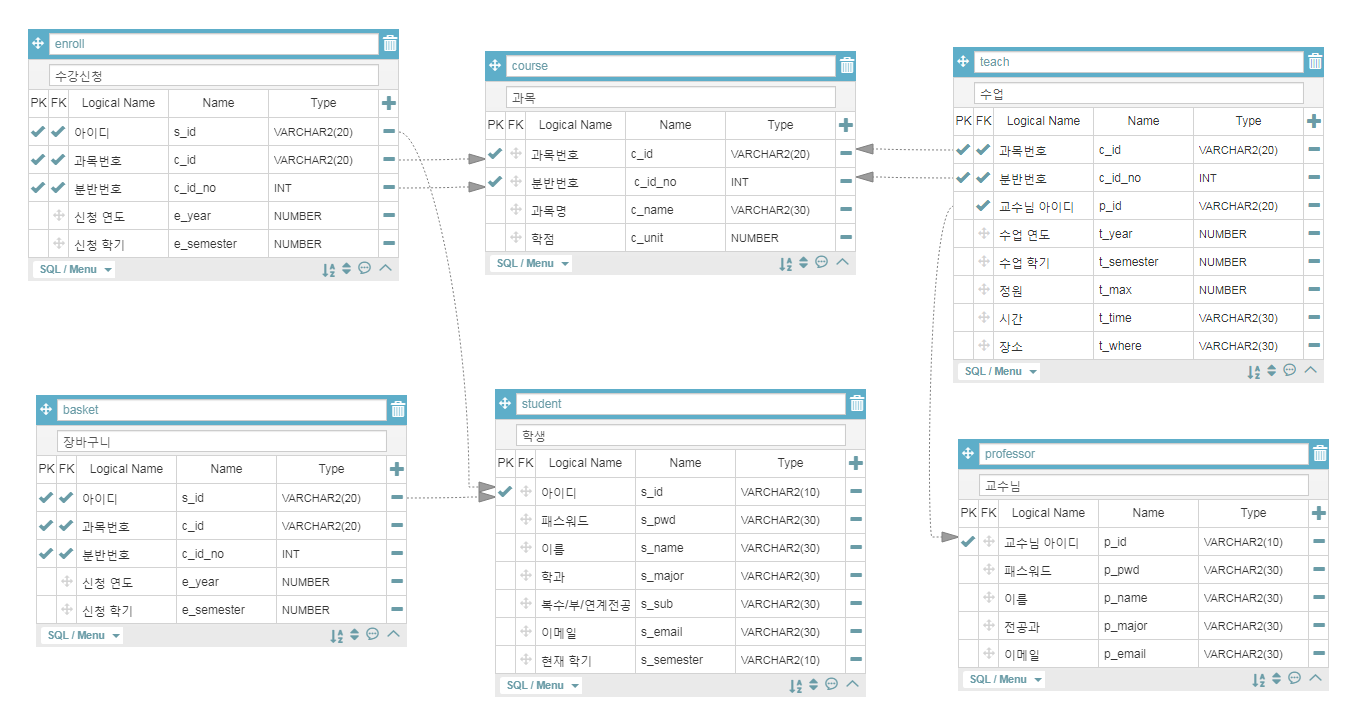
**데이터베이스 프로그래밍 프로젝트**

종강시켜조

김윤아, 원효선

1. DB 설계도



2. 사용한 주요 SQL 질의문 /커서(파라미터) 소개 및 소스 위치

1) 파라미터 있는 커서

* c\_cur(s\_id VARCHAR2, year NUMBER, semester NUMBER): 아이디와 연도, 학기를 입력받아 학생이 수강 신청한 총 과목 수와 총 학점을 select한 결과를 저장하는 커서. project/procedureFunction/verify\_enroll.sql에 위치. 프로시저 selectTimeTable 안에 선언되어 있다.

2) stored procedure

project/procedureFunction에 위치. 프로시저 이름과 sql 파일명이 같지만, 다른 경우 별도로 적었다.

* DeleteBasket(): 아이디와 과목번호, 분반번호를 IN 파라미터로 받아 해당 수업을 장바구니 basket 테이블에서 지우는 프로시저
* deleteEnroll(): 아이디와 과목번호, 분반번호를 IN 파라미터로 받아 해당 수업을 enroll 테이블에서 지우고 그 결과 문자열을 OUT 파라미터로 내보내는 프로시저
* InsertBasket(): 아이디, 과목번호, 분반번호를 IN 파라미터로 받아 해당 수업을 basket 테이블에 insert하는 프로시저
* InsertEnroll(): 아이디, 과목번호, 분반번호를 IN 파라미터로 받아 해당 수업을 enroll 테이블에 추가하는 프로시저. 최대 학점을 초과하거나 동일한 과목이거나 수강 인원을 초과하거나 시간이 중복되면 오류를 일으킨다.
* selectTimeTable(): 아이디, 연도, 학기를 IN 파라미터로 받아 학생이 수강 신청한 과목 목록과 총 과목 수, 총 학점을 OUT 파라미터로 내보내는 프로시저. 과목 목록은 동적 커서로 내보낸다. project/procedureFunction/verify\_enroll.sql에 위치.
* selectTimeTable2(): 아이디, 연도, 학기를 IN 파라미터로 받아 교수님이 담당하는 강의 목록을 동적 커서 형태로 OUT 파라미터로 내보낸다.
* ShowBasket(): 아이디를 IN 파라미터로 받아 학생이 장바구니에 넣은 수업 목록을 동적 커서로 OUT 파라미터를 이용해 내보낸다.
* ShowBasketCandidate(): 아이디를 입력 받아 장바구니에 넣을 수 있는 수업 목록을 동적 커서로 OUT 파라미터를 이용해 내보낸다.

