2021

{노력을 멈추지 않는
개발자가 되겠습니다.}

윤종무

프로젝트 소개

진행한 프로젝트에서
제가 맡은 부분의 일부를 설명하였습니다.

KOSTA 학습관리시스템

https://github.com/buchonsi/kosta_final_project

- MVC 모델 기반(Spring Boot + MySQL) 수강신청 시스템
- 사용자는 수업을 검색하고 수강신청을 할 수 있습니다.
- 관리자는 수업정보를 저장하고 회원관리를 할 수 있습니다.
- KOSTA EDU 페이지와 관리자 페이지를 구현하였습니다.
- 실제 근무하시고 있는 연구원님의 요구사항을 토대로 프로젝트를 진행하였습니다.

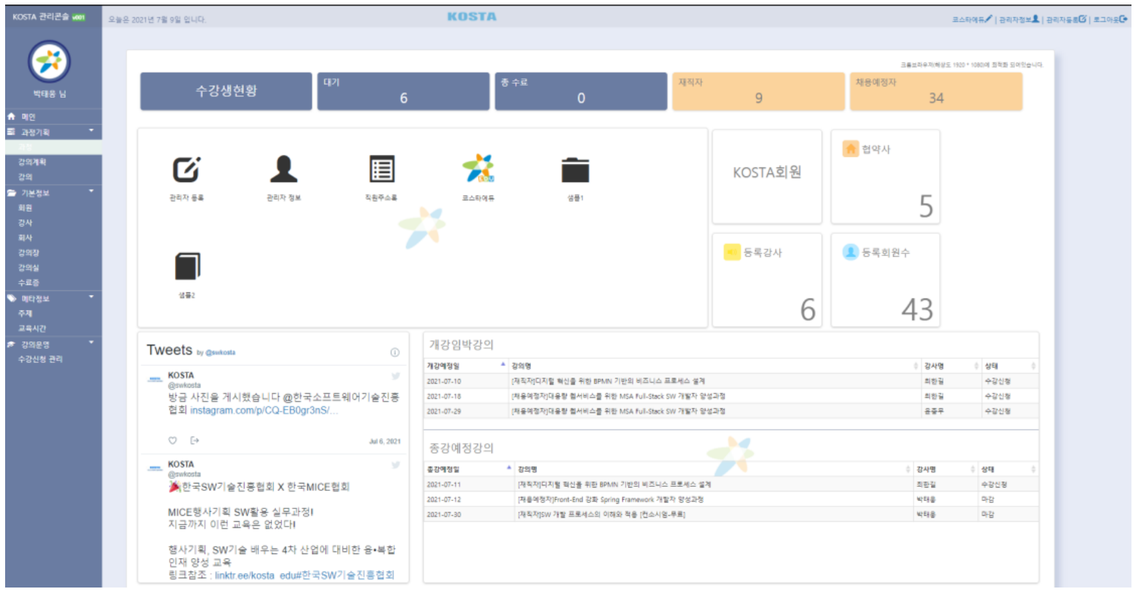
2021.06.01 - 2021.06.30 (4주)

윤종무, 김현유, 최한길, 박태웅

사용 기술

- Java, Spring Boot, Spring Security,
- JPA(Java Persistent API), Hibernate,
- JavaScript, Ajax,
- jQuery, Thymeleaf,
- MariaDB,
- AWS EC2, RDS
- GitHub

