		
기능	<p>비 로그인 시에도 진입이 가능하나, 답글 쓰기 버튼 클릭 시 로그인이 필요하다는 안내 팝업 후 로그인 화면으로 이동한다.</p> <p>로그인 후 진입하여 답글 쓰기 버튼 클릭 시에는 글 작성 화면으로 이동한다.</p> <p>만약, 로그인 한 아이디가 조회한 글의 작성자와 동일할 경우 수정, 삭제 버튼이 추가</p>		

	로 표시되며 클릭 시 각각의 기능을 수행한다.
bbsView Form.jsp	<pre> if (session.getAttribute("usrId")!=null){ usrId = (String) session.getAttribute("usrId"); } 로그인하지 않아도 글 조회는 가능하므로 세션에 저장된 아이디만 불러오고 별도의 팝업안내는 하지 않는다. if (request.getParameter("bbsId")!=null) { bbsId = Integer.parseInt(request.getParameter("bbsId")); } if (bbsId==0) { PrintWriter script = response.getWriter(); script.println("<script>"); script.println("alert('유효하지 않은 글입니다.');""); script.println("location.href='bbsList.jsp'"); script.println("</script>"); } 글 번호가 있어야 해당 글을 조회할 수 있기 때문에 이전 화면에서 받아온 값이 있으 면 bbsId 변수에 값을 저장하고 없을 경우 팝업으로 안내 후 글 목록으로 이동한다. <td colspan="3"> <%= bbs.getBbsTitle().replaceAll(" ", "&nbsp;").replaceAll("<", "&lt;"); .replaceAll(">", "&gt;").replaceAll("\n", "&br>") %> </td> 표시할 제목과 내용에 특수문자나 공백 등이 추가될 경우 html기호와 구분이 되지 않 아 제대로 출력이 되지 않을 수 있기 때문에 값을 치환해준다. </pre>
bbsDAO. java	<pre> pstmt = conn.prepareStatement("update bbs set readCount = readCount+1 where bbsId = ?"); pstmt.setInt(1, bbsId); pstmt.executeUpdate(); closeDBResources(pstmt); pstmt = conn.prepareStatement("select * from bbs where bbsId = ?"); pstmt.setInt(1, bbsId); rs = pstmt.executeQuery(); </pre> <p>조회를 했으니 조회수에 +1을 해 주고 글 번호에 대한 내용들을 받아 bbs클래스에 저 장시킨다.</p>
화면 구현	<p>관리자 계정으로 공지 글 조회 시. (다음 페이지)</p> 

일반 사용자 계정으로 공지 글 조회 시

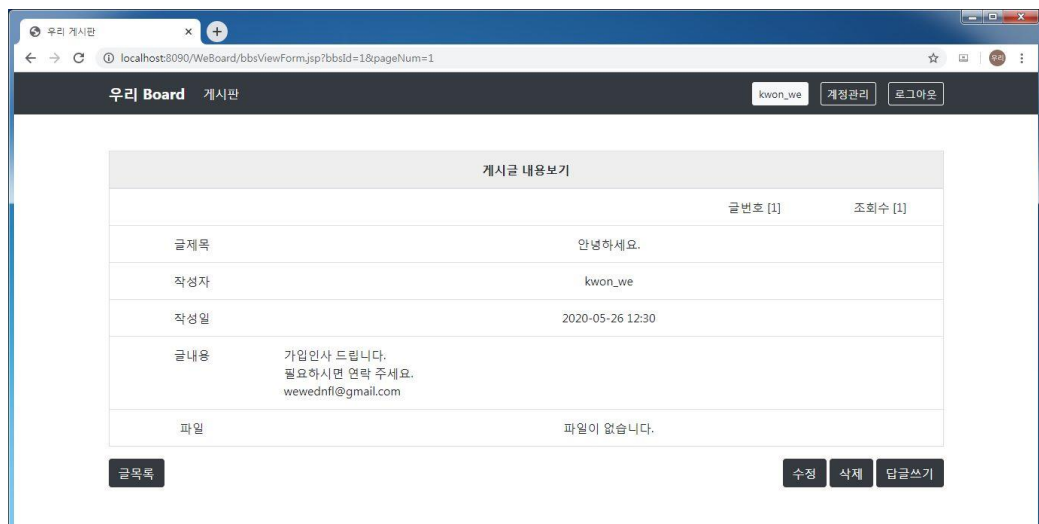


우리 Board | 게시판 | kwon_we | 계정관리 | 로그아웃

게시글 내용보기		공지사항	조회수 [2]
글제목	알립니다.		
작성자	admin		
작성일	2020-05-26 12:32		
글내용	문의사항은 아래로 연락 바랍니다. wewednfi@gmail.com		
파일	파일이 없습니다.		