3) stored function

project/procedureFunction\function.sql에 있다.

* Date2EnrollYear(): 날짜를 IN 파라미터로 받아 해당하는 연도를 반환하는 함수. 리턴 타입은 NUMBER
* Date2EnrollSemester(): 날짜를 IN 파라미터로 받아 해당하는 학기를 반환하는 함수. 리턴 타입은 NUMBER. project\procedureFunction\function.sql에 위치

4) trigger

project/procedureFunction\trigger.sql에 있다.

* BeforeUpdateStudent: student 테이블을 감시하고 학생 패스워드를 수정하기 전에 패스워드가 4자리 이상인지, 공란은 없는지 확인하는 트리거.
* BeforeUpdateProfessor: professor 테이블을 감시하고 교수 패스워드를 수정하기 전에 패스워드가 4자리 이상인지, 공란은 없는지 확인하는 트리거.

<trigger.sql>

CREATE OR REPLACE TRIGGER BeforeUpdateStudent

BEFORE

UPDATE ON student

FOR EACH ROW

DECLARE

underflow\_length EXCEPTION;

invalid\_value EXCEPTION;

nLength NUMBER;

nBlank NUMBER;

BEGIN

SELECT length(:new.s\_pwd)

INTO nLength

FROM dual;

SELECT instr(:new.s\_pwd, ' ')

INTO nBlank

FROM dual;

IF (nLength < 4) THEN

RAISE underflow\_length;

ELSIF (nBlank >= 1) THEN

RAISE invalid\_value;

END IF;

EXCEPTION

WHEN underflow\_length THEN

RAISE\_APPLICATION\_ERROR(-20002, '암호는 4자리 이상이어야 합니다');

WHEN invalid\_value THEN

RAISE\_APPLICATION\_ERROR(-20003, '암호에 공란은 입력되지 않습니다');

END;

/

CREATE OR REPLACE TRIGGER BeforeUpdateProfessor

BEFORE

UPDATE ON professor

FOR EACH ROW

DECLARE

underflow\_length EXCEPTION;

invalid\_value EXCEPTION;

nLength NUMBER;

nBlank NUMBER;

BEGIN

SELECT length(:new.p\_pwd)

INTO nLength

FROM dual;

SELECT instr(:new.p\_pwd, ' ')

INTO nBlank

FROM dual;

IF (nLength < 4) THEN

RAISE underflow\_length;

ELSIF (nBlank >= 1) THEN

RAISE invalid\_value;

END IF;

EXCEPTION

WHEN underflow\_length THEN

RAISE\_APPLICATION\_ERROR(-20004, '암호는 4자리 이상이어야 합니다');

WHEN invalid\_value THEN

RAISE\_APPLICATION\_ERROR(-20005, '암호에 공란은 입력되지 않습니다');

END;

/

5) prepared statement

* check\_st\_verify.jsp 29번째 줄

교수님 버전) 선택한 강의의 정원을 수정하는 sql문을 실행

String sql = "update teach set t\_max = ? where c\_id = ? and c\_id\_no = ?"

PreparedStatement pstmt = conn.prepareStatement(sql);

pstmt.setString(1, t\_max); //희망 정원

pstmt.setString(2, c\_id); //과목번호

pstmt.setInt(3, c\_id\_no); //분반번호

* check\_st.jsp
  + 81번째 줄

교수님 버전) 선택한 강의의 수강 인원을 계산하기 위해 enroll 테이블에서 해당 강의를 신청한 학생들의 수를 count()한 sql문 실행

PreparedStatement pstmt = myConn.prepareStatement("select count(\*) s\_num from enroll where c\_id = ? and c\_id\_no = ?");

pstmt.setString(1, c\_id); //과목번호

pstmt.setInt(2, c\_id\_no); //분반번호

* 107번째 줄

교수님 버전) 선택한 강의의 수강생 학번과 전공을 선택. 서브쿼리로 enroll 테이블에서 해당 강의를 신청한 학생들의 학번을 select한 후 IN 연산자로 student 테이블에서 학번과 전공을 select하는 sql문 실행

sql = "select s.s\_id, s.s\_major from student s where s.s\_id in (select e.s\_id from enroll e where c\_id = ? and c\_id\_no = ?)"

PreparedStatement pstmt2 = myConn.prepareStatement(sql);

pstmt2.setString(1, c\_id); //과목번호

pstmt2.setInt(2, c\_id\_no); //분반번호

* find\_verify.jsp
* 31번째 줄

student 테이블에서 입력한 이름과 이메일에 맞는 회원의 아이디와 패스워드를 select하는 sql문 실행

String sql = "select s\_id, s\_pwd from student where s\_name = ? and s\_email = ?"

