SHBook
스프링 포트폴리오

Aws 배포 주소 : http://3.38.153.113/
Git 주소 : GitHub - flypigmong/SH-shop
블로그 주소 : https://h-this-and-that.tistory.com/

목차

1. 기획 의도

- 그동안 배운 자바 개발자 과정에서의
  자바,자바스크립트,JSP,HTML5,CSS,MySQL, Spring과 게시판
  CRUD 등 기능들을 잘 담을 수 있을 거라 생각해,

도서 판매 사이트를 구상 하게 되었습니다.

2. 개발 환경

소프트웨어	배포
Spring Tool Suite3 , JAVA11, HTML5, CSS3, Jqeury, MySQL(8.0), Tomcat 9, Maven	Amazon Ligthsail, Ubuntu 18.0.4 LTS 2 GB RAM, 2 vCPUs, 60 GB SSD Tomcat9, MySQL(8.0) FTP client(FilZilla 이용)

3. 요구사항 정의서

번호	이름	내용	유형	동작
1	메인화면	광고 배너 슬라이드	비기능	O
		평점 순 상품 슬라이드	비기능	O
		상품 목록 별점보이도록 함	비기능	O
		우측 상단에 로그인,회원가입,고객센터 아이콘 배치	비기능	O
		로그인 성공시 관리자면 관리자페이지,로그아웃,마이룸,장바구니,고객센터 배치하고 일반회원이면 로그아웃,마이룸,장바구니,고객센터 아이콘 배치	비기능	O
		상단 정중앙에 책 검색 배치	기능	O
		책 검색 하단에 국내,국외 배치	기능	O
		우측 상단에 자신의 아이디,충전금액,포인트 보이도록 함	비기능	O
		입력하지 않은 정보가 있다면 가입 막기	기능	O
2	회원가입(DB저장)	비밀번호 확인 일치할때만 가입가능함	기능	O
		이메일 인증번호 전송(SMTP 방식)	기능	O
		주소 (다음 API) 검색	기능	O
		회원가입 성공시 DB에 저장(비밀번호는 암호화)	기능	O

번호	이름	내용	유형	동작
3	검색(메인페이지)	검색 목록으로는 책 제목, 작가 (일부를 적더라도 검색 가능)	기능	O
4	평점 순 상품(메인페이지)	책 사진, 평점 평균 , 제목이 보이도록 함	비기능	O
		이전,다음 버튼을 누르면 이동하도록 함	비기능	O
		클릭 시 상품 상세페이지로 이동하도록 함	비기능	O
5	광고 배너(메인페이지)	광고 상품이 보이고 시간이 지나면 자동으로 슬라이드 (slick 적용)	비기능	O
6	로그인	아이디와 비밀번호를 입력하도록 함	비기능	O
		아이디 혹은 비밀번호 둘 중 하나라도 공백이거나 틀리면 경고 창 띄움	비기능	O
		로그인 성공 시 메인 화면으로 이동하도록 함	비기능	O
7	비밀번호 변경	기존비밀번호가 일치해야 함	기능	O
		새 비밀번호와 새 비밀번호 확인이 일치해야 함	기능	O
8	관리자 페이지	관리자 로그인 시 뜨는 관리자 아이콘 클릭 시, 관리자 페이지로 이동하도록 하고, 사이드에 상품 등록, 상품 관리, 작가 등록, 작가 관리, 회원 관리, 주문 현황 배너를 클릭할 수 있음	비기능	O
9	관리자 페이지-상품 등록	상품 등록 클릭 시, 책 제목,작가,출판일,출판사, 책 카테고리, 상품 가격, 상품 재고, 상품 할인율, 책 소개, 책 목차, 상품 이미지를 기입할 수 있음	비기능	O

번호	이름	내용	우선 순위	동작
9	관리자페이지 - 상품 등록	작가선택 클릭 시 팝업 창을 통해 DB에 저장 되어있는 작가를 선택 할 수 있음	비기능	O
		출판일은 날짜 선택을 통해 날짜를 선택할 수 있음(jquery의 datepicker 활용)	비기능	O
		책 카테고리는 대분류-중분류-소분류로 나누어 선택할 수 있음	기능	O
		책 소개, 책 목차는 위지윅(CK Editor-CKEditor5)활용	비기능	O
		상품 이미지는 jpg,png 형식만 허용, 파일의 크기는 1048576byte(1MB) 크기만 허용함	기능	O
		상품 이미지는 등록 전 썸네일로 보여지게 함(thumbnailator 라이브러리 사용)	기능	O
		상품 할인율은 기입한 숫자를 이용해서 태그에 할인 가격을 보여줌.	기능	O
		상품 등록 클릭 시 DB에 저장	기능	O
10	관리자페이지 – 상품 관리	상품 이름 클릭 시 상품 상세 페이지로 이동해서 보여줌 [책 제목,등록 날짜,최근 수정 날짜,작가,출판일,출판사,책 카테고리,상품 가격,상품 재고,상품 할인율,책 소개,책 목차,상품 이미지]	비기능	O
		상품관리 페이징 처리(10개씩), 상품 검색 기능	기능	O
		상품 상세페이지에서 수정 버튼 클릭 시 수정 페이지로 이동하고 삭제 버튼 클릭 시 삭제할 수 있게 함	기능	O
11	관리자 페이지 –작가 등록	작가 이름, 소속 국가, 작가 소개를 기입할 수 있게 함 소속 국가는 국내,국외로 나눔	비기능	O

번호	이름	내용	유형	동작
11	관리자 페이지 – 작가 등록	작가 등록 클릭 시 DB에 저장	기능	O
12	관리자 페이지 – 작가 관리	작가관리 페이징 처리(10개씩), 작가 검색 기능	기능	O
		작가 이름 클릭 시 작가 상세 페이지를 보여줌 [작가 번호, 작가 이름, 소속 국가, 작가소개, 등록 날짜, 수정 날짜]	비기능	O
13	관리자 페이지- 회원 관리	회원 페이징 처리와 검색 기능	기능	O
		회원 아이디, 회원 이름, 회원 메일, 회원 주소1, 회원 주소2, 회원 주소3, 가입 날짜, 돈, 포인트를 보여줌	비기능	O
14	관리자 페이지 - 주문 현황	주문 번호, 주문 아이디, 주문 날짜, 주문 상태, 취소를 보여주고 관리자는 이를 취소할 수 있음	기능	O
		주문 현황 페이징 처리와 검색 기능	기능	O
15	장바구니	상품 상세 페이지에서 주문 수량 선택 후 장바구니 담기를 클릭하면 장바구니에 담긴다. DB에 저장	기능	O
		주문 수량의 +,- 버튼으로 수량을 조절할 수 있음	기능	O
		장바구니 페이지에서는 상품 이미지, 상품 명, 가격, 수량, 합계, 삭제가 보여지며 하단에 총 상품 가격,배송비, 총 주문 상품 수, 총 결제 예상 금액, 총 적립 예상 마일리지, 주문하기를 보여준다. 주문하기를 클릭하면 주문페이지로 이동함	비기능	O
		삭제 버튼을 클릭하면 DB에서 삭제 됨	기능	O

번호	이름	내용	유형	동작
16	주문하기	장바구니 페이지 혹은 상품 상세페이지의 바로구매를 클릭해서 이동한 주문하기 페이지에서 주문하기를 클릭하면, 주문하기 상세페이지로 이동함	기능	O
		주문자 아이디, 이메일 ,주소록, 주문상품 이미지, 상품 이름, 판매가, 포인트 사용, 상품 금액, 배송비, 할인 금액, 최종 결제 금액, 적립예정 포인트를 보여줌 결제하기를 클릭하면 DB에 저장하고, 메인페이지로 이동함	비기능	O
		포인트는 가진 포인트 한도 내에서 사용 가능하도록 함	기능	O
		주소는 회원 정보의 주소를 이용하며, 직접 입력 클릭 시 이름과 주소를 직접 입력할 수 있음	기능	O
17	리뷰	상품 상세페이지, 메인화면에서 보이는 기능으로 일반회원들이 리뷰와 평점을 함께 달 수 있음	기능	O
		리뷰는 작성자 아이디, 날짜, 평점, 내용이 보여짐	비기능	O
		리뷰는 ajax를 통해 상세 페이지 새로고침 없이 DB에 저장되면 보일 수 있도록 하고, 페이징 처리(10개씩)를 함	기능	O
		리뷰의 평균 평점은 메인 화면에 상품 이미지 밑 별 모양 옆에 표시됨	비기능	O
		리뷰 등록 클릭 시 DB에 저장	기능	O
18	고객 센터	고객들이 질문 글 등을 올릴 수 있는 공간으로 관리자 뿐만 아니라 일반회원들도 댓글을 달 수 있음	기능	O
		댓글을 달면 글쓴이에게 댓글이 달렸다는 알림이 감	기능	O
		게시판 글들은 페이징 처리(10개씩)를 함	기능	O
		댓글은 ajax를 통해 상세 페이지 새로고침 없이 DB에 저장되면 보일 수 있도록 하고, 페이징 처리(10개씩)를 함	기능	O
		이미 댓글을 달았으면 댓글을 달 수 없도록 함	기능	O