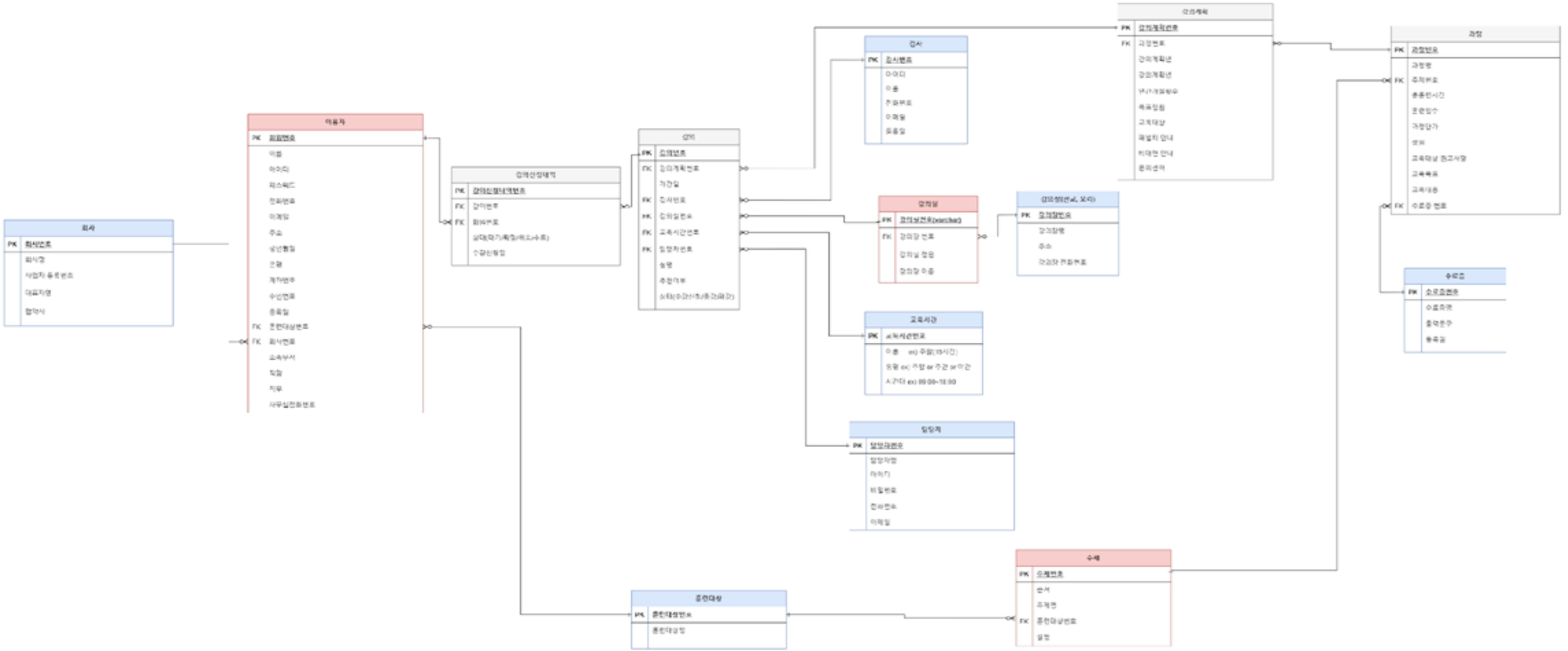
<사용자 페이지>



<관리자 페이지>

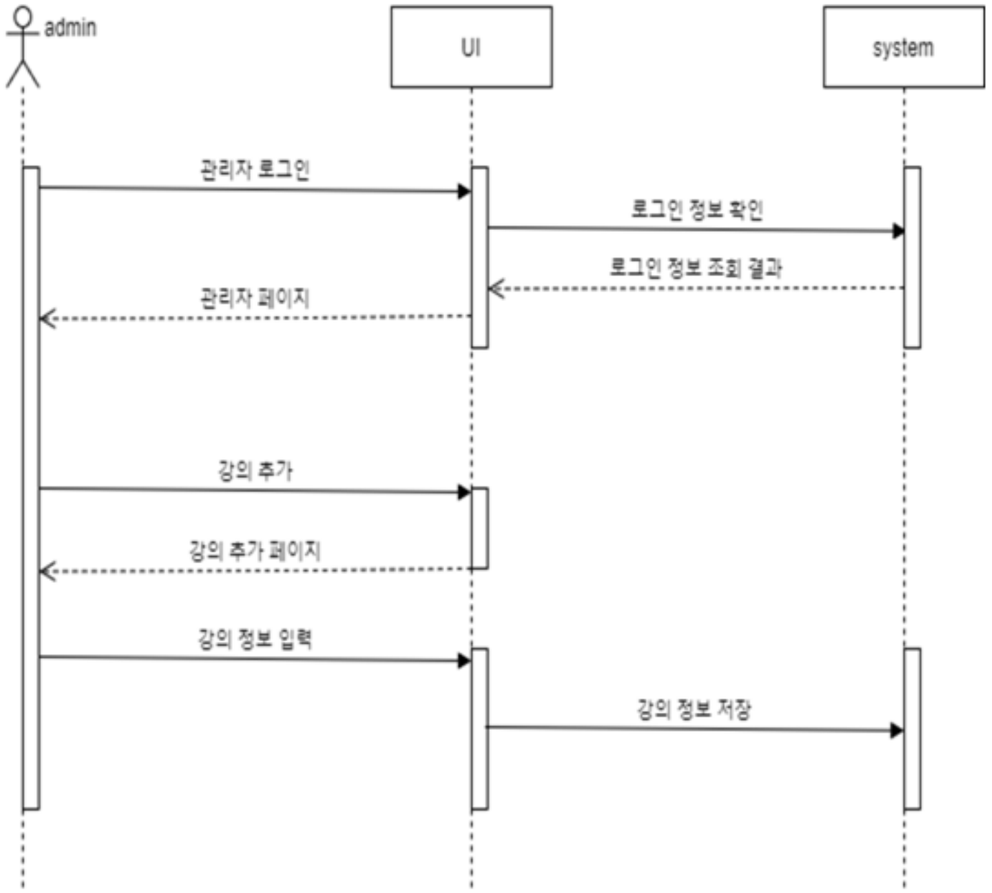
KOSTA 학습관리시스템-ERD

논리적 데이터 모델링

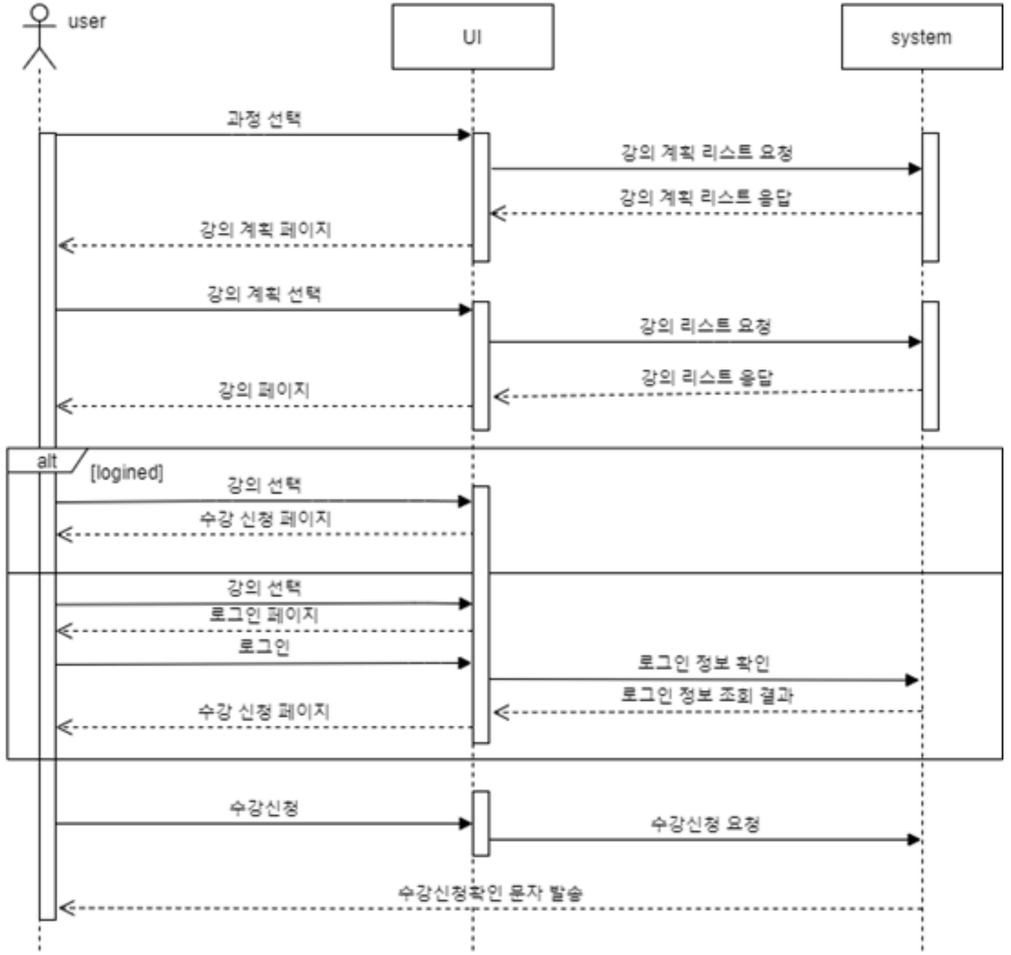


KOSTA 학습관리시스템-Sequence Diagram

강의 추가



수강신청



KOSTA 학습관리시스템-JPA

```java
public interface CourseRepository extends CrudRepository<Course, Long>, QuerydslPredicateExecutor<Course>{

    @Query(value="select c.courseName, l.lectureOpenCount, c.courseTotalTrainTime, c.courseNo, l.lecturePlanYear "
            + "from Course c left outer join Lecture l on (l.course = c.courseNo) "
            + "where c.subject = ?1 AND l.lecturePlanYear=YEAR(CURDATE())")
    public List<Object[]> getCourseWithLecture(Subject subjectNo);
```

```java
//Querydsl은 처음 2레벨의 깊이까지만 초기화한다.
//더 깊은 경로로 초기화 해야한다면 @QueryInit으로 초기화 시켜줘야한다.
@ManyToOne
@QueryInit("course.*")
private Lecture lecture;
@ManyToOne
private Teacher teacher;

//Querydsl은 처음 2레벨의 깊이까지만 초기화한다.
//더 깊은 경로로 초기화 해야한다면 @QueryInit으로 초기화 시켜줘야한다.
@ManyToOne
@QueryInit("lectureHall.*")