공지글 목록

일반 사용자 계정으로 본인이 작성한 글 조회 시



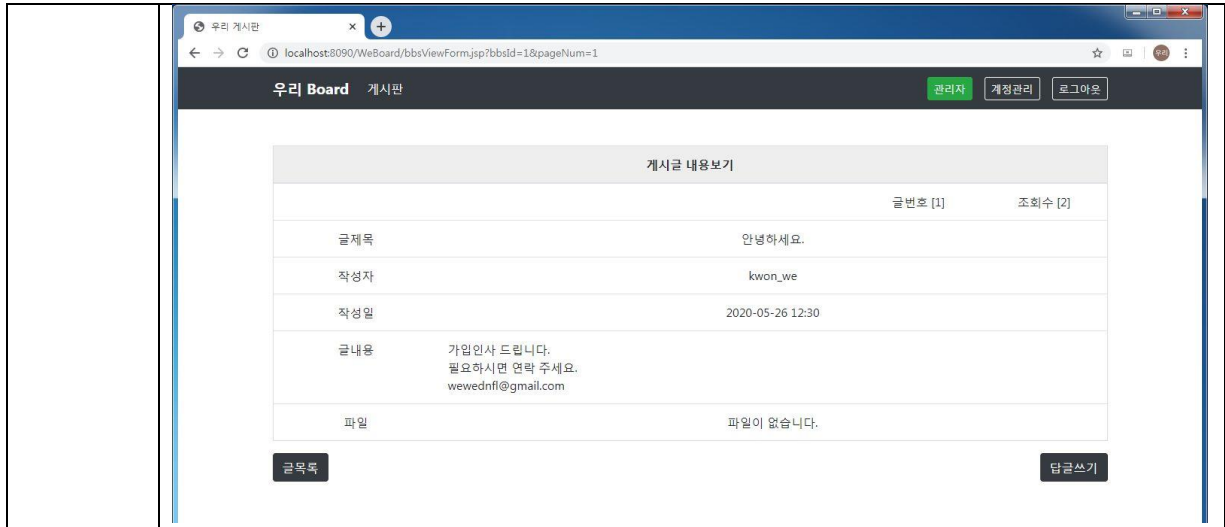
우리 Board | 게시판 | kwon_we | 계정관리 | 로그아웃

게시글 내용보기		글번호 [1]	조회수 [1]
글제목	안녕하세요.		
작성자	kwon_we		
작성일	2020-05-26 12:30		
글내용	가임인사 드립니다. 필요하시면 연락 주세요. wewednfi@gmail.com		
파일	파일이 없습니다.		