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setString(1, name); //이름

stmt.setString(2, email); //이메일

* 52번째 줄

student 테이블에서 입력한 이름과 이메일에 맞는 회원의 아이디와 패스워드를 select하는 sql문 실행

sql = "select p\_id, p\_pwd from professor where p\_name = ? and p\_email = ?";

stmt = conn.prepareStatement(sql);

stmt.setString(1, name); //이름

stmt.setString(2, email); //이메일

* joinProcess.jsp
  + 32번째 줄

교수님 회원가입. professor 테이블에 insert문 실행

pstmt=conn.prepareStatement("Insert into professor values(?,?,?,?,?)");

pstmt.setString(1,id); //아이디

pstmt.setString(2,pwd); //패스워드

pstmt.setString(3,name); //이름

pstmt.setString(4,major); //전공과

pstmt.setString(5,email); //이메일

* 40번째 줄

학생 회원가입. student 테이블에 insert문 실행

pstmt=conn.prepareStatement("Insert into student values(?,?,?,?,?,?,?)");

pstmt.setString(1,id); //아이디

pstmt.setString(2,pwd); //패스워드

pstmt.setString(3,name); //이름

pstmt.setString(4,major); //전공

pstmt.setString(5,sub); //복수/부/연계전공

pstmt.setString(6,email); //이메일

pstmt.setString(7,semester); //현재 학기

* update\_verify.jsp
  + 31번째 줄

학생 버전) 입력 받은 사용자 정보를 student 테이블에 update하는 sql문 실행

PreparedStatement stmt = myConn.prepareStatement("UPDATE student SET s\_id = ?, s\_pwd = ?, s\_major = ?, s\_sub = ?, s\_email = ?, s\_semester = ? WHERE s\_id = ?");

stmt.setString(1, userID); //아이디

stmt.setString(2, userPassword); //패스워드

stmt.setString(3, userMajor); //전공

stmt.setString(4, userSub); //복수/부/연계전공

stmt.setString(5, userEmail); //이메일

stmt.setString(6, userSemester); //현재 학기

stmt.setString(7, userID); //현재 아이디

* update.jsp 62번째 줄

학생 버전) 로그인한 아이디로 student 테이블에서 사용자 정보를 select하는 selec문 실행

mySQL = "select s\_id, s\_pwd, s\_major, s\_sub, s\_email, s\_semester from student where s\_id = ?";

pstmt = myConn.prepareStatement(mySQL);

pstmt.setString(1, session\_id); // 로그인한 아이디

* update2.jsp 69번째 줄

교수님 버전) 로그인한 아이디로 professor 테이블에서 사용자 정보를 select하는 sql문 실행

mySQL = "select p\_id, p\_pwd, p\_major, p\_email from professor where p\_id = ?";

pstmt = myConn.prepareStatement(mySQL);

pstmt.setString(1, session\_id); //로그인한 아이디

7) callable statement

* basket\_insert.jsp 64번째 줄

해당 연도와 학기에 아직 수강 신청하지 않은 수업 목록을 동적 커서로 반환하는 ShowBasketCandidate() 프로시저를 실행

CallableStatement cstmt = conn.prepareCall("{call ShowBasketCandidate(?,?)}");

cstmt.setString(1, s\_id); // 학생 아이디

cstmt.registerOutParameter(2, OracleTypes.CURSOR); //수강 신청하지 않은 수업들 커서

* basket\_show.jsp 61번째 줄

장바구니에 넣은 수업 정보를 보여주는 ShowBasket 프로시저를 실행

CallableStatement cstmt = conn.prepareCall("{call ShowBasket(?,?)}");

cstmt.setString(1, s\_id); //아이디

cstmt.registerOutParameter(2, OracleTypes.CURSOR); //커서

* basket\_verify\_enroll.jsp
  + 36번째 줄

장바구니에서 수강 신청할 수 있도록 선택한 과목을 수강 신청하는 프로시저 insertEnroll()을 실행

CallableStatement cstmt = conn.prepareCall("{call InsertEnroll(?,?,?,?)}");

cstmt.setString(1, s\_id); //아이디

cstmt.setString(2, sel\_c\_id); //선택한 수업 과목번호

cstmt.setInt(3, sel\_c\_id\_no); //선택한 수업 분반번호

cstmt.registerOutParameter(4, java.sql.Types.VARCHAR); //결과 문자열

* 48번째 줄

장바구니에서 수강 신청한 수업은 지워지도록 장바구니에서 선택한 과목을 장바구니에서 지우는 프로시저 DeleteBasket() 실행

cstmt = conn.prepareCall("{call DeleteBasket(?, ?, ?)}");

cstmt.setString(1, s\_id); //아이디

cstmt.setString(2, sel\_c\_id); //수강 신청한 과목번호

cstmt.setInt(3, sel\_c\_id\_no); //수강 신청한 분반번호

* 78번째 줄

선택한 과목을 장바구니에서 해제하기 위해 프로시저 DeleteBasket()을 실행

CallableStatement cstmt = conn.prepareCall("{call DeleteBasket(?, ?, ?)}");

cstmt.setString(1, s\_id); //아이디

cstmt.setString(2, del\_c\_id); //해제 선택한 과목번호

cstmt.setInt(3, del\_c\_id\_no); //해제 선택한 분반번호

* basket\_verify.jsp 32번째 줄

선택한 과목을 장바구니에 추가하는 프로시저 InsertBasket() 실행

CallableStatement cstmt = conn.prepareCall("{call InsertBasket(?, ?, ?)}");

cstmt.setString(1, s\_id); //아이디

cstmt.setString(2, sel\_c\_id); //과목번호

cstmt.setInt(3, sel\_c\_id\_no); //분반번호

* delete\_verify.jsp 22번째 줄

선택한 과목을 수강 신청 취소하는 프로시저 deleteEnroll() 실행

CallableStatement cstmt = myConn.prepareCall("{call deleteEnroll(?,?,?,?)}");