private ClassRoom classRoom;
@ManyToOne
private EducationTime educationTime;
@ManyToOne
private Admin admin;
```

```java
public default Predicate makePredicateSubHallClasses(String keyword, Long subNo, Long lecHallNo) {
    BooleanBuilder builder = new BooleanBuilder();
    QClasses classes = QClasses.classes;
    // 강사명
    builder.or(classes.teacher.teacherName.contains(keyword));
    // 과정명
    builder.or(classes.lecture.course.courseName.contains(keyword));
    // 주제
    builder.or(classes.lecture.course.subject.subName.contains(keyword));

    //과정명과 주제의 classes로의 깊이가 2단계 이상이기 때문에 @QueryInit을 사용해야 한다.

    if(subNo != null) {
        builder.and(classes.lecture.course.subject.subjectNo.eq(subNo));
    }
    //강의장
    if(lecHallNo != null) {
        builder.and(classes.classRoom.lectureHall.lectureHallNo.eq(lecHallNo));
    }
    return builder;
}
```

- 조회를 하는 과정에서 Querydsl을 사용했습니다.
- Querydsl은 처음 2레벨의 깊이까지만 초기화합니다.
- 더 깊은 경로를 조회해야 한다면 해당 Column에 QueryInit으로 초기화 시켜줘야 합니다.