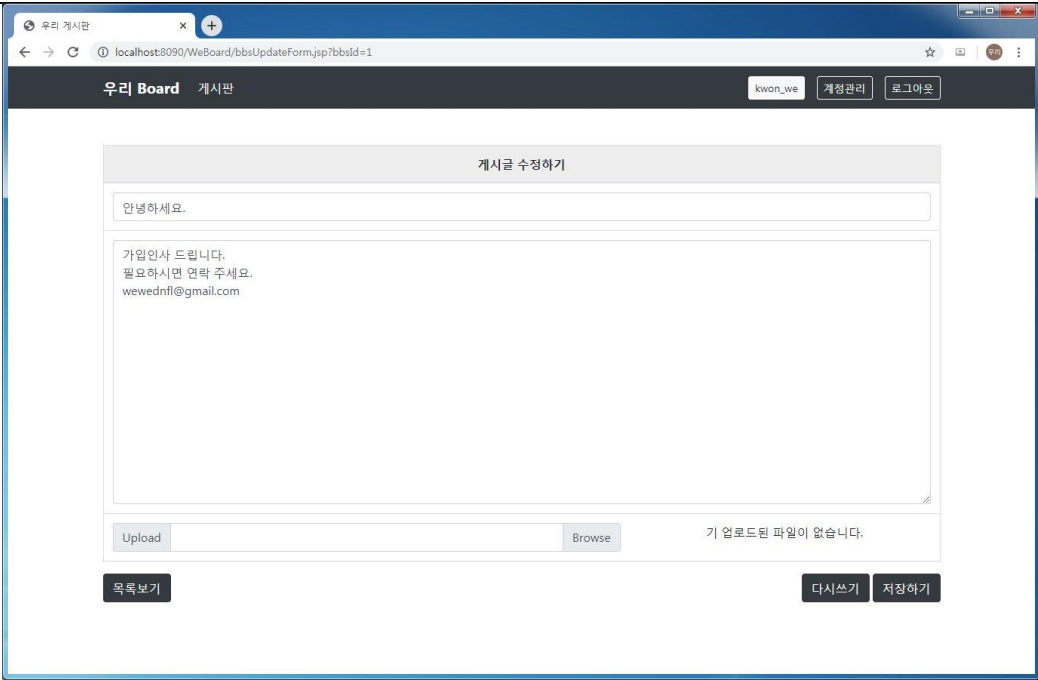
공지글 목록

수정 | 삭제 | 답글쓰기

타인이 작성한 글 조회 시



환경	개발환경	서비스그룹	화면 처리
W-B-3	게시 글 수정		
기능	<p>게시 글 조회 화면에서 표시된 버튼 클릭 시 진입 된다.</p> <p>각 항목에 기존에 저장된 정보가 자동으로 입력되며, 글 작성 시와 마찬가지로 제목과 내용에는 공백이 없어야 한다.</p> <p>기 업로드 된 파일이 있을 경우 파일 업로드 선택 칸 우측에 해당 파일의 이름을 표시 한다.</p>		
bbs Update Form.jsp	<pre> if (session.getAttribute("usrId")!=null){ usrId = (String) session.getAttribute("usrId"); } if (usrId==null) { PrintWriter script = response.getWriter(); script.println("<script>"); script.println("alert('로그인 후 수정 가능합니다.');""); script.println("location.href='usrLoginForm.jsp'"); script.println("</script>"); } int bbsId = 0; try{ if (request.getParameter("bbsId")!=null) { bbsId = Integer.parseInt(request.getParameter("bbsId")); } if (bbsId==0) { PrintWriter script = response.getWriter(); script.println("<script>"); script.println("alert('유효하지 않은 글입니다.');""); script.println("location.href='bbsList.jsp'"); script.println("</script>"); } } </pre> <p>로그인 후 본인 글에 대해 수정이 가능하므로 로그인 되지 않았거나 이전 화면에서 가지고 온 글 번호가 없을 경우 각각의 팝업 안내 후 관련 페이지로 이동한다.</p>		

	<pre><% String filename = bbs.getFileName(); if(filename !=null && !filename.equals("")) { %> <td>기 업로드 된 파일 : <%=filename %></td> <% } else { %> <td>기 업로드된 파일이 없습니다.</td> <% } %></pre> <p>업로드 할 파일에 대한 정보도 나타내 준다.</p>		
bbsDAO. java	<pre>pstmt = conn.prepareStatement("select * from bbs where bbsId = ?"); pstmt.setInt(1, bbsId); rs = pstmt.executeQuery(); 현재 글에서 수정을 해야 하므로 입력 창에 미리 띄우기 위해 글 번호에 저장된 값들 을 가져온다. 이때 글을 조회한 것은 아니므로 조회수 증가는 시키지 않는다. pstmt = conn.prepareStatement("select usrId from Bbs where bbsId = ?"); pstmt.setInt(1, bbs.getBbsId()); rs = pstmt.executeQuery(); if(rs.next()){ if(rs.getString("usrId").equals(bbs.getUsrId())){ closeDBResources(pstmt); sql = "update Bbs set bbsTitle = ?, bbsContent = ?, fileName = ? where bbsId = ?"; pstmt = conn.prepareStatement(sql); pstmt.setString(1, bbs.getBbsTitle()); pstmt.setString(2, bbs.getBbsContent()); pstmt.setString(3, bbs.getFileName()); pstmt.setInt(4, bbs.getBbsId()); pstmt.executeUpdate(); return 1; } }</pre> <p>글 번호에 대한 값들이 있는지 먼저 조회한 후 있으면 수정한 내용들을 저장, 1을 반 환한다. 글 번호가 없다면 0, DB오류는 -1을 반환한다.</p>		
화면 구현			
환경	개발환경	서비스그룹	화면 처리
W-B-4	게시 글 삭제		
기능	게시 글 조회 화면에서 표시된 버튼 클릭 시 화면 이동 없이 안내 팝업 후 삭제된다.		