cstmt.setString(1, s\_id); //아이디

cstmt.setString(2, c\_id); //과목번호

cstmt.setInt(3, c\_id\_no); //분반번호

cstmt.registerOutParameter(4, java.sql.Types.VARCHAR); //결과 문자열

* insert\_verify.jsp 22번째 줄

선택한 과목을 수강 신청하는 프로시저 InsertEnroll() 실행

CallableStatement cstmt = myConn.prepareCall("{call InsertEnroll(?,?,?,?)}");

cstmt.setString(1, s\_id); //아이디

cstmt.setString(2, c\_id); //과목번호

cstmt.setInt(3, c\_id\_no); //분반번호

cstmt.registerOutParameter(4, java.sql.Types.VARCHAR); //결과 문자열

* insert.jsp
  + 54번째 줄

해당 연도를 반환하는 함수 Date2EnrollYear() 실행

String sql = "{? = call Date2EnrollYear(sysdate)}" //현재 날짜를 입력 매개변수로

CallableStatement yStmt = myConn.prepareCall(sql);

yStmt.registerOutParameter(1, java.sql.Types.INTEGER); //결과 연도

* 60번째 줄

해당 학기를 반환하는 함수 Date2EnrollSemester() 실행

String sql2 = "{? = call Date2EnrollSemester(sysdate)}" //현재 날짜를 입력 매개변수로

CallableStatement eStmt = myConn.prepareCall(sql2);

eStmt.registerOutParameter(1, java.sql.Types.INTEGER); //결과 학기

* select.jsp
  + 67번째 줄

해당 연도를 반환하는 함수 Date2EnrollYear() 실행

String sql = "{? = call Date2EnrollYear(sysdate)}" //현재 날짜를 입력 매개변수로

CallableStatement yStmt = myConn.prepareCall(sql);

yStmt.registerOutParameter(1, java.sql.Types.INTEGER); //결과 연도

* 72번째 줄

해당 학기를 반환하는 함수 Date2EnrollSemester() 실행

String sql2 = "{? = call Date2EnrollSemester(sysdate)}" //현재 날짜를 입력 매개변수로

CallableStatement eStmt = myConn.prepareCall(sql2);

eStmt.registerOutParameter(1, java.sql.Types.INTEGER); //결과 학기

* 78번째 줄

위에서 받아 온 해당 연도, 학기와 아이디를 받아 학생이 수강 신청한 목록을 동적 커서로 반환하는 프로시저 selectTimeTable() 실행

CallableStatement cstmt = myConn.prepareCall("{call selectTimeTable(?,?,?,?,?,?)}");

cstmt.setString(1, s\_id); //아이디

cstmt.setInt(2, nYear); //해당 연도

cstmt.setInt(3, nSemester); //해당 학기

cstmt.registerOutParameter(4, java.sql.Types.INTEGER); //총 과목 수

cstmt.registerOutParameter(5, java.sql.Types.INTEGER); //총 학점

cstmt.registerOutParameter(6, OracleTypes.CURSOR); //커서

* select2.jsp
  + 65번째 줄

교수님 버전) 해당 연도를 반환하는 함수 Date2EnrollYear() 실행

String sql = "{? = call Date2EnrollYear(sysdate)}" //현재 날짜를 입력 매개변수로

CallableStatement yStmt = myConn.prepareCall(sql);

yStmt.registerOutParameter(1, java.sql.Types.INTEGER); //결과 연도

* 71번째 줄

교수님 버전) 해당 학기를 반환하는 함수 Date2EnrollSemester() 실행

String sql2 = "{? = call Date2EnrollSemester(sysdate)}" //현재 날짜를 입력 매개변수로

CallableStatement eStmt = myConn.prepareCall(sql2);

eStmt.registerOutParameter(1, java.sql.Types.INTEGER); //결과 학기

* 77번째 줄

교수님 버전) 위에서 받아 온 해당 연도, 학기와 아이디를 받아 교수님이 담당하는 강의 목록을 동적 커서로 반환하는 프로시저 selectTimeTable2() 실행

CallableStatement cstmt = myConn.prepareCall("{call selectTimeTable2(?,?,?,?)}");