KOSTA 학습관리시스템-메뉴바 생성

<userMain.html>

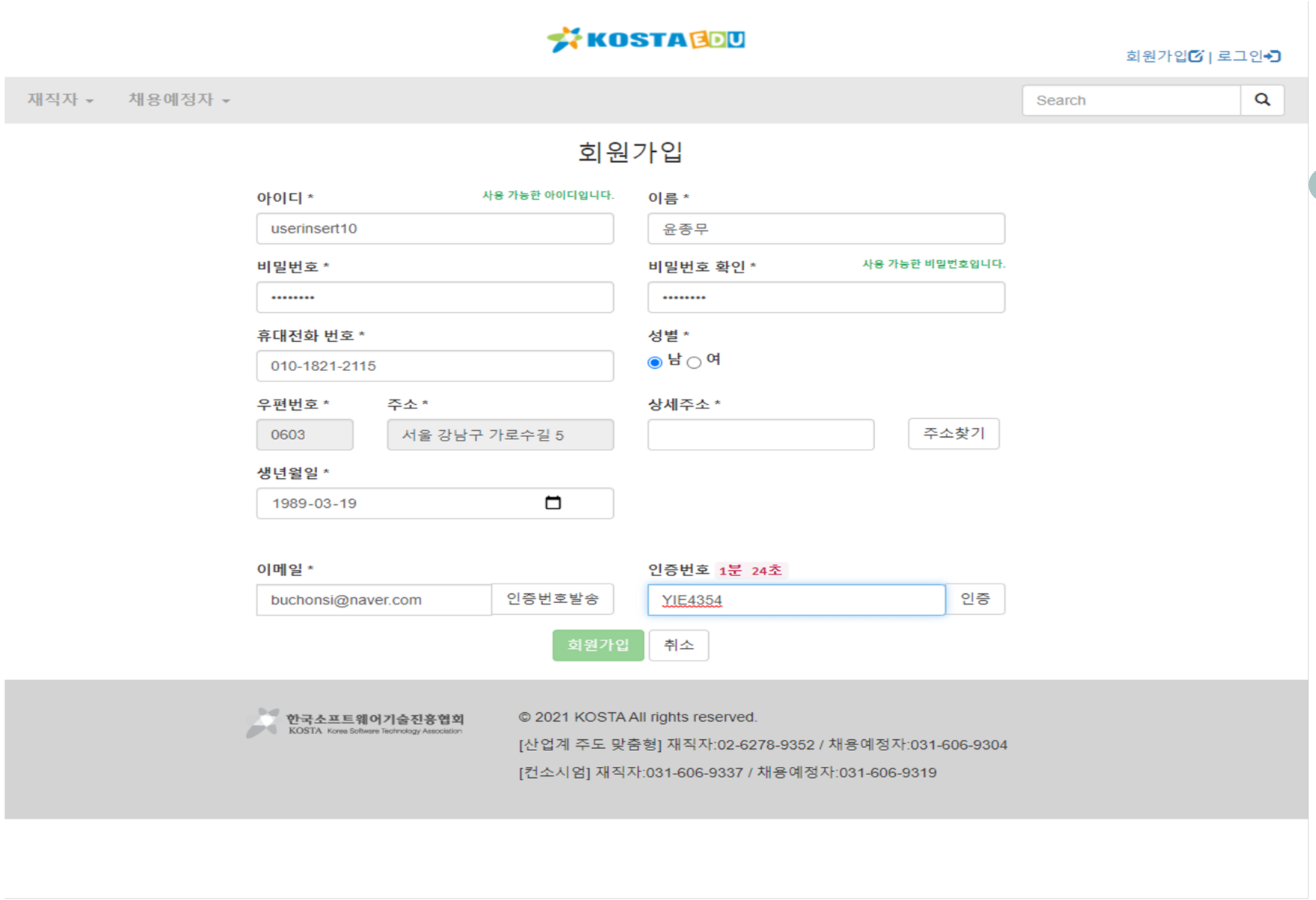
```
$.ajax({
     url:"/fragments/headerUser",
     type:"get",
     success:printList
   });
function printList(trainee){
  var str="";
  $.each(trainee, function(idx,trainee){
    str += `<div id="traineeBox${idx}" class="row kt-p-25 border-top">
              <h3 class="topic-title"><b>${trainee.traineeName}</b></h3>
              <div class="col-md-12 kt-plr-50" id="subjectBox${trainee.traineeNo}">
              </div>
            </div>
          </div>`;

    $("#topics").html(str);
    subjectList(trainee);
  });
}

function subjectList(trainee){
  var str="";
  $.getJSON("/user/userMain/"+trainee.traineeNo,function(subject){
    $.each(subject,function(idx,subject){
       str += `<div class="col-md-3">
                 <h4 class="font-light topic-sub-title line-height-1">
                    <a href='/courseInfo/${subject.subjectNo}'>${subject.subName}</a>
                 </h4>
               </div>`;
    });
    $("#subjectBox"+trainee.traineeNo).html(str);
  });
```