4. UI 설계

메인

로그인

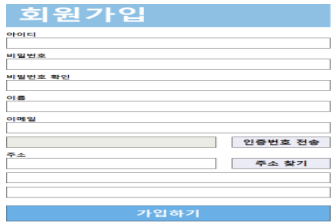
정보 변경

관리자페이지

관리자 로그인

일반회원 로그인

회원가입



마이룸

고객센터

검색 페이지

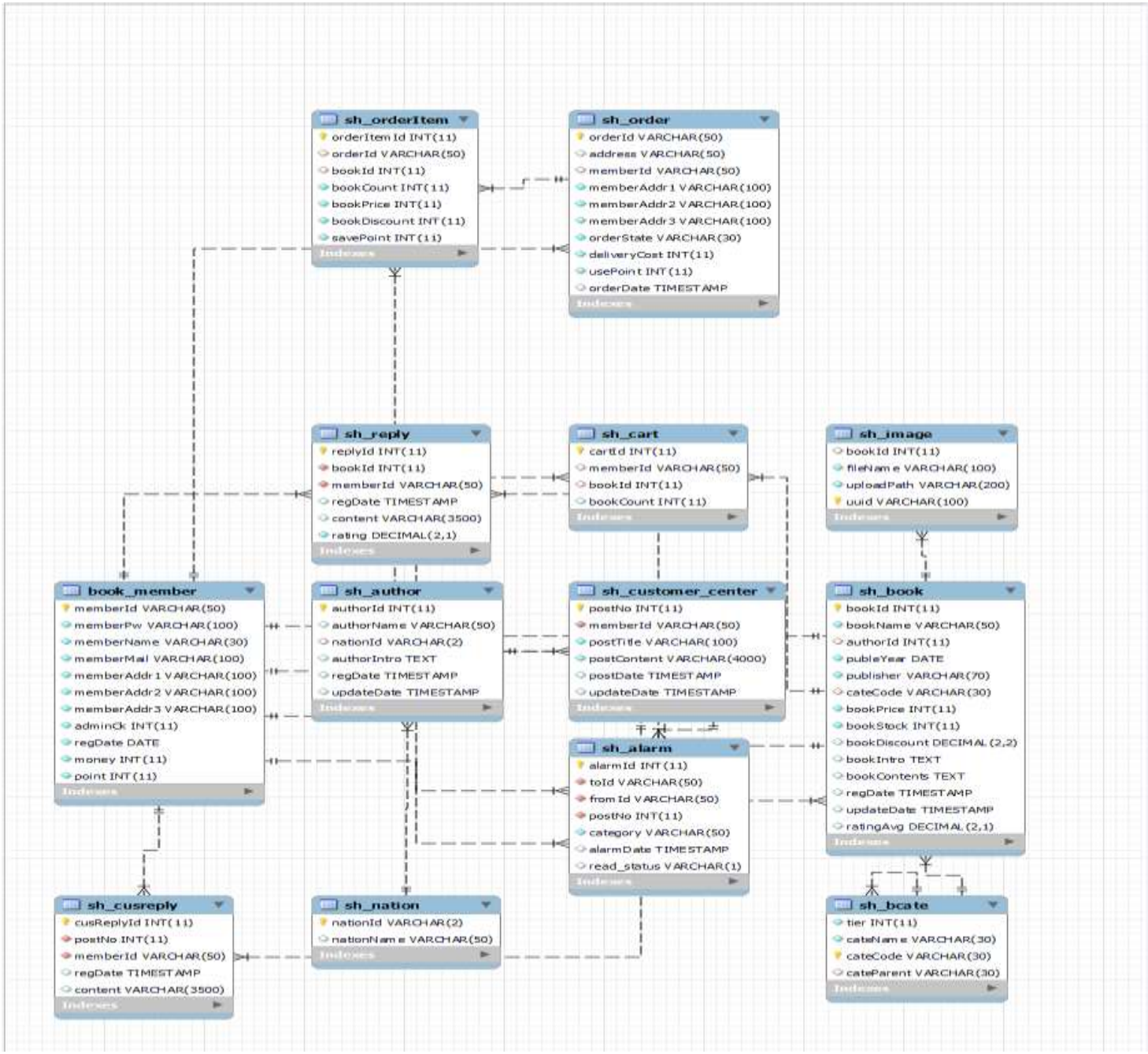
상세 페이지

장바구니

주문하기

5. DB 설계

테이블 명	테이블 설명
book_member	회원 테이블
sh_book	상품 테이블
sh_author	작가 테이블
sh_bcate	상품 카테고리 테이블
sh_nation	작가 국적 테이블
sh_image	상품 이미지 테이블
sh_cart	장바구니 테이블
sh_orderItem	주문하기 테이블
sh_order	주문 상세 테이블
sh_reply	상품 리뷰 테이블
sh_customer_center	고객센터 테이블
sh_cusreply	고객센터 댓글 테이블
sh_alarm	고객센터 알림 테이블



5. DB 설계

테이블 설명

book_member

필드	타입	NULL 허용	Primary Key	Default	비고
memberId	VARCHAR(50)	O	PK		회원 아이디
memberPw	VARCHAR(100)	O			회원 비밀번호
memberName	VARCHAR(30)	O			회원 이름
memberMail	VARCHAR(100)	O			회원 이메일
memberAddr1	VARCHAR(100)	O			회원 우편번호
memberAddr2	VARCHAR(100)	O			회원 주소
memberAddr3	VARCHAR(100)	O			회원 상세주소
adminCk	INT(11)	O			관리자 여부
regDate	DATE	O		CURRENT_TIMESTAMP	등록날짜
money	INT(11)	O			회원 돈
point	INT(11)	O			회원 포인트

sh_book

필드	타입	NULL 허용	Primary Key	Default	비고
bookId	INT(11)	O	PK		AUTO_INCREMENT, 상품 번호
bookName	VARCHAR(50)	O			상품 이름
authorId	INT(11)	NULL		NULL	작가 번호
pubLeYear	DATE	O			출간 연도
publisher	VARCHAR(70)	O			출판사
cateCode	VARCHAR(30)	NULL		NULL	카테고리
bookPrice	INT(11)	O			상품 가격
bookStock	INT(11)	O			상품 재고
bookDiscount	DECIMAL(2,2)	NULL		NULL	상품 할인율
bookIntro	TEXT	NULL		NULL	상품 목차
bookContents	TEXT	NULL		NULL	상품 내용
regDate	TIMESTAMP	O		CURRENT_TIMESTAMP	등록 날짜
updateDate	TIMESTAMP	O		CURRENT_TIMESTAMP	수정 날짜
ratingAvg	DECIMAL(2,1)	O			평균 평점

sh_author

필드	타입	NULL 허용	Primary Key	Default	비고
authorId	INT(11)	NOT NULL	Primary Key		AUTO_INCREMENT,작가 번호
authorName	VARCHAR(50)	O			작가 이름
nationId	VARCHAR(2)	O			작가 국적
authorIntro	TEXT	O			작가 설명
regDate	TIMESTAMP	O		CURRENT_TIMESTAMP	등록 날짜
updateDate	TIMESTAMP	O		CURRENT_TIMESTAMP	수정 날짜

sh_bcate

필드	타입	NULL 허용	Primary Key	Default	비고
tier	INT(11)	NOT NULL			카테고리 등급 (ex:1,2,3)
cateName	VARCHAR(30)	NOT NULL			카테고리 이름
cateCode	VARCHAR(30)	NOT NULL	Primary Key		카테고리
cateParent	VARCHAR(30)	O			상위 카테고리