bbsDAO. java	<pre> pstmt = conn.prepareStatement("select usrId from Bbs where bbsId = ?"); pstmt.setInt(1, bbsId); rs = pstmt.executeQuery(); if(rs.next()){ if(rs.getString("usrId").equals(usrId)){ closeDBResources(pstmt); pstmt = conn.prepareStatement("delete from bbs where bbsId=?"); pstmt.setInt(1, bbsId); pstmt.executeUpdate(); return 1; //글삭제 성공 } } 글 번호가 있는지 확인한 후 있으면 삭제 및 1을 반환, 없으면 0, DB오류는 -1을 반환한다. </pre>
-----------------	---

화면 구현	
-------	--

환경	개발환경	서비스그룹	화면 처리
W-B-5	게시판 목록 조회		
기능	<p>비 로그인 시에도 진입이 가능하나, 글 쓰기 버튼 클릭 시 로그인이 필요하다는 안내 팝업 후 로그인 화면으로 이동한다.</p> <p>한 페이지 당 표시할 글의 수를 사용자가 선택 가능하도록 한다.</p> <p>제목+내용, 제목, 내용, 작성자로 게시 글 검색이 가능하다.</p> <p>목록의 글 제목을 클릭 시 해당 글의 조회 화면으로 이동한다.</p> <p>목록의 작성자 클릭시 작성자 아이디 아래로 게시글 보기와 이메일 보내기 탭이 생성되며 클릭 시 각각의 기능을 수행한다.</p> <p>저장된 글이 없을 경우 해당 내용을 안내한다.</p>		

bbsLost .jsp	<pre> if (session.getAttribute("usrId")!=null){ usrId = (String) session.getAttribute("usrId"); } int pageSize = 10; if (request.getParameter("pageSize")!=null){ pageSize = Integer.parseInt(request.getParameter("pageSize")); } String option = request.getParameter("option"); String searchWord = request.getParameter("searchWord"); String pageNum = (String)request.getParameter("pageNum"); if (pageNum == null) { pageNum = "1"; } 글 목록은 로그인을 하지 않아도 볼 수 있으므로 로그인한 정보는 usrId에 저장만 시 킨다. 페이지 당 글 수(pageSize)는 기본 10개로 하되 이전 화면에서 받아온 값이 있 을 경우 그 값을 대입시킨다. 페이지 수(pageNum) 또한 이전 화면에서 받아온 값을 대입시키되 없을 경우 1을 대입한다. 이전화면에서 받아온 검색 항목과 검색할 단어를 각 변수에 대입시키고 값의 유무에 따라 아래와 같이 진행한다. if (option!=null && searchWord!=null){ totalBbs = bbsDAO.totalSearch(option, searchWord, startRow, endRow, 0); if (totalBbs > 0) { bbsList = bbsDAO.searchBbs(option, searchWord, startRow, endRow, 0); } totalNotice = bbsDAO.totalSearch(option, searchWord, startRow, endRow, 1); if (totalNotice > 0){ noticeList = bbsDAO.searchBbs(option, searchWord, 1, 10, 1); } } else { totalBbs = bbsDAO.totalBbs(0); //16 if (totalBbs > 0) { bbsList = bbsDAO.getBbsList(startRow, endRow, 0); } totalNotice = bbsDAO.totalBbs(1); if (totalNotice > 0) { noticeList = bbsDAO.getBbsList(1, 10, 1); } } number = totalBbs-(currentPage-1)*pageSize; </pre> <p>큰 틀은 회원 목록 화면과 동일하나, 게시판의 경우 어느 페이지 에서든 상단에 공지 글이 표시되어야 하므로 별도의 테이블을 두어 공지 글과 일반 글을 별도의 리스트에 담았다.</p> <p>글 번호는 목록에 표시될 글을 기준으로 표시되어야 하므로 별도의 변수를 선언하여 총 글 수와 페이지 번호 등에 따라 계산되도록 하였다.</p>
---------------------	---


```

if (totalNotice == 0) { %>
    <tr>
        <td colspan="5">공지사항이 없습니다.</td>
    </tr>
} else {
    for (int i = 0 ; i < noticeList.size() ; i++) {
        bbs.Bbs bbs = noticeList.get(i);    %>
        <tr>