<Controller>

```
//헤더
@ResponseBody
@GetMapping("/fragments/headerUser")
public List<Trainee> userHeader() {
    List<Trainee> traineeList = cservice.findTraineeAll();
    return traineeList;
}

public List<Trainee> findTraineeAll() {
    return (List<Trainee>)traineeRepo.findAll();
}
```

- 메뉴바의 대분류는 추가/삭제가 가능하기 때문에 AJAX를 이용하여 DB에 저장된 정보에 따라 메뉴바 구성이 달라지는 방법을 사용하였습니다.

KOSTA 학습관리시스템 – 사용자 페이지 로그인



- 아이디, 패스워드 , 전화번호, 주소, 이메일에 필요한 validation은 자바스크립트로 처리 하였습니다.

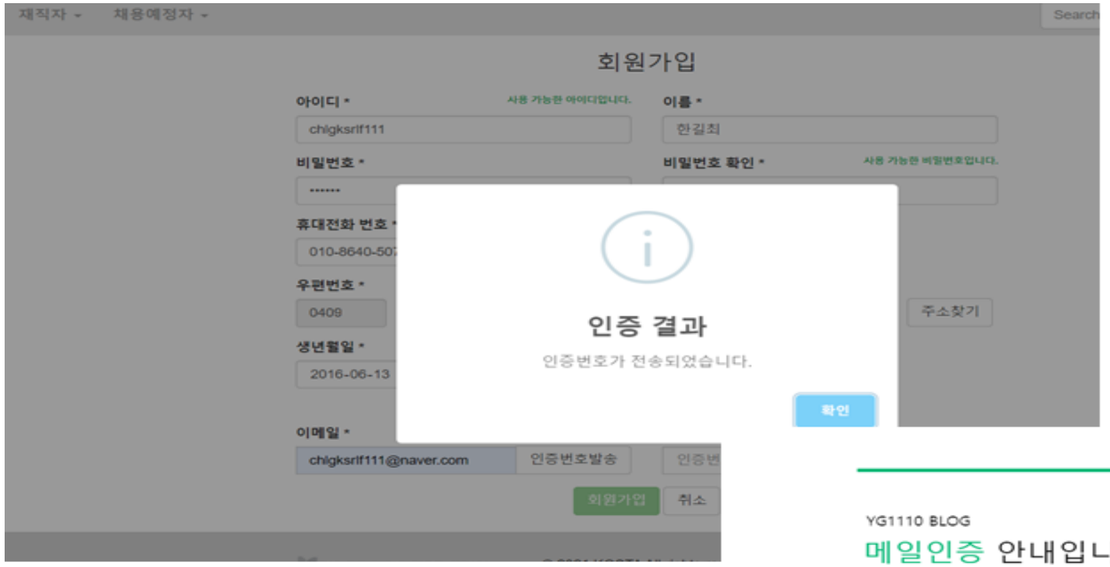
```
$(function(){
        //값 초기화, 회원가입 버튼 비활성화
        $("#insertBtn").prop("disabled",true);
        var idOk = false;
        var pw1Ok = false;
        var pw2Ok = false;
        var emailOk = false;
        var isOk = false;

        //주소 검색
        $("#searchAddress").click(function(){
            addressSearch();
        });

        //정규식 활용 id 체크(영소문자+숫자 5자 이상)----------------
        $("#inputUserId").keyup(function(){

            //배열 초기화
            var viewData = {};
            var inputTxt = $("#inputUserId").val();
            //data[키] = 밸류
            viewData["userId"] = inputTxt;
            var chkOnlyEngNum = /^[a-z0-9_-]{4,20}$/;

            if( chkOnlyEngNum.test(inputTxt) && inputTxt.length > 3){

                //정규식 조건을 통과한 경우
                //$("#signupWarn").text( '   ' );
                //초기값 valid 적용
                $("#idEnable").attr("class","is-invalid");
                $("#idDisable").attr("class","is-invalid");
                $.ajax({
                    type: "POST"
                    //동기식으로 변경 >> ajax에서 리턴값을 이용하기 위해
                    , async: false
                    , contentType:"application/json"
                    , url: "/user/userIdChk"
                    , datatype: "json"
                    , data: JSON.stringify(viewData)
                    , success: function(data){
                        if( data == 0 ){
                            //컨트롤러에서 넘어온 결과가 0인 경우 사용 가능한 아이디로 판단
                            //alert('ok data 트루라고 했다');
                            $("#idEnable").removeClass("is-invalid");
                            $("#idEnable").addClass("is-valid");
                            idOk = true;
                            //$("#signupWarn").text( '   ' );
                            isOk = allOk( idOk, pw1Ok, pw2Ok, emailOk );
                            if( isOk == true ){
                                $("#insertBtn").prop("disabled",false);
                            }
                            //$("#chkAllOk").text(idOk + ', ' + pw1Ok + ', ' + pw2Ok +  ', ' + emailOk + '마지막 idOk:' + isOk );
                        }else{
                            //중복된 ID인 경우
                            $("#idDisable").removeClass("is-invalid");
                            $("#idDisable").addClass("is-valid");
                            idOk = false;
                            //$("#chkAllOk").text(idOk + ', ' + pw1Ok + ', ' + pw2Ok + ', ' + emailOk + '마지막 idOk:' + isOk );
                            $("#idDisable").text( '중복된 ID입니다. 다른 ID를 입력해 주세요.' );
                            $("#btnConfirm").prop("disabled",true);
                        }
                    }, error: function( error ){
                        alert('error : ' + error);
                    }
                });
            }else{
                //조건에 맞지 않는 경우
                idOk = false;
                $("#insertBtn").prop("disabled",true);
                $("#idEnable").attr("class","is-invalid");
                $("#idDisable").text("4~20자 소문자 또는 숫자를 넣어주세요");
                $("#idDisable").attr("class","is-valid");
            }
        });
```