sh_nation

필드	타입	NULL 허용	Primary Key	Default	비고
nationId	VARCHAR(2)	NOT NULL	Primary Key		국가 번호(01,02)
nationName	VARCHAR(10)	O			국가 이름(국내,국외)

sh_image

필드	타입	NULL 허용	Primary Key	Default	비고
bookId	INT(11)	O			상품 번호
filename	VARCHAR(100)	NOT NULL			사진 이름
uploadPath	VARCHAR(200)	NOT NULL	Primary Key		업로드 경로
Uuid	VARCHAR(100)	O			uuid(일렬번호)

sh_cart

필드	타입	NULL 허용	Primary Key	Default	비고
cartId	INT(11)	NOT NULL	Primary Key		장바구니 번호
memberId	VARCHAR(50)	O			회원 아이디
bookId	INT(11)	O			상품 아이디
bookCount	INT(11)	O			상품 수량

sh_orderItem

필드	타입	NULL 허용	Primary Key	Default	비고
orderItemId	INT(11)	NOT NULL	Primary Key		주문 상품 번호
orderId	VARCHAR(50)	O			주문 번호
bookId	INT(11)	O			상품 번호
bookCount	INT(11)	NOT NULL			주문 상품 수량
bookPrice	INT(11)	NOT NULL			주문 상품 가격
bookDiscount	INT(11)	NOT NULL			주문 상품 할인율
savePoint	INT(11)	NOT NULL			주문 상품 마일리지

sh_order

필드	타입	NULL 허용	Primary Key	Default	비고
orderId	VARCHAR(50)	NOT NULL	Primary Key		주문 번호
Address	VARCHAR(50)	O			배송받는사람이름
memberId	VARCHAR(50)	O			회원 번호
memberAddr1	VARCHAR(100)	NOT NULL			우편 번호
memberAddr2	VARCHAR(100)	NOT NULL			주소
memberAddr3	VARCHAR(100)	NOT NULL			상세 주소
orderStart	VARCHAR(30)	NOT NULL			주문 상태
deliveryCost	INT(11)	NOT NULL			배송비
usePoint	INT(11)	O			사용한 포인트
orderDate	TIMESTAMP	O		CURRENT_TIMESTAMP	주문 날짜

sh_reply

필드	타입	NULL 허용	Primary Key	Default	비고
replyId	INT(11)	NOT NULL	Primary Key		AUTO INCREMENT, 댓글 번호
bookId	INT(11)	NOT NULL			책 번호
memberId	VARCHAR(50)	NOT NULL			회원 아이디
regDate	TIMESTAMP	O			등록 날짜
content	VARCHAR(3500)	O			내용
rating	DECIMAL(2,1)	NOT NULL		CURRENT_TIMESTAMP	평점

sh_customer_center

필드	타입	NULL 허용	Primary Key	Default	비고
postNo	INT(11)	NOT NULL	Primary Key		AUTO INCREMENT, 글 번호
memberId	VARCHAR(50)	NOT NULL			회원 아이디
postTitle	VARCHAR(100)	NOT NULL			글 제목
postContent	VARCHAR(4000)	NOT NULL			글 내용
postdate	TIMESTAMP	O		CURRENT_TIMESTAMP	작성 날짜
updateDate	TIMESTAMP	O		CURRENT_TIMESTAMP	수정 날짜

sh_cusreply

필드	타입	NULL 허용	Primary Key	Default	비고
cusReplyId	INT(11)	NOT NULL	Primary Key		AUTO INCREMENT, 댓글 번호
postNo	INT(11)	NOT NULL			댓글 번호
memberId	VARCHAR(50)	NOT NULL			회원 아이디
regDate	TIMESTAMP	O		CURRENT_TIMESTAMP	등록 날짜
content	VARCHAR(3500)	O			댓글 내용

sh_alarm

필드	타입	NULL 허용	Primary Key	Default	비고
alarmId	INT(11)	NOT NULL			AUTO INCREMENT, 알람 번호
toId	VARCHAR(50)	NOT NULL			알람 받는 사람 아이디
fromId	VARCHAR(50)	NOT NULL			알람 보내는 사람 아이디
postNo	INT(11)	NOT NULL			글 번호
category	VARCHAR(50)	NOT NULL			글 카테고리(알람 구분짓기 위함)
alarmDate	TIMESTAMP	O		CURRENT_TIMESTAMP	알림 날짜
read_status	VARCHAR(1)	O		"X"	알림 읽음 여부

05 기술상세(핵심코드)

1 회원 가입

1.1 회원가입
1.2 아이디 중복 검사
1.3 비밀번호 암호화
1.4 이메일 인증
1.5 주소록 API
1.6 로그인

2 메인 화면

2.1 평점 순 메인 페이지
2.2 메인페이지 카테고리 링크

3 검색(&페이징)

3.1 책 제목,작가 키워드 검색
3.2 키워드 검색 후 카테고리별 필터링

4 관리자페이지

4.1 상품 등록
4.1.1 작가 선택 팝업창(상품 등록)
4.2 상품 수정
4.2.1 기존 업로드 이미지 삭제(수정페이지)
4.3 상품 삭제
4.4 작가 등록
4.5 작가 수정
4.6 작가 삭제

5 업로드(이미지)

5.1 업로드
5.2 업로드 이미지 화면 띄우기

6 주문하기(&장바구니)

6.1 장바구니 등록
6.2 구매

7 댓글

7.1 댓글 등록
7.2 댓글 수정
7.3 댓글 삭제

8 마이룸

8.1 회원정보 가져오기
8.2 비밀번호 변경

9 인터셉터

9.1 관리자/장바구니/마이룸/로그인/고객센터 인터셉터 적용

10 알림

10..1 알림
10. 2 알림 목록(&DB저장)

1. 회원 가입

회원가입 페이지 필요정보 입력후 '회원가입' 버튼을 누르면 회원가입 기능 실행

Views/member/join.jsp

회원가입 필요한 정보 입력 후 ' 가입하기' 버튼을 클릭

com/sh/controller/MemberController.java

joinPost() 실행

Views/main.jsp

회원가입이 완료되면 메인페이지로 리다이렉트 이동

com/sh/service/MemberService.java

memberJoin() 실행

com/sh/service/MemberServicImpl.java

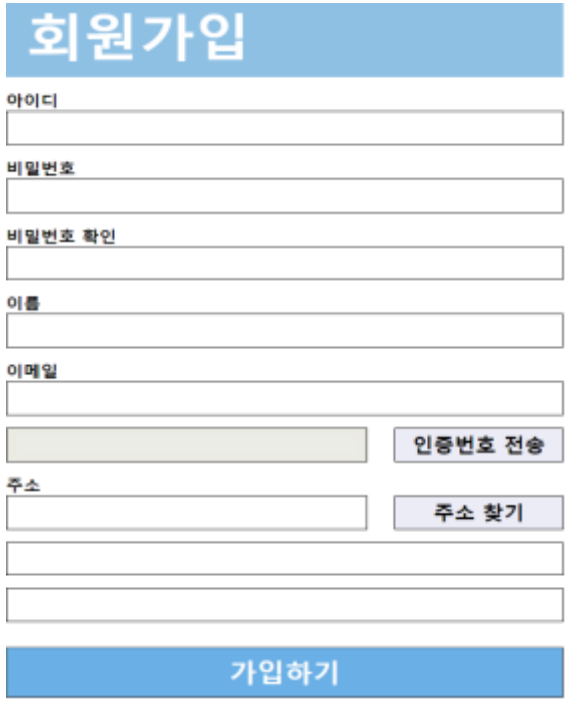
memberJoin() 실행

com/sh/mapper/MemberMapper.java

memberJoin() 실행

Resoureces/com/sh/mapper/MemberMapper.xml

id="memberJoin" 실행



1.1 아이디 중복 검사

회원가입 페이지의 아이디 입력란에 아이디 입력 시마다 아이디 중복 검사 실행.
중복 된 아이디 존재 시 아이디란 아래에 빨간 경고창 표시. 중복 아닐 시 초록 창 표시.