cstmt.setString(1, p\_id); //아이디

cstmt.setInt(2, nYear); // 연도

cstmt.setInt(3, nSemester); //학기

cstmt.registerOutParameter(4, OracleTypes.CURSOR); //담당 강의 커서

3. 웹 화면 구성과 자바 소스 관계

1. 요약
   1. main.jsp - 첫 화면이자 학생용 메인 화면. top.jsp를 포함한다.
   2. main2.jsp - 교수 전용 메인 화면. 교수님 계정으로 로그인 시 메인 화면이다. top2.jsp를 포함한다.
   3. top.jsp - 학생용 메뉴. 로그아웃/사용자 정보 수정/수강신청 입력/수강신청 조회/회원가입/장바구니
   4. top2.jsp - 교수님 메뉴 로그아웃/사용자 정보 수정/강의목록 조회
   5. logout.jsp - 로그아웃
   6. login.jsp - 로그인 화면. 아이디와 패스워드를 입력 받아 login\_verify.jsp로 넘긴다.
   7. login\_verify.jsp - login.jsp에서 받은 파라미터로 로그인을 처리한다.
   8. find.jsp - 아이디/비밀번호 찾기 화면. 이름과 이메일을 입력 받아 find\_verify.jsp로 넘긴다.
   9. find\_verify.jsp - find.jsp에서 받은 파라미터로 아이디와 비밀번호를 찾아 출력한다.
   10. update.jsp - 학생용 사용자 정보 수정 화면. 입력 받은 값을 update\_verify.jsp로 넘긴다.
   11. update\_verify.jsp - 학생용 사용자 정보 수정 처리. update.jsp에서 넘겨 받은 파라미터로 student 테이블을 update한다.
   12. updat2.jsp - 교수용 사용자 정보 수정 화면. 입력 받은 값을 update\_verify2.jsp에 넘긴다.
   13. update\_verify2.jsp - 교수용 사용자 정보 수정 처리. update2.jsp에서 넘겨 받은 파라미터로 professor 테이블을 update한다.
   14. insert.jsp - 수강신청 입력 화면. 현재 날짜에 맞는 연도, 학기 수업 중 아직 수강 신청하지 않은 수업을 출력한다. 신청 버튼을 누르면 insert\_verify.jsp로 값을 넘긴다.
   15. insert\_verify.jsp - 수강신청 입력 처리. insert.jsp에서 넘겨 받은 파라미터로 enroll 테이블에 insert한다.
   16. select.jsp - 수강신청 조회 화면. 수강 신청한 수업 목록과 총 과목 수, 총 학점을 출력한다. 신청 취소를 누르면 delete\_verify.jsp로 값을 넘긴다.
   17. delete\_verify.jsp - 수강신청 취소 처리. select.jsp에서 넘겨 받은 값으로 enroll 테이블에서 delete한다.
   18. select2.jsp - 교수용 강의 목록 조회 화면. 담당하는 강의 목록을 출력한다. ‘수강생’ 열의 ‘확인’을 누르면 chekc\_st.jsp로 연결돼 해당 강의 수강생을 확인할 수 있다.
   19. check\_st.jsp - 강의 수강생 확인과 정원 수정 화면. 정원 수정 ‘변경’ 버튼을 누르면 check\_st\_verify.jsp로 값을 넘긴다.
   20. check\_st\_verify.jsp - check\_st.jsp에서 넘겨 받은 값으로 해당 강의 정원을 수정한다.
   21. joinForm..jsp - 회원가입 화면. 입력 받은 값을 joinProcess.jsp로 넘긴다.
   22. joinProcess.jsp - 아이디 길이로 학생과 교수를 구분해 각각 student, professor에 insert한다.
   23. basket.jsp - 장바구니 화면. 장바구니 추가/장바구니 조회 메뉴가 있다.
   24. basket.insert.jsp - 장바구니 추가 화면. 현재 날짜에 맞는 연도, 학기의 수업 중 수강 신청하지 않은 수업 목록을 출력한다. ‘저장’ 버튼을 누르면 basket\_verify.jsp로 값을 넘긴다.
   25. basket\_verify.jsp - 장바구니 추가 처리. basket.jsp에서 넘겨 받은 파라미터로 basket 테이블에 insert한다.
   26. basket\_show.jsp - 장바구니 조회 화면. 수강 신청과 장바구니 해제를 할 수 있다. ‘저장’ 버튼을 누르면 basket\_verify\_enroll.jsp로 값을 넘긴다.
   27. basket\_verify\_enroll.jsp - 장바구니에서 수강 신청과 장바구니 해제 처리. basket\_show.jsp에서 넘겨 받은 값으로 선택한 과목을 수강신청하거나 장바구니에서 해제한다.
2. 첫 화면 main.jsp



1. 학생
   1. 로그인 화면 login.jsp



처음 로그인 화면 login.jsp이다.

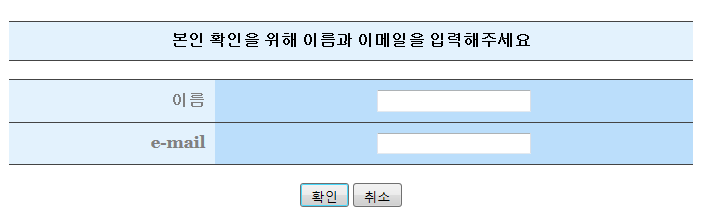
아이디는 학번 혹은 교번으로 하기 때문에 아이디의 길이로 사용자를 구분한다. 학생이 로그인하면 학생 전용 화면으로 전환한다는 알림창이 뜨고 첫 화면이자 학생용 화면이기도 한 main.jsp로 연결된다.

‘취소’ 버튼을 누르면 아이디, 패스워드 필드에 입력한 값이 지워진다.

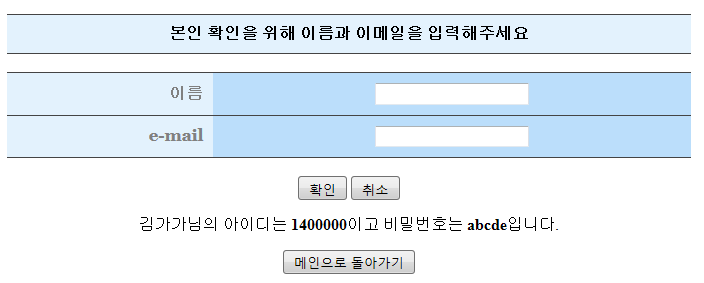
‘메인으로’ 버튼을 누르면 ‘메인화면으로 돌아가기’라는 알림창이 뜬 후 main.jsp로 연결된다.