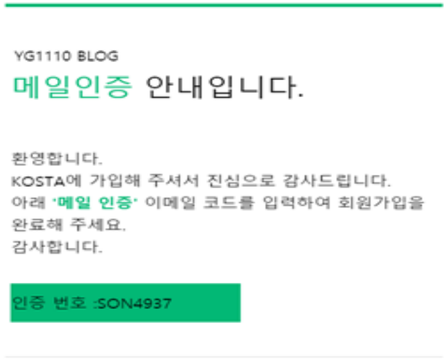
```
//pw 체크----------------
$("#confirmPassword").keyup(function(){
    var pw1 = $("#newPassword").val();
    pw1Ok = true;
    var pw2 = $("#confirmPassword").val();
    // : 숫자, 특문 각 1회 이상, 영문은 2개 이상 사용하여 8자리 이상 입력
    //var regExpPw = /(?=.*\d{1,50})(?=.*[~`!@#$%\^&*()-+=]{1,50})(?=.*[a-zA-Z]{2,50}).{8,50}$/;
    if( pw1 == pw2 ){
        pw2Ok = true;
        //$("#signupWarn").text( '   ' );
        isOk = allOk( idOk, pw1Ok, pw2Ok, emailOk );
        if( isOk == true ){
            $("#insertBtn").prop("disabled",false);
        }
        //$("#chkAllOk").text(idOk + ', ' + pw1Ok + ', ' + pw2Ok + ', '  + emailOk + '마지막 idOk:' + isOk );
        $("#pwDisable").removeClass("is-valid");
        $("#pwDisable").addClass("is-invalid");
        $("#pwEnable").removeClass("is-invalid");
        $("#pwEnable").addClass("is-valid");

    }else{
        pw2Ok = false;
        $("#insertBtn").prop("disabled",true);
        $("#pwEnable").removeClass("is-valid");
        $("#pwEnable").addClass("is-invalid");
        $("#pwDisable").removeClass("is-invalid");
        $("#pwDisable").addClass("is-valid");
    }
});

//이메일 인증
var AuthTimer=0;
$("#emailSendBtn").click(function() {// 메일 입력 유효성 검사
    var mail = $("#email").val(); //사용자의 이메일 입력값.
    if (mail == "") {
        alert("메일 주소가 입력되지 않았습니다.","info");
    } else {
        //mail = mail+"@"+$(".domain").val(); //셀렉트 박스에 @뒤 값들을 더함.
        $.ajax({
            type : "post",
            url : "/user/email/send",
            data : {
                email:mail
                },
            dataType :'json',
        });
        alert("인증번호가 전송되었습니다.","info");
        emailOk = false;
        if(AuthTimer!=0){
            AuthTimer.fnStop();
        }
        timerStarter();
    }
});
```