com/sh/controller/MemberController.java

```java
// 아이디 중복 검사
@PostMapping(value = "/memberIdChk")
@ResponseBody
public String memberIdChkPOST(String memberId) throws Exception{

    logger.info("memberIdChk() 진입");

    int result = memberservice.idCheck(memberId);

    logger.info("결과값 = " + result);

    if(result != 0) {

        return "fail";  // 중복 아이디가 존재

    } else {

        return "success";    // 중복 아이디 x

    }
} // memberIdChkPOST() 종료
```

Views/member/join.jsp

```jsp
//아이디 중복검사
$('.id_input').on("propertychange change keyup paste input", function(){

    //console.log("keyup 테스트");

    var memberId = $('.id_input').val();        // .id_input에 입력되는 값
    var data = {memberId : memberId}            // '컨트롤에 넘길 데이터 이름' : '데이터(.id_input에 입력되는 값)'

    $.ajax({
        type : "post",
        url : "/member/memberIdChk",
        data : data,
        success : function(result){
            //console.log("성공 여부" + result);
            if(result != 'fail'){
                $('.id_input_re_1').css("display","inline-block");
                $('.id_input_re_2').css("display", "none");
                idckCheck = true; // 아이디 중복이 없는 경우
            } else {
                $('.id_input_re_2').css("display","inline-block");
                $('.id_input_re_1').css("display", "none");
                idckCheck = false; // 아이디 중복된 경우
            }
        } // success 종료
    }); // ajax 종료

});// function 종료
```

Resources/com/sh/mapper/MemberMapper.xml

```
select count(*) from book_member where memberId = #{memberId}
```

1.2 비밀번호 암호화

회원가입 페이지의 아이디 입력란에 아이디 입력 시마다 아이디 중복 검사 실행.
중복 된 아이디 존재 시 아이디란 아래에 빨간 경고창 표시. 중복 아닐 시 초록 창 표시.

com/sh/service/MemberService.java

```
/* 회원가입 */
public void memberJoin(MemberVO member) throws Exception{
        String rawPw = ""; // 인코딩 전 비밀번호
        String encodePw = ""; // 인코딩 후 비밀번호

    rawPw = member.getMemberPw();            // 비밀번호 데이터 얻음
    encodePw = pwEncoder.encode(rawPw);        // 비밀번호 인코딩
    member.setMemberPw(encodePw);            // 인코딩된 비밀번호 member객체에 다시 저장
    logger.info("success1:" + rawPw,encodePw);
    /* 회원가입 쿼리 실행 */
    membermapper.memberJoin(member);            // DAO에게 회원가입 쿼리 요청

    }
```

DB에 암호화 돼서 저장된 모습

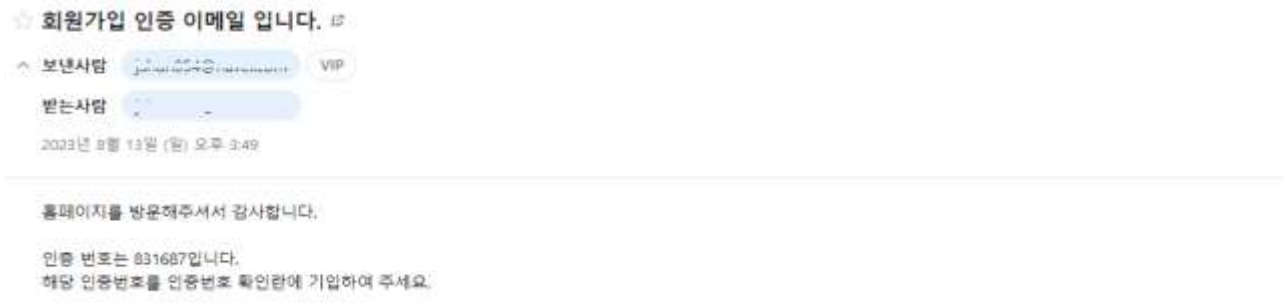
memberPw
$2a$10$RbCnAZnhYikBa1gQuRCu9OhfV4P46ACPQQvJ0IhDhNcbZUbY9Kohm
$2a$10$n9yfk2jHUGCCD8DpMYgkyOS1lXmGsZLejO7nVXbrQf8AoyQgGxJUa
$2a$10$pCCW2uSvBmZfxvLk6UvaUe2wbl2UygKy9MhCedB.EOkC0/F7jX2Me
$2a$10$o8ZI2a.MfviMdhqQRTWvOd.ygas9lXQUkim.CZL2nXo3zxXhw8Xy

1.3 이메일 인증

mailSender 라이브러리 활용.

Views/member/join.jsp

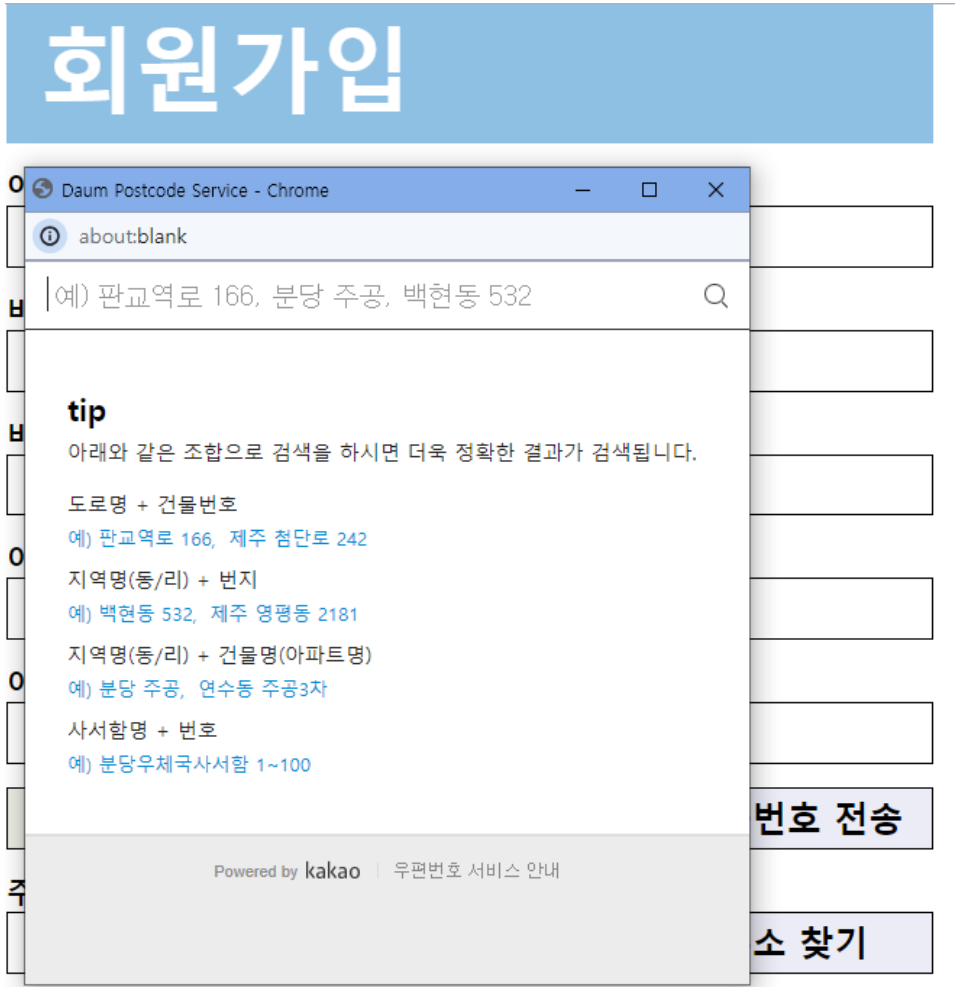
```
/* 인증번호 이메일 전송 */
$(".mail_check_button").click(function(){

   var email = $(".mail_input").val(); //입력한 이메일
   var checkBox = $(".mail_check_input"); // 인증번호 입력란
   var boxWrap = $(".mail_check_input_box"); //인증번호 입력란 박스
   var warnMsg = $(".mail_input_box_warn");   // 이메일 입력 경고글

   /* 이메일 형식 유효성 검사 */
   if(mailFormCheck(email)){
      warnMsg.html("이메일이 전송 되었습니다. 이메일을 확인해주세요.");
      warnMsg.css("display", "inline-block");
   } else {
      warnMsg.html("올바르지 못한 이메일 형식입니다.");
      warnMsg.css("display", "inline-block");
      return false;
   }

   $.ajax({

      type:"GET",
      url:"mailCheck?email=" + email,
      success:function(data){

         //console.log("data : " + data);
         checkBox.attr("disabled", false);
         boxWrap.attr("id", "mail_check_input_box_true");
         code = data;

      }

   });
});
```




```
/* 인증번호 비교 */

$(".mail_check_input").blur(function(){

   var inputCode = $(".mail_check_input").val();      // 입력코드
   var checkResult = $("#mail_check_input_box_warn");   // 비교 결과

   if(inputCode == code){                              // 일치할 경우
      checkResult.html("인증번호가 일치합니다.");
      checkResult.attr("class", "correct");
      mailnumCheck = true;
   } else {                                            // 일치하지 않을 경우
      checkResult.html("인증번호를 다시 확인해주세요.");
      checkResult.attr("class", "incorrect");
      mailnumCheck = false;
   }

});
```

1.4 주소록 API

다음 주소록 API 이용.