‘회원가입’은 joinForm.jsp로 연결된다.

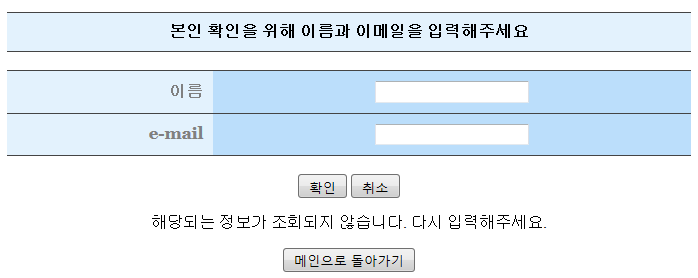
‘아이디/비밀번호 찾기’를 누르면 find.jsp로 연결되고 화면은 아래와 같다.



이름과 이메일 입력 후 ‘확인’ 버튼을 누르면 해당하는 아이디와 패스워드를 출력한다.



만약 존재하지 않는 이름과 이메일이라면 다음과 같이 출력된다.



‘메인으로 돌아가기’ 버튼을 누르면 main.jsp로 연결된다.



로그인 후 main.jsp이다. 학생용인 top.jsp를 포함하고 있다.

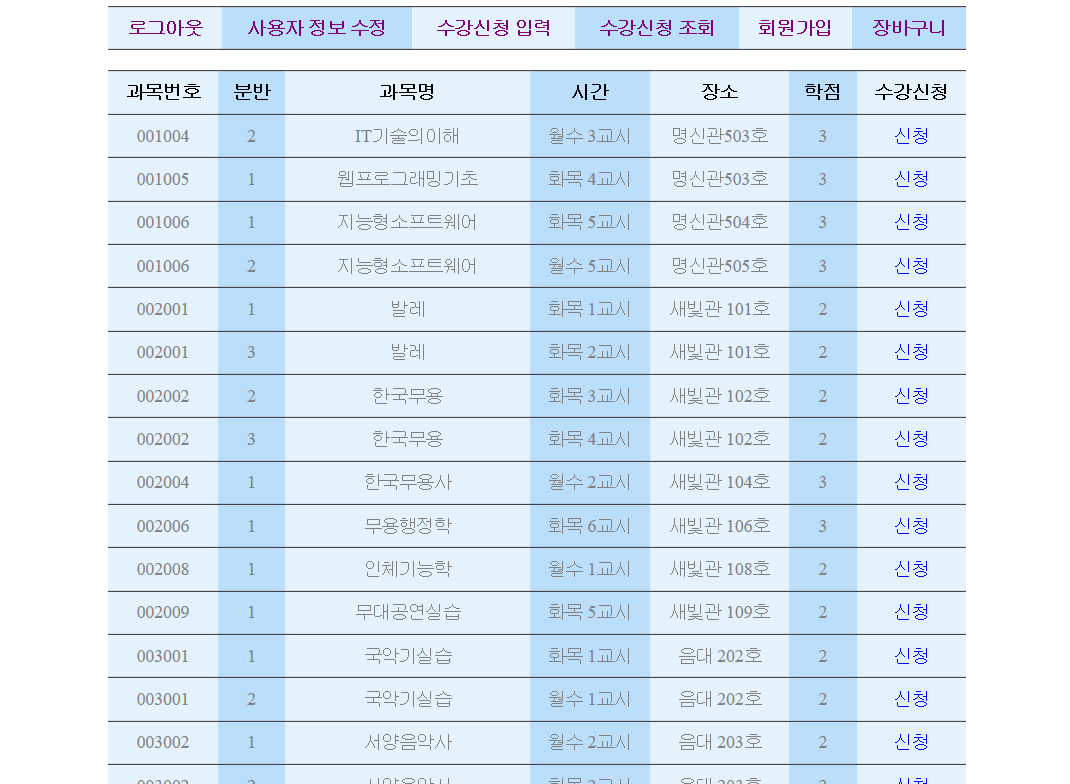
* 1. 사용자 정보 수정 update.jsp



사용자 정보 수정 메뉴를 클릭하면 update.jsp로 연결된다. 정보를 수정 후 ‘저장’ 버튼을 누르면 ‘수정되었습니다’ 알림창이 뜬다. 또한 ‘메인으로’ 버튼을 누르면 ‘메인화면으로 돌아가기’ 알림창이 뜨면서 학생 전용 메인 화면 main.jsp로 연결된다.

패스워드가 4자리 미만이면 ‘암호는 4자리 이상이어야 합니다.’ 알림창이 뜨고 패스워드에 공란이 있으면 ‘암호에 공란은 입력되지 않습니다.’라는 알림창이 뜨고 다시 update.jsp로 연결된다.