```

작성된 글이 없을 경우 안내가 필요하므로, 공지 글과 일반 글 모두 위와 같이 총 글 수에 따라 구분하였다.

```

<div class="dropdown">
    <a class="dropdown-toggle" href="#" role="button" id="dropdownMenuLink"
        <%=bbs.getUsrId()%>
    </a>
    <div class="dropdown-menu" aria-labelledby="dropdownMenuLink">
        <a class="dropdown-item"
            href="bbsList.jsp?option=usrId&searchWord=<%=bbs.getUsrId()%>">
            게시글 보기
        </a>
        <a class="dropdown-item"
            href="mailto:<%=bbsDAO.getWriterEmail(usrId)%>">
            이메일 보내기
        </a>
    </div>
</div>

```

부트스트랩의 dropdown 클래스를 활용하여 작성자 클릭 시 아래로 게시 글 보기와 이메일 보내기 라는 2개의 탭이 생성되고 클릭 시 각각의 기능을 수행한다.

```

if (totalUsr > 0) {
    int pageCount = totalUsr / pageSize + (totalUsr % pageSize == 0 ? 0 : 1);
    int startPage = 1;

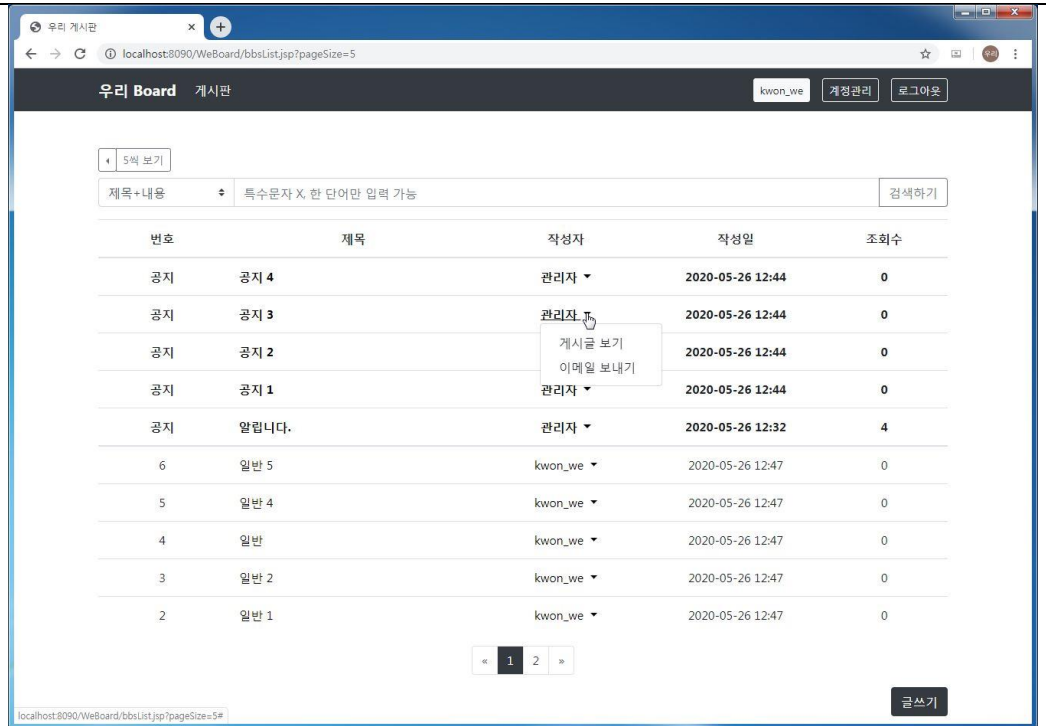
    if(currentPage % 10 != 0)
        startPage = (int)(currentPage/10)*10 + 1;
    else
        startPage = ((int)(currentPage/10)-1)*10 + 1;

    int pageBlock = 10;
    int endPage = startPage + pageBlock - 1;
    if (endPage > pageCount) endPage = pageCount;

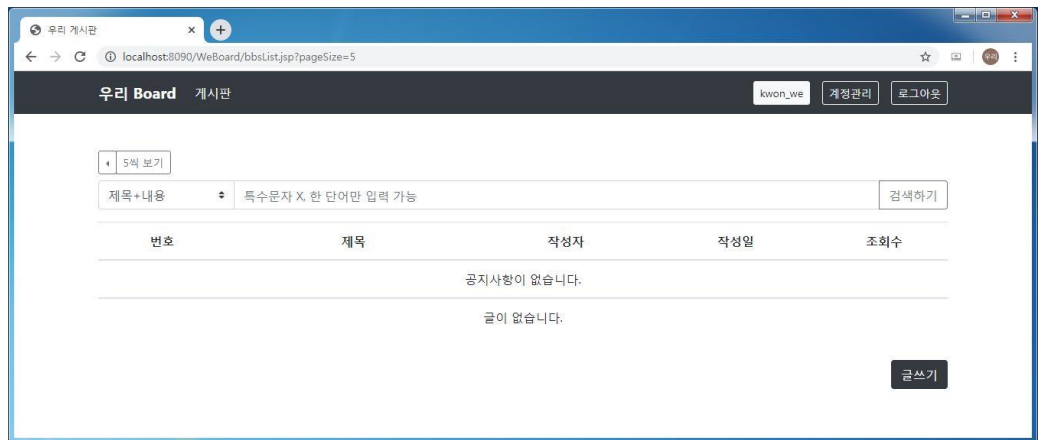
```