Views/member/join.jsp

```
/* 다음 주소 연동 */
function execution_daum_address(){
    new daum.Postcode({
        oncomplete: function(data) {
            // 팝업에서 검색결과 항목을 클릭했을때 실행할 코드를 작성하는 부분입니다.  // 각 주소의 노출 규칙에 따라 주소를 조합한다.
            // 내려오는 변수가 값이 없는 경우엔 공백('')값을 가지므로, 이를 참고하여 분기 한다.
            var addr = ''; // 주소 변수
            var extraAddr = ''; // 참고항목 변수
            //사용자가 선택한 주소 타입에 따라 해당 주소 값을 가져온다.
            if (data.userSelectedType === 'R') { // 사용자가 도로명 주소를 선택했을 경우
                addr = data.roadAddress;
            } else { // 사용자가 지번 주소를 선택했을 경우(J)
                addr = data.jibunAddress;
            }
            // 사용자가 선택한 주소가 도로명 타입일때 참고항목을 조합한다.
            if(data.userSelectedType === 'R'){
                // 법정동명이 있을 경우 추가한다. (법정리는 제외) // 법정동의 경우 마지막 문자가 "동/로/가"로 끝난다.
                if(data.bname !== '' && /[동|로|가]$/g.test(data.bname)){
                    extraAddr += data.bname;
                }
                // 건물명이 있고, 공동주택일 경우 추가한다.
                if(data.buildingName !== '' && data.apartment === 'Y'){
                    extraAddr += (extraAddr !== '' ? ', ' + data.buildingName : data.buildingName);
                }
                // 표시할 참고항목이 있을 경우, 괄호까지 추가한 최종 문자열을 만든다.
                if(extraAddr !== ''){
                    extraAddr = ' (' + extraAddr + ')';
                }
                // 주소변수 문자열과 참고항목 문자열 합치기
                addr += extraAddr;
            } else {
                addr += ' ';
            }
            $(".address_input_1").val(data.zonecode);
            //$("[name=memberAddr1]").val(data.zonecode);   // 대체가능
            $(".address_input_2").val(addr);
            //$("[name=memberAddr2]").val(addr); // 대체가능  // 상세주소 입력란 disabled 속성 변경 및 커서를
            $(".address_input_3").attr("readonly",false);
             $(".address_input_3").focus();
        }
    }).open();
```

1.5 로그인

아이디,패스워드 입력 후 로그인 버튼 클릭 -> 세션제거(인터셉터) -> 컨트롤러 실행 후 로그인 성공 시, 세션에 회원 데이터 담고 뷰(View)로 이동

Views/member/login.jsp

아이디,비밀번호 입력 후 로그인 버튼 클릭

↓

com/sh/interceptor/LoginInterceptor.java

Session.invalidate() 를 통해 세션 제거

↓

com/sh/controller/MemberController.java

Memberserver.memberLogin(member)를 통해 결과 값을 lvo(회원 아이디,이름,관리자권환, 돈, 포인트) 에 담음

↓

com/sh/service/MemberService.java

com/sh/service/MemberServiceImpl.java

memberLogin() 실행

com/sh/mapper/MemberMapper.java

memberLogin() 실행

↓

com/sh/mapper/MemberMapper.xml

select memberId, memberPw, memberName, adminck, money, point from book_member where memberId = #{memberId}

↓

Views/main.jsp

우측에 로그인 성공한 회원의 정보(아이디,돈,포인트) 상자를 띄운다.

2. 메인화면

2.1 평점 순 상품 배치

리뷰 평점 순으로 상품이 배치됩니다. (이전,버튼 버튼 클릭시 슬라이드)

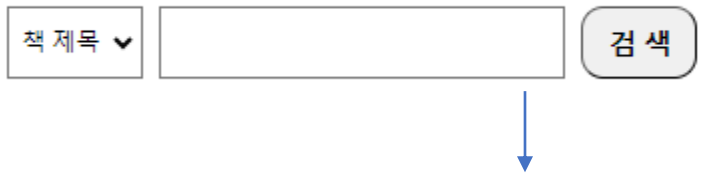
com/sh/mapper/BookMapper.xml

```
select bookId, bookName, ratingAvg, (select
cateName from sh_bcate where sh_book.cateCode =
sh_bcate.cateCode) as cateName
from sh_book
order by ratingAvg desc limit 8
```

2.2 메인 페이지 카테고리 링크

3. 책 제목,작가 검색(&페이징)

Views/main.jsp



com/sh/controller/BookController.java

```
/* 상품 검색 */
@GetMapping("/search")
public String searchGoodsGET(Criteria cri, Model model) {

    logger.info("cri : " + cri);

    List<BookVO> list = bookService.getGoodsList(cri);
    logger.info("pre list : " + list);
    if(!list.isEmpty()) {
        model.addAttribute("list", list);
        logger.info("list : " + list);
    } else {
        model.addAttribute("listCheck", "empty");

        return "search";
    }

    model.addAttribute("pageMaker", new PageDTO(cri, bookService.goodsGetTotal(cri)));
    logger.info("::pageMaker" + new PageDTO(cri, bookService.goodsGetTotal(cri)));

    String[] typeArr = cri.getType().split("");

    for (String s : typeArr) {   // type이 "A","AC","T", "TC" 인 경우에만 조회
        if (s.equals("T") || s.equals("A")) { //"T" 나 "A"를 포함
            model.addAttribute("filter_info", bookService.getCateInfoList(cri));
            logger.info("::filter_info" + bookService.getCateInfoList(cri));
        }
    }

    return "search";

}
```

com/sh/service/BookService.java

com/sh/service/BookServiceImpl.java

```
/* 상품 검색 */
@Override
public List<BookVO> getGoodsList(Criteria cri) {
    log.info("getGoodsList().....");

    String type = cri.getType();
    String[] typeArr = type.split("");
    String[] authorArr = bookMapper.getAuthorIdList(cri.getKeyword());

    for(String t : typeArr) {
        if(t.equals("A")) {
            cri.setAuthorArr(authorArr);
        }
    }

    List<BookVO> list = bookMapper.getGoodsList(cri);
    log.info("list ::::::: " + list);
    System.out.println("list ::::::::: " +list );
    list.forEach(book -> {

        int bookId = book.getBookId();
        System.out.println("bookId ::::" +bookId);

        List<AttachImageVO> imageList = attachMapper.getAttachList(bookId);
        book.setImageList(imageList);
        System.out.println("imageList ::::::: " + imageList );
    });

    return list;
}
```

com/sh/mapper/BookMapper.java

```java
/* 상품 검색 (검색 결과에 대한 데이터) */
public List<BookVO> getGoodsList(Criteria cri);

/* 상품 총 갯수 (페이징 데이터) */
public int goodsGetTotal(Criteria cri);
```

Resources/com/sh/mapper/BookMapper.xml

작가(A),카테고리(C),제목(T)을 검색하면 결과가 나오도록 함.