* 1. 수강신청 입력 insert.jsp



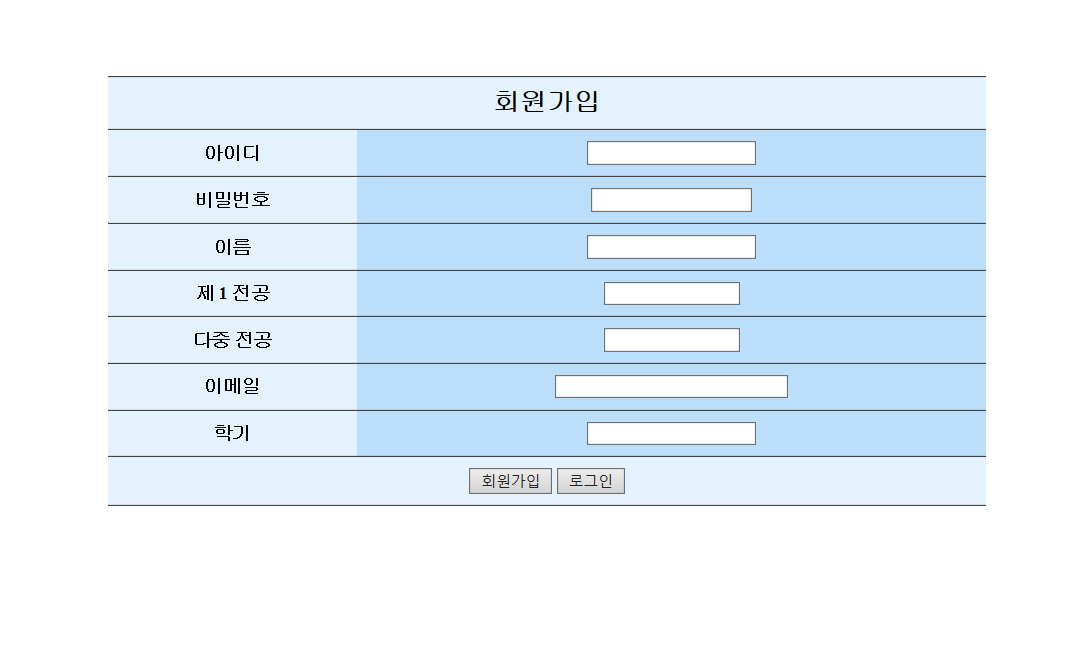
표의 수강신청 열인 ‘신청’을 누르면 insert\_verify.jsp에서 수강 신청을 처리한다. 수강 신청 시 오류가 나면 알림창을 띄운다.

* 1. 수강신청 조회 select.jsp



신청 취소를 누르면 delete\_verify.jsp에서 수강 신청 취소를 처리하고 ‘수강신청 변경이 완료되었습니다.’라는 알림창을 띄운 후 다시 select.jsp로 돌아온다.

* 1. 회원가입 joinForm.jsp

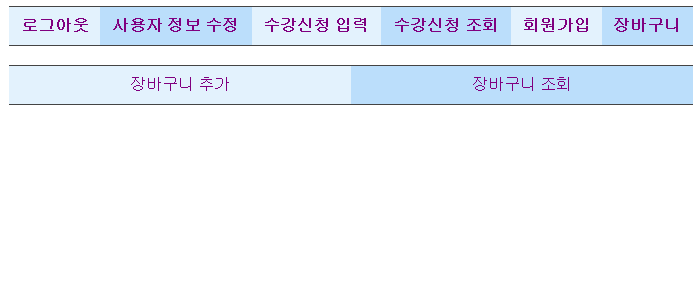


‘회원가입’ 버튼을 누르면 입력한 정보를 joinProcess.jsp가 받아 처리한다. joinProcess.jsp에서는 아이디의 길이로 학생/교수님을 구분해 student 테이블이나 professor 테이블에 새로운 행을 insert한다.

만약 이미 존재하는 회원이라면 ‘이미 존재하는 회원입니다.’라는 알림창을 띄우지만 정상적으로 회원가입이 완료되면 ‘회원가입이 완료되었습니다.’라는 알림창을 띄운다.

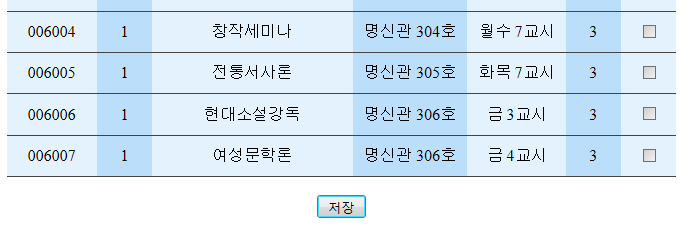
‘로그인’ 버튼을 누르면 ‘로그인 화면으로 돌아가기’라는 알림창이 뜬 후 login.jsp로 연결된다.

* 1. 장바구니 basket.jsp



장바구니 메뉴를 클릭하면 basket.jsp로 연결된다.