위와 같이 페이지에 나타낼 글의 수와 전체 글 수를 이용하여 페이지 처리를 위한 계산을 한 후 한번에 최대 10개의 페이지까지 보이도록 하였다.

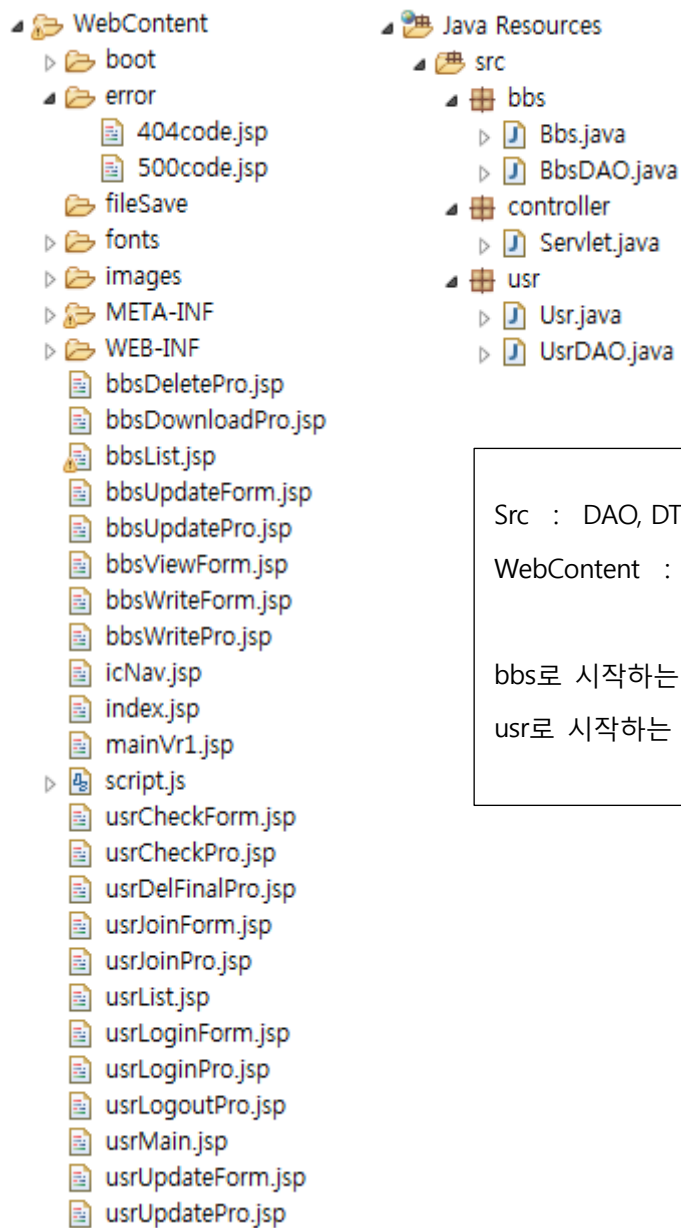
bbsDAO. java	<pre> //각 글의 총 개수 public int totalBbs(int what) throws Exception { Connection conn = null; PreparedStatement pstmt = null; ResultSet rs = null; try { conn = getConnection(); pstmt = conn.prepareStatement("select count(*) from Bbs where notice = ?"); pstmt.setInt(1, what); rs = pstmt.executeQuery(); if (rs.next()) { return rs.getInt(1); } } catch (Exception ex) { ex.printStackTrace(); } finally { closeDBResources(rs, pstmt, conn); } return 0; } </pre> <p>공지 여부에 따라 총 2개의 리스트에 목록을 담을 예정이므로 글의 총 개수도 목록에 따라 구하도록 일반 글은 0, 공지 글은 1을 매개변수로 받아서 조건문으로 걸러낸다.</p> <pre> //각 글의 전체 목록 보기 //1:공지//0:일반 public List<Bbs> getBbsList(int start, int end, int what) throws Exception { Connection conn = null; PreparedStatement pstmt = null; ResultSet rs = null; List<Bbs> bbsList = null; try { conn = getConnection(); String sql = "select * from (select rowNum rnum, B.