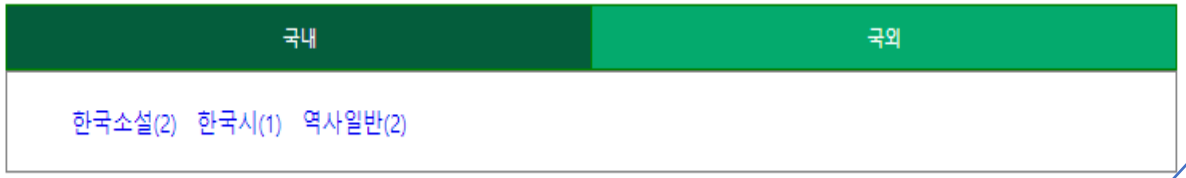
```xml
<!-- criteria(검색조건) -->
<sql id="criteria">

  <trim prefix="where (" suffix=")" prefixOverrides="AND">
    <foreach item="type" collection="typeArr">
      <trim prefix="AND">
        <choose>
          <when test="type == 'A'.toString()">
            <trim prefixOverrides="or">
              <foreach collection="authorArr" item="authorId">
                <trim prefix="or">
                  a.authorId = #{authorId}
                </trim>
              </foreach>
            </trim>
          </when>
          <when test="type == 'C'.toString()">
            a.cateCode like concat ('%', #{cateCode}, '%')
          </when>
          <when test="type == 'T'.toString()">
            bookName like concat ('%', #{keyword}, '%')
          </when>
        </choose>
      </trim>
    </foreach>
  </trim>

</sql>
```

리뷰 평점 순으로 상품이 배치됩니다. (이전,버튼 버튼 클릭시 슬라이드)

```xml
<!-- 상품 검색 -->
<select id="getGoodsList" resultType="com.sh.model.BookVO">

select  a.bookId, bookName, b.authorName, a.authorId, c.cateName, a.cateCode, publisher, publeYear, bookPrice, bookDiscount, ratingAvg
from sh_book a left outer join sh_author b on a.authorId = b.authorId
left outer join sh_bcate c on a.cateCode = c.cateCode
<include refid="criteria"></include>
order by bookId desc
limit #{skip}, #{amount}

</select>


<!-- 상품 총 개수 -->
<select id="goodsGetTotal" resultType="int">

  select count(*) from sh_book a

  <include refid="criteria"></include>

</select>
```

Views/search.jsp

3.2 키워드 검색 후 카테고리 별 필터링



com/sh/controller/BookController.java

```
    for (String s : typeArr) {   // type이 "A","AC","T", "TC" 인 경우에만 호출
        if (s.equals("T") || s.equals("A")) { //"T" 나 "A"를 포함
            model.addAttribute("filter_info", bookService.getCateInfoList(cri));
            logger.info("::filter_info" + bookService.getCateInfoList(cri));
        }
    }

    return "search";

}
```

com/sh/service/BookService.java

com/sh/service/BookServiceImpl.java

```
/* 검색결과 카테고리 필터 정보*/
@Override
public List<CateFilterDTO> getCateInfoList(Criteria cri) {

    List<CateFilterDTO> filterInfoList = new ArrayList<CateFilterDTO>();
    log.info("filterInfoList" + filterInfoList );

    String[] typeArr = cri.getType().split(""); // 검색 타입을 저장할 배열 생성
    log.info("::: typeArr" + cri.getType().split("") );

    String [] authorArr; // 저자정보 저장할 배열

    for (String type : typeArr) { // typeArr 배열의 각 요소를 반복실행
        if(type.equals("A")) { // type=A이면 ,
                                //getAuthorIdList 메서드로 cri 객체의 getKeyword 메서드로 검색 키워드를
                                //가져와서 저자 아이디 리스트를 반환하여 그 값을 authorArr에 저장
            authorArr = bookMapper.getAuthorIdList(cri.getKeyword());
            if(authorArr.length == 0 ) { //authorArr 배열이 요소를 가지지 않는 경우 getCateInfo() 메서드 실행X
                return filterInfoList;
            }
                cri.setAuthorArr(authorArr); // cri객체에 저자정보 저장
        }
    }

    //cateCode(카테고리 코드)를 반환해주는 getAuthorIdList를 호출하고 반환 값을 cateList 변수에 저장
    String[] cateList = bookMapper.getCateList(cri);
    log.info("getCateList" +cateList);
    String tempCateCode = cri.getCateCode(); // 임시로 cateCode 값 담기

    for (String cateCode : cateList) {
        cri.setCateCode(cateCode); // cateList 요소에 있는 cateCode를 cateCode 변수에 저장
        CateFilterDTO filterInfo = bookMapper.getCateInfo(cri);
        filterInfoList.add(filterInfo);
    }

    //임시로 저장해둔 tempCateCode 값을 cateCode 에 저장
    cri.setCateCode(tempCateCode);

    return filterInfoList;
}
```

3.2 키워드 검색 후 카테고리 별 필터링

com/sh/mapper/BookMapper.java

```java
/* 검색 대상 카테고리 리스트 */
public String[] getCateList(Criteria cri);

/* 카테고리 정보(+검색대상 갯수) */
public CateFilterDTO getCateInfo(Criteria cri);
```

Resources/com/sh/mapper/BookMapper.xml

```xml
<!-- 검색 대상 카테고리 리스트 (작가(type=A) 혹은 제목(type=T)으로 cateCode(코드번호) 가져옴 -->
<select id="getCateList" resultType="String">

   select DISTINCT cateCode from sh_book where
   <foreach item="type" collection="typeArr">
         <choose>
            <when test="type == 'A'.toString()">
               <trim prefixOverrides="or">
                  <foreach collection="authorArr" item="authorId">
                     <trim prefix="or">
                        authorId = #{authorId}
                     </trim>
                  </foreach>
               </trim>
            </when>
            <when test="type == 'T'.toString()">
               bookName like concat ('%', #{keyword}, '%')
            </when>
         </choose>
   </foreach>

</select>
```

Views/search.jsp

리뷰 평점 순으로 상품이 배치됩니다. (이전,버튼 버튼 클릭시 슬라이드)

```xml
<!-- 카테고리 정보(+검색대상 갯수) 검색조건(C or T)과 카테고리 코드번호 두 조건을 충족하는
카테고리 정보(catecount, catecode, cateName)가 출력되도록 함 -->
<select id="getCateInfo" resultType="com.sh.model.CateFilterDTO">

   select DISTINCT count(*) cateCount, a.cateCode,b.cateName from sh_book a left join sh_bcate b on a.cateCode = b.cateCode

   where

   <foreach item="type" collection="typeArr">
         <choose>
            <when test="type == 'A'.toString()">

               <trim prefix="(" suffix=")" prefixOverrides="or">

                  <foreach collection="authorArr" item="authorId">

                     <trim prefix="or">

                        authorId = #{authorId}

                     </trim>

                  </foreach>

               </trim>

               and a.cateCode = #{cateCode}

            </when>

            <when test="type == 'T'.toString()">

               bookName like concat ('%', #{keyword}, '%') and a.cateCode = #{cateCode}

            </when>

         </choose>
   </foreach>

</select>
```

4. 관리자 페이지-상품 등록(팝업,카테고리)

상품 등록- 작가 검색 팝업 창

com/sh/controller/AdminController.java

```
/* 작가 검색 팝업창 */
@GetMapping("/authorPop")
public void authorPopGET(Criteria cri, Model model) throws Exception {

    /* 게시물 출력 데이터 */
    cri.setAmount(5);

    List list = authorService.authorGetList(cri);
    logger.info("list " + list);
    int total = authorService.authorGetTotal(cri);
    logger.info("total" + total);

    logger.info("cri.setAmount() : " + cri.getAmount());

    if(!list.isEmpty()) {
        model.addAttribute("list",list);  // 작가 존재 경우
    } else {
        model.addAttribute("listCheck", "empty");  // 작가 존재하지 않을 경우
    }


    /* 페이지 이동 인터페이스 데이터 */
    model.addAttribute("pageMaker", new PageDTO(cri, total));
    logger.info("model " + model.toString());
    logger.info("authorPopGET.....");

}
```

상품 등록- 카테고리 리스트 : Jackson을 활용하여 '카테고리 리스트' 데이터 객체를 생성하고 해당 객체를 JSON 으로 변환한 후 뷰(View)로 데이터를 넘긴다.