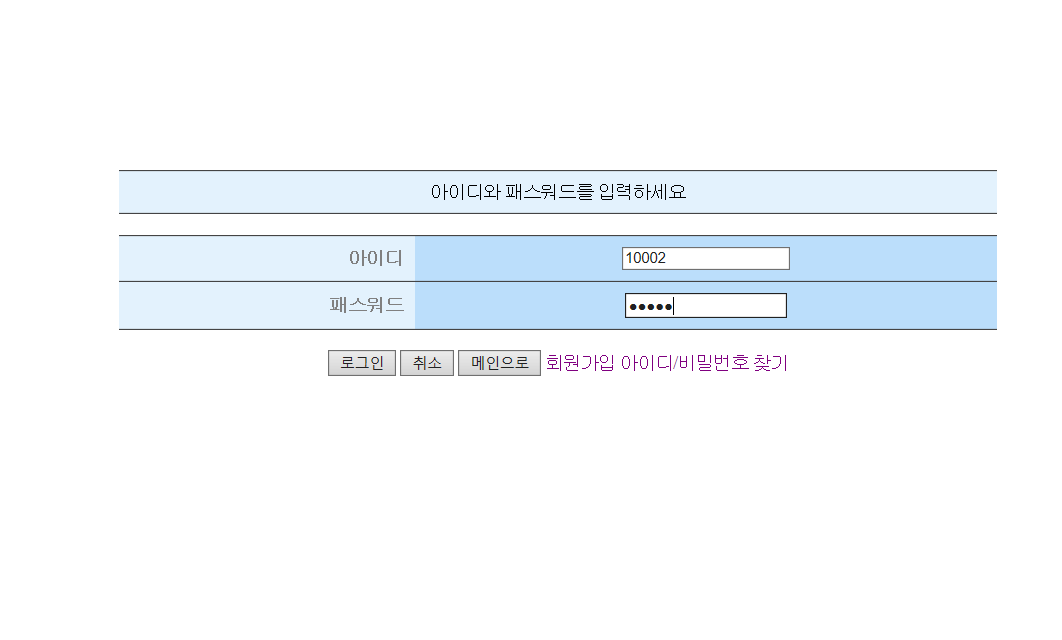


‘장바구니 추가’를 선택하면 basket\_insert.jsp로 연결된다. bakset\_insert.jsp는 basket.jsp를 포함한다. 체크박스로 과목을 다중 선택해 아래에 있는 저장 버튼을 누르면 basket\_verify.jsp가 장바구니 추가를 처리한다.



‘장바구니 조회’를 선택하면 basket\_show.jsp로 연결된다. 마찬가지로 bakset.jsp를 포함한다. 체크박스 다중선택으로 장바구니에서 바로 수강신청을 할 수도 있고 장바구니에서 과목을 해제할 수도 있다. 수강 신청 시 결과에 따른 알림창이 뜨고 장바구니 해제 시 ‘해제되었습니다.’ 알림창이 뜬다.

1. 교수님



교수님으로 로그인 한다. 교번과 패스워드를 입력한다.

* 1. 메인 main2.jsp



‘교수 전용 화면을 전환합니다.’라는 알림창으 뜨고 교수님 전용인 main2.jsp로 연결된다. 교수님용 top2.jsp를 포함한다.

* 1. 사용자 정보 수정 update2.jsp



교수 사용자 정보를 수정할 수 있는 update2.jsp이다. 기능은 학생 사용자 정보 수정과 동일하다.

* 1. 강의 목록 조회 select2.jsp



강의 목록 조회를 선택하면 담당하는 강의 목록을 출력하는 select2.jsp로 연결된다. ‘수강생’ 열의 ‘확인하기’를 클릭하면 수강생을 확인하고 정원을 수정할 수 있는 chekc\_st.jsp로 연결된다.



check\_st.jsp화면이다. 정원 수정 후 ‘변경’ 버튼을 클릭하면 check\_st\_verify.jsp가 이를 받아 처리 후 ‘정원이 변경되었습니다.’ 알림창을 띄운다. 그 다음 다시 select2.jsp로 연결된다.

1. 로그아웃

‘로그아웃 되었습니다.’ 알림창이 뜨고 main.jsp로 연결된다.

4. 기타

1) css를 이용한 전체를 관통하는 UI제작 코드:

<style type="text/css">

<style>

table {

width: 80%;

border-top: 1px solid #444444;

border-collapse: collapse;

border-left:none;

border-right:none;

}

a { text-decoration:none } //하이퍼링크의 모든 밑줄을 지움

th, td {

border-bottom: 1px solid #444444;

padding: 10px;

text-align: center;

border-left:none;

border-right:none;

}

th:nth-child(2n), td:nth-child(2n) {

background-color: #bbdefb; //짝수열에 적용되는 색깔

}

th:nth-child(2n+1), td:nth-child(2n+1) {

background-color: #e3f2fd;//홀수열에 적용되는 색깔 구분

}

</style>

5. 조원 구성원과 분담 내역

1. 조 이름: 종강시켜조
2. 조원: 김윤아, 원효선
3. 분담 내역
   1. 데이터베이스 설계: 김윤아, 원효선
   2. 데이터 테이블 작성
      1. student 테이블: 김윤아
      2. basket, course, enroll, professor, teach 테이블: 김윤아, 원효선
   3. 프로시저/함수
      1. 김윤아: deleteEnroll, selectTimeTable2
      2. 원효선: DeleteBasket, Date2EnrollYear, Date2EnrollSemester, InsertBasket, InsertEnroll, SelectTimeTable, ShowBasket, ShowBasketCandidate

* 1. 트리거
     1. 원효선: BeforeUpdateStudent, BeforeUpdateProfessor
  2. jsp
     1. 김윤아, 원효선: check\_st.jsp, login.jsp, main.jsp, select.jsp, top.jsp, update\_verify.jsp, update.jsp
     2. 김윤아: delete\_verify.jsp, insert\_verify.jsp, insert.jsp, joniForm.jsp, joinProcess.jsp, login\_verify.jsp, update\_verify2.jsp, update2.jsp
     3. 원효선: basket\_insert.jsp, basket\_show.jsp, basket\_verify\_enroll.jsp, basket\_verify.jsp, chekc\_st\_verify.jsp, find\_verifiy.jsp, find.jsp, logout.jsp
  3. 모든 디자인: 김윤아