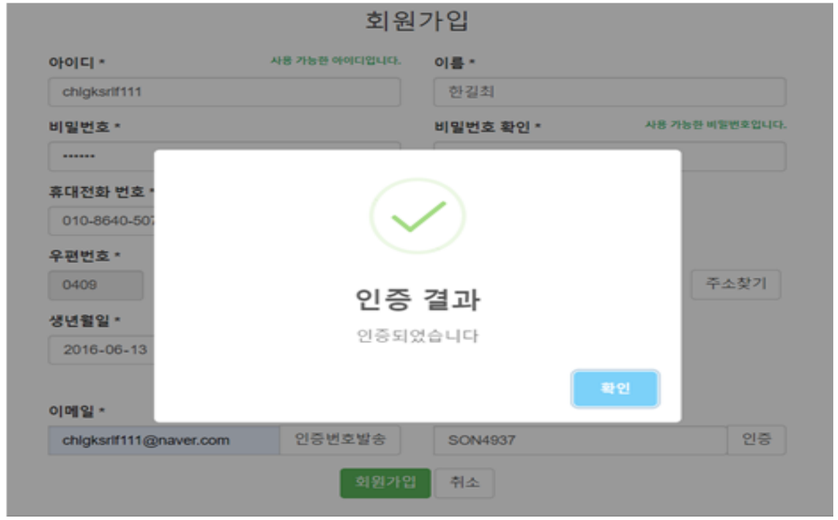
```
//이메일 코드 일치 확인
$("#emailConfirm").click(function() {
    var authNum = $("#authNum").val();
    $.ajax({
        type:"post",
        url:"/user/email/certificate",
        data:{"authNum":authNum},
        dataType :'json',
        success:function(item){
            alert(item.message,item.info);
            emailOk = item.emailOk;
            if(emailOk){
                AuthTimer.fnStop();
                $("#email").attr("readonly","readonly");
            }
            isOk = allOk( idOk, pw1Ok, pw2Ok, emailOk );
            if( isOk == true ){
              $("#insertBtn").prop("disabled",false);
            }
            //$("#chkAllOk").text(idOk + ', ' + pw1Ok + ', ' + pw2Ok + ', ' + emailOk + '마지막 idOk:' + isOk );

        }
    });
});

//이메일 타이머
// 타이머 구현_daldal
function $ComTimer(){
    //prototype extend
}
//타이머 로직
$ComTimer.prototype = {
    comSecond : ""
    , fnCallback : function(){}
    , timer : ""
    , domId : ""
    , fnTimer : function(){
        var m = Math.floor(this.comSecond / 60) + "분 " + (this.comSecond % 60) + "초"; // 남은 시간 계산
        this.comSecond--;                  // 1초씩 감소
        console.log(m);
        this.domId.innerText = m;
        if (this.comSecond < 0) {          // 시간이 종료 되었으면..
            clearInterval(this.timer);     // 타이머 해제
            alert("인증시간이 초과하였습니다. 다시 인증해주시기 바랍니다.","warning");
            $.ajax({
                type:"post",
                url:"/user/email/end",
            });
        }
    }
    ,fnStop : function(){
        clearInterval(this.timer);
    }
}
//타이머 설정
function timerStarter(){
    AuthTimer = new $ComTimer()
    AuthTimer.comSecond = 180; // 제한 시간
    AuthTimer.fnCallback = function(){alert("다시인증을 시도해주세요.","warning")}; // 제한 시간 만료 메세지
    AuthTimer.timer =  setInterval(function(){AuthTimer.fnTimer()},1000);

    AuthTimer.domId = document.getElementById("emailTimer");
}
```

```
//이메일 인증 알림창
var alert = function(msg, ifo) {
    swal({
        title : "인증 결과",
        text : msg,
        icon : ifo,
        timer : 3000,
        customClass : "sweet-size",
         button: {
             text: "확인"
        },
        showConfirmButton : true,
        closeOnConfirm : true
    });
}
```

KOSTA 학습관리시스템 – 이메일 인증







- 이메일 인증 과정입니다.
- 이메일을 입력하고 3분안에 인증을 해야합니다.

KOSTA 학습관리시스템 – 이메일 인증

```
//이메일인증
@ResponseBody
@PostMapping("/user/email/send")
public void sendmail(String email, HttpServletRequest request) throws MessagingException {
    HttpSession session = request.getSession();
    //코드 생성
    //난수 생성을 위한 랜덤 클래스
    Random random=new Random();
    //인증번호
    String key="";
    //입력 키를 위한 코드
    for(int i =0; i<3;i++) {
        int index=random.nextInt(25)+65; //A~Z까지 랜덤 알파벳 생성
        key+=(char)index;
    }
    int numIndex=random.nextInt(9999)+1000; //4자리 랜덤 정수를 생성
    key+=numIndex;

    //본문 작성
    StringBuffer emailcontent = new StringBuffer();
    emailcontent.append("<!DOCTYPE html>");
    emailcontent.append("<html>");
    emailcontent.append("<head>");
    emailcontent.append("<body>");
    emailcontent.append(
            " <div"
          + "   style=\"font-family: 'Apple SD Gothic Neo', 'sans-serif' !important; width: 400px; height: 600px; border-top: 4px solid #02b875; margin: 16
          + "   <h1 style=\"margin: 0; padding: 0 5px; font-size: 28px; font-weight: 400;\">"
          + "       <span style=\"font-size: 15px; margin: 0 0 10px 3px;\">YG1110 BLOG</span><br />"
          + "       <span style=\"color: #02b875\">메일인증</span> 안내입니다."
          + "   </h1>\n"
          + "   <p style=\"font-size: 16px; line-height: 26px; margin-top: 50px; padding: 0 5px;\">"
          + "       환영합니다.<br />"
          + "       KOSTA에 가입해 주셔서 진심으로 감사드립니다.<br />"
          + "       아래 <b style=\"color: #02b875\">'메일 인증'</b> 이메일 코드를 입력하여 회원가입을 완료해 주세요.<br />"
          + "       감사합니다."
          + "   </p>"
          + "   <p"
          + "       style=\"display: inline-block; width: 210px; height: 45px; margin: 30px 5px 40px; background: #02b875; line-height: 45px; vertical-alig
          + "       인증 번호 :"
          + "       key
          + "   </p>"
          + "   <div style=\"border-top: 1px solid #DDD; padding: 5px;\"></div>"
          + " </div>"
    );
    emailcontent.append("</body>");
    emailcontent.append("</html>");
    emailService.checkMain(email, "[KOSTA 이메일 인증]", emailcontent.toString());
    session.setAttribute("key", key);
}
```