* from " + "(select * from Bbs where notice = ? " + "order by ref desc, re_step asc) B) " + "where rnum >= ? and rnum <= ?"; pstmt = conn.prepareStatement(sql); pstmt.setInt(1, what); pstmt.setInt(2, start); pstmt.setInt(3, end); rs = pstmt.executeQuery(); } } </pre> <p>각각의 글을 RowNum을 활용하여 글 그룹의 내림차순, 글 단계의 오름차순으로 정렬한 뒤, bbsList에서 계산한 첫 번호와 끝 번호만 받아서 리스트에 저장한다.</p> <p>만약, 검색을 했다면, 검색 항목을 switch문으로 구분하고 위의 내용의 조건문에 검색어 앞 뒤로 검색어를 포함하는 값을 구하기 위한 % 기호를 붙여, 같은 값을 찾는 내용을 추가한다. (검색에 대한 소스코드 생략)</p>
화면 구현	글 있음, 작성자 클릭 시 dropdown (다음 페이지)



글 없음.



2. 파일 구조



Src : DAO, DTO, Controller(Servlet)

WebContent : View, Controller(xxxPro)

bbs로 시작하는 파일들은 게시판 관련 파일.

usr로 시작하는 파일들은 회원정보 관련 파일.

3. 추후 개선해야 할 사항

- 게시 글 삭제 시 팝업 창에 취소버튼을 추가하여 확인 클릭 시에만 게시 글이 삭제되고 취소를 클릭할 경우 처리되지 않도록 한다.
- 회원이 탈퇴를 하지 않더라도 관리자의 권한으로 계정을 삭제할 수 있도록 한다.
- 회원이 본인이 작성한 글을 삭제하지 않더라도 관리자의 권한으로 부적합하다 판단된 게시 글을 삭제할 수 있도록 한다.
- 관리자가 작성한 글 수정 시 공지 글 또는 일반 글로 변경 여부를 선택할 수 있도록 한다.

배포 환경

Apache tomcat v8.5 이상

IDE : Eclipse EE 4.0 이상

DBMS : Oracle Developer 18 이상

Oracle Express Edition 11 이상

JDK : jdk 1.8 이상 cos.jar (파일 업로드, 다운로드 라이브러리) WeBoard.war

형상관리 : git, github(<https://github.com/KwonUriWe>)

Contact Me

이름 : 권우리

연락처 : 010-5146-3979

이메일 주소 : wewednfl@gmail.com