Resources/com/sh/mapper/BookMapper.xml

```
<!-- 카테고리 리스트 -->
<select id="cateList" resultType="com.sh.model.CateVO">

   select * from sh_bcate order by catecode

</select>
```

com/sh/controller/AdminController.java

```
/* 상품 등록 페이지 접속 */
@GetMapping(value = "goodsEnroll")
public void goodsEnrollGET(Model model) throws Exception{
    logger.info("상품 등록 페이지 접속");

    ObjectMapper objm = new ObjectMapper(); //jackson 사용하기 위해 ObjectMapper클래스를 인스턴스화

    List list = adminService.cateList();

    String cateList = objm.writeValueAsString(list);

    model.addAttribute("cateList", cateList);

    logger.info("변경 전....." + list);
    logger.info("변경 후....." + cateList);

}
```

4. 관리자 페이지-상품 등록(카테고리)

'cateList' 데이터를 각 등급(tier)에 맞게 분류하여 배열에 저장함
각 티어별로 변수에 저장시키고, 대분류>중분류 순으로 선택한 하위 값 태그 출력

Views/admin/goodsEnroll.jsp

```
/* 카테고리 */
let cateList = JSON.parse('${cateList}');

let cate1Array = new Array(); //cateList 데이터 중 tire가 1인 데이터들만 저장
let cate2Array = new Array();
let cate3Array = new Array();
let cate1Obj = new Object();
let cate2Obj = new Object();
let cate3Obj = new Object();

let cateSelect1 = $(".cate1");
let cateSelect2 = $(".cate2");
let cateSelect3 = $(".cate3");


/* 카테고리 배열 초기화 메서드 */
function makeCateArray(obj,array,cateList, tier){
   for(let i = 0; i < cateList.length; i++){
      if(cateList[i].tier === tier){
         obj = new Object();

         obj.cateName = cateList[i].cateName;
         obj.cateCode = cateList[i].cateCode;
         obj.cateParent = cateList[i].cateParent;

         array.push(obj);

      }
   }
}

/* 배열 초기화 */
makeCateArray(cate1Obj,cate1Array,cateList,1);
makeCateArray(cate2Obj,cate2Array,cateList,2);
makeCateArray(cate3Obj,cate3Array,cateList,3);
```

```
/* 대분류 <option> 태그 */
for(let i = 0; i < cate1Array.length; i++){
   cateSelect1.append("<option value='"+cate1Array[i].cateCode+"'>" + cate1Array[i].cateName + "</option>"); //<select>
}

/* 중분류 <option> 태그 */
$(cateSelect1).on("change", function() {

   let selectVal1 = $(this).find("option:selected").val(); //대분류 선택값 가져오기
   cateSelect2.children().remove(); //option태그(중분류) 모두 지우기 ( 다시 대분류 선택하는 경우 기존의 option태그 없애기 위함)
   cateSelect3.children().remove(); //소분류도 지우기

   cateSelect2.append("<option value='none'>선택</option>")
   cateSelect3.append("<option value='none'>선택</option>");

   for(let i = 0; i < cate2Array.length; i++){
      if(selectVal1 === cate2Array[i].cateParent){
         cateSelect2.append("<option value='"+cate2Array[i].cateCode+"'>" + cate2Array[i].cateName + "</option>");
      } //대분류 선택값과 일치하는 'cateParent' 값을 가진 중분류 option태그 출력
   }


   /* 소분류 <option>태그 */
   $(cateSelect2).on("change",function(){

      let selectVal2 = $(this).find("option:selected").val(); //중분류 선택값 가져오기
      cateSelect3.children().remove();  // 소분류 option 태그 지우기
      cateSelect3.append("<option value='none'>선택</option>"); // 기본 option 태그 추가

      for(let i = 0; i < cate3Array.length; i++){
         if(selectVal2 === cate3Array[i].cateParent){
            cateSelect3.append("<option value='"+cate3Array[i].cateCode+"'>" + cate3Array[i].cateName + "</option>");
         }
      }
   });

});
```

4. 관리자 페이지-상품 수정(이미지 바꾸기)

com/sh/service/AdminService.java

com/sh/service/AdminServiceImpl.java

```
/*  상품 정보 수정 */
@Transactional
@Override
public int goodsModify(BookVO vo) {

   int result = adminMapper.goodsModify(vo);

   // 이미지파일이 정상적(result==1)이고 , 이미지정보가 존재할 때
   if(result == 1 && vo.getImageList() != null && vo.getImageList().size() > 0) {
      adminMapper.deleteImageAll(vo.getBookId()); //정보 모두 삭제

      // 각 요소 순서대로 이미지 정보를 DB에 저장..
      for (AttachImageVO attach : vo.getImageList()) {

         attach.setBookId(vo.getBookId());
         adminMapper.imageEnroll(attach);

      };

   }

   return result;
}
```

4. 관리자 페이지-상품 삭제

Views/admin/goodsEnroll.jsp

"삭제"버튼 클릭

↓

com/sh/controller/AdminController.java

goddsDelete() 실행

```
/* 상품 정보 삭제 */
@PostMapping("/goodsDelete")
public String goodsDeletePOST(int bookId, RedirectAttributes rttr) {

    logger.info("goodsDeletePOST.....");

    /* 서버파일 삭제 */
    // 이미지 정보 가져와서 fileList에 저장
    List<AttachImageVO> fileList = adminService.getAttachInfo(bookId);

    if(fileList != null) {

        List<Path> pathList = new ArrayList();

        for (AttachImageVO vo : fileList) {

            // 원본 이미지
            Path path = Paths.get("/var/lib/tomcat9/webapps/upload/", vo.getUploadPath(), vo.getUuid() + "_" + vo.getFileName());
            pathList.add(path);

            // 섬네일 이미지
            path = Paths.get("/var/lib/tomcat9/webapps/upload/", vo.getUploadPath(), "s_" + vo.getUuid()+"_" + vo.getFileName());
            pathList.add(path);

        }

        for (Path path : pathList) {
            path.toFile().delete();
        }

    }
    int result = adminService.goodsDelete(bookId);
    rttr.addFlashAttribute("delete_result", result);

    return "redirect:/admin/goodsManage";
}
```

com/sh/service/Admin Service.java

com/sh/service/Admin ServiceImpl.java

```
/* 상품 정보 삭제 */
@Override
@Transactional
public int goodsDelete(int bookId) {
    log.info("service:::goodsDelete().....");

    //DB 데이터 삭제( 상품 정보, 이미지 정보)
    adminMapper.deleteImageAll(bookId);

    return adminMapper.goodsDelete(bookId) ;
}
```

↓

com/sh/mapper/AdminMapper.java

goddsDelete() 실행

↓

Resources/com/sh/mapper/AdminMapper.xml

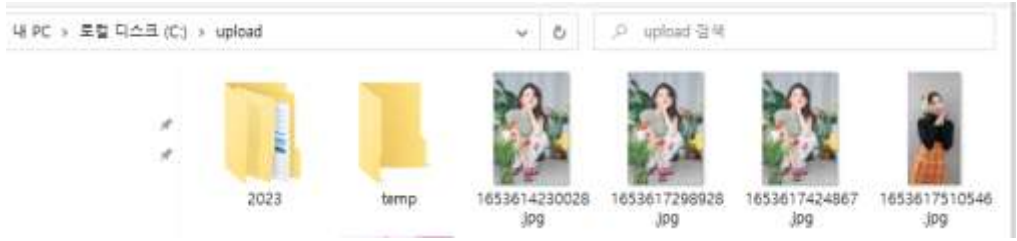
```
<!-- 상품 정보 삭제 -->
<delete id="goodsDelete">

    delete from sh_book where bookId = #{bookId}

</delete>
```

5. 업로드- 저장 폴더 생성

MultipartResolver 등록.
업로드 하는 날짜에 맞는 폴더 생성 후 해당 폴더에 이미지 저장. 용량 초과시 temp 폴더에 임시 저장.

com/sh/controller/AdminController.java

```
/* 첨부 파일 업로드 */
@PostMapping(value="/uploadAjaxAction", produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
public ResponseEntity<List<AttachImageVO>> uploadAjaxActionPOST(MultipartFile[] uploadFile) throws IOException {

    logger.info("uploadAjaxActionPOST.....");

    /* 이미지 파일 체크... */
    for(MultipartFile multipartFile : uploadFile) {
      File checkFile = new File(multipartFile.getOriginalFilename()); //전달받은 파일 File 객체로 만들고 File 참조 변수에 대입...
      String type = null; //MIME TYPE을 저장할 String 타입의 type 변수...

      //반환하는 MIME TYPE 데이터를 type 변수에 대입...(checkFile을 Path객체로 만들기위해 toPath 메서드사용)
      //type = Files.probeContentType(checkFile.toPath());
      //logger.info("MIME TYPE : " + type); //MIME TYPE 확인(image면 image 출력)
      type = multipartFile.getContentType();
      // image가 아니면
      if(!type.startsWith("image")) {

        List<AttachImageVO> list = null;
        return new ResponseEntity<>(list, HttpStatus.BAD_REQUEST);
      }
    }

    String uploadFolder = "/var/lib/tomcat9/webapps/upload/";  //파일 저장될 기본 경로

    /* 날짜 폴더 경로 */

    SimpleDateFormat myDate = new SimpleDateFormat("yyyy-MM-dd");//오늘의 날짜를 지정된 형식의 문자열 데이터로 생성하기 위한 SimpleDateFormat 객체 생성
    Date date = new Date(); //오늘의 날짜를 얻기위해 사용
    String str = myDate.format(date);//String에 대입
    String datePath = str.replace("-", File.separator);//날짜를 2023-08-10이면 '-'를 '/'로 변경

    /* 폴더 생성 */
    // 부모 경로 = uploadFolder
    // 자식 경로 = datePath
    File uploadPath = new File(uploadFolder, datePath);

    if(uploadPath.exists() == false) { //대상 파일 혹은 디렉터리가 없으면
      uploadPath.mkdirs(); //여러개(-s) 폴더 생성하는 메서드
    }
```




```
    /* 이미지 정보 담는 객체 */
    List<AttachImageVO> list = new ArrayList();

    for(MultipartFile multipartFile : uploadFile) {
      /* 이미지 정보 객체생성 */
      AttachImageVO imageVo = new AttachImageVO();

      /* 파일 이름 */
      String uploadFileName = multipartFile.getOriginalFilename();
      imageVo.setFileName(uploadFileName);
      imageVo.setUploadPath(datePath);

      /* uuid 적용 파일 이름 */
      // String 타입으로 변경하고 AttachImageVO 객체에 저장
      String uuid = UUID.randomUUID().toString();
      imageVo.setUuid(uuid);

      //파일 이름을 "UUID_파일 이름" 형식이 되도록 변경
      uploadFileName = uuid + "_" + uploadFileName;

      /* 파일 위치, 파일 이름을 합친 File 객체 */
      File saveFile = new File(uploadPath, uploadFileName);

      /* 파일 저장 */
      try {
        multipartFile.transferTo(saveFile);
      /* 방법 2 :thumbnailator 라이브러리 사용 */
      File thumbnailFile = new File(uploadPath, "s_" + uploadFileName);
      BufferedImage buf_oriImage = ImageIO.read(saveFile);

      //비율
      double ratio = 3;
      //넓이 높이
      int width = (int) (buf_oriImage.getWidth() / ratio);
      int height = (int) (buf_oriImage.getHeight() / ratio);

      Thumbnails.of(saveFile)
      .size(width, height)
      .toFile(thumbnailFile);
      } catch (Exception e) {
        e.printStackTrace();
      }
      list.add(imageVo);
    }
    // List<AttachImageVO> 이고 상태코드가 OK(200)인 ResponseEntity 객체가 생성 (Http의 바디에 추가될 데이터)
    ResponseEntity<List<AttachImageVO>> result = new ResponseEntity<List<AttachImageVO>>(list, HttpStatus.OK);
    return result;
}
```

5. 업로드- 상세 페이지 출력(JSON)

비동기 방식으로 요청 한 반환방식 ResponseEntity 클래스 사용
뷰(view)로 반환하는 이미지 정보는 JSON 형식으로 전달

com/sh/controller/AdminController.java

```
/* 이미지 정보 반환 */
@GetMapping(value="/getAttachList", produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
public ResponseEntity<List<AttachImageVO>>getAttachList(int bookId) {

logger.info("getAttachList....." + bookId);

   return new ResponseEntity(attachMapper.getAttachList(bookId),HttpStatus.OK);

}
```

com/sh/service/AttachService.java

com/sh/service/AttachServiceImpl.java

```
/* 이미지 데이터 반환 */
@Override
public List<AttachImageVO> getAttachList(int bookId) {

    log.info("getAttachList.....");

    return attachMapper.getAttachList(bookId);
}
```

com/sh/mapper/AttachMapper.java

```
/* 이미지 데이터 반환 */
public List<AttachImageVO> getAttachList(int bookId);
```

Resources/com/sh/mapper/AdminMapper.xml

```
<select id="getAttachList" resultType="com.sh.model.AttachImageVO">

   select * from sh_image where bookId = #{bookId}

</select>
```

6. 주문하기(&장바구니)

로그인 한 회원이 제품 상세 페이지에서 장바구니 담기 버튼을 클릭 시 회원의 장바구니에 해당 제품이 담긴다.

com/sh/controller/CartController.java

```
/* 장바구니 추가 */
@PostMapping("/cart/add")
@ResponseBody
public String addCartPOST(CartDTO cart, HttpServletRequest request) {

    // 로그인 체크
    HttpSession session = request.getSession();
    MemberVO mvo = (MemberVO) session.getAttribute("member");
    System.out.println("mvo: " + mvo);
    if (mvo == null) {
        return "5";
    }

    // 카트 등록
    int result = cartService.addCart(cart);
    System.out.println("result : " + result);
    return result + "";
}
```

com/sh/service/CartService.java

com/sh/service/CartServiceImpl.java

```
@Override
public int addCart(CartDTO cart) {
    // 장바구니 데이터 체크
    CartDTO checkCart = cartMapper.checkCart(cart);
    System.out.println("checkCart: " + checkCart);
    if(checkCart != null) {
        return 2;
    }

    // 장바구니 등록 & 에러 시 0 반환
    try {
        return cartMapper.addCart(cart);
    } catch (Exception e) {
        return 0;
    }
}
```

6. 주문하기(&장바구니)

로그인 한 회원이 장바구니 페이지 혹은 제품 상세페이지에서 주문하기 버튼을 클릭 후 포인트 사용과 주소 입력 후 결제하기 버튼을 누르면 주문하기가 실행된다.

com/sh/controller/OrderController.java → com/sh/service/OrderServiceImpl.java

```
@PostMapping("/order")
public String orderPagePost(OrderDTO od, HttpServletRequest request) {
    System.out.println(od);

    orderService.order(od);

    MemberVO member = new MemberVO();
    member.setMemberId(od.getMemberId());
        |
    member.setMemberId(od.getMemberId());

    HttpSession session = request.getSession();

    try {
        MemberVO memberLogin = membermapper.memberLogin(member);

        memberLogin.setMemberPw("");

        session.setAttribute("member", memberLogin);

    } catch (Exception e) {

        e.printStackTrace();
    }


    return "redirect:/main";
}
```

사용하는 포인트를 차감하고 얻는 포인트를 저장한 뒤, 상품 재고를 DB에서 차감한다.
장바구니의 상품 정보를 삭제한다.

```
@Override
@Transactional
public void order(OrderDTO OrderDTO) {
    // 사용할 데이터 가져오기
    //1.회원 정보 가져오기
    MemberVO member = memberMapper.getMemberInfo(OrderDTO.getMemberId());

    //2. 주문 정보 가져오기
    List<OrderItemDTO> ords = new ArrayList<>();
    for(OrderItemDTO OIdto : OrderDTO.getOrders()) {
        OrderItemDTO orderItem = orderMapper.getOrderInfo(OIdto.getBookId());
        // 수량 셋팅
        orderItem.setBookCount(OIdto.getBookCount());
        // 기본정보 셋팅
        orderItem.initSaleTotal();
        //List객체 추가
        ords.add(orderItem);
    }
    // OrderDTO 셋팅
    OrderDTO.setOrders(ords);
    OrderDTO.getOrderPriceInfo();

    // 3.DB 주문,주문상품(,배송정보) 넣기
    // orderId만들기 및 OrderDTO객체 orderId에 저장
    Date date = new Date();
    SimpleDateFormat format = new SimpleDateFormat("_yyyyMMddmmm");
    String orderId = member.getMemberId() + format.format(date);
    OrderDTO.setOrderId(orderId);

    