- 이메일 인증 키 생성입니다.
- 알파벳과 숫자의 조합을 난수로 생성합니다.

```
@Service
public class EmailService {

    private JavaMailSender javaMailSender;

    public EmailService(JavaMailSender javaMailSender) {
        this.javaMailSender = javaMailSender;
    }

    public void checkMain(String email, String subject, String contents) throws MessagingException {
        MimeMessage mimeMessage = javaMailSender.createMimeMessage();
        MimeMessageHelper message = new MimeMessageHelper(mimeMessage, "utf-8");

        message.setTo(email); //스크립트에서 보낸 메일을 받을 사용자 이메일 주소
        message.setSubject(subject);
        message.setText(contents, true);

        javaMailSender.send(mimeMessage);
    }

}
```

- EmailService(위쪽) JavaMailSender를 사용했습니다.
- 이메일 주소, 이메일 제목, 본문 내용을 담아 JavaMailSender를 사용하여 이메일을 보내었습니다.

KOSTA 학습관리시스템-이메일 인증 확인

```java
//이메일인증 키확인
@ResponseBody
@PostMapping("/user/email/certificate")
public Map<String, Object> emailCertificate(String authNum, HttpServletRequest request) {
    String emailKey="";
    String message="";
    String info="";
    boolean emailOk = false;
    Map<String, Object> map = new HashMap<>();
    HttpSession session = request.getSession();
    emailKey = (String)session.getAttribute("key");
    //테스트용 콘솔창 키 표시
    //System.out.println(session.getAttribute("key"));
    if(authNum.equals("")){
        message = "메일 주소가 입력되지 않았습니다.";
        info = "warning";
    }
    else if (authNum.equals(emailKey)) {   //인증 키 값을 비교를 위해 텍스트인풋과 벨류를 비교
        message = "인증되었습니다";
        info = "success";
        emailOk = true;
    } else {
        message = "인증 실패";
        info = "warning";
    }
    map.put("message", message);
    map.put("info", info);
    map.put("emailOk", emailOk);
    session.removeAttribute("key");
    return map;
}
```

```javascript
//이메일 타이머
// 타이머 구현_daldal
function $ComTimer(){
    //prototype extend
}
//타이머 로직
$ComTimer.prototype = {
    comSecond : ""
    , fnCallback : function(){}
    , timer : ""
    , domId : ""
    , fnTimer : function(){
        var m = Math.floor(this.comSecond / 60) + "분 " + (this.comSecond % 60) + "초"; // 남은 시간 계산
        this.comSecond--;                  // 1초씩 감소
        console.log(m);
        this.domId.innerText = m;
        if (this.comSecond < 0) {          // 시간이 종료 되었으면..
            clearInterval(this.timer);     // 타이머 해제
            alert("인증시간이 초과하였습니다. 다시 인증해주시기 바랍니다.","warning");
            $.ajax({
                type:"post",
                url:"/user/email/end",
            });
        }
    }
    ,fnStop : function(){
        clearInterval(this.timer);
    }
}
//타이머 설정
function timerStarter(){
    AuthTimer = new $ComTimer()
    AuthTimer.comSecond = 180; // 제한 시간
    AuthTimer.fnCallback = function(){alert("다시인증을 시도해주세요.","warning")}; // 제한 시간 만료 메세지
    AuthTimer.timer =  setInterval(function(){AuthTimer.fnTimer()},1000);

    AuthTimer.domId = document.getElementById("emailTimer");
}
```

- 인증키를 발급받으면 이메일 타이머가 작동합니다.
- 세션에 저장된 인증 키를 입력한 값과 비교하여 메세지정보를 리턴합니다.(왼쪽)
- 이메일 타이머의 시간이 초과하면 세션에 저장된 키는 제거한다. (오른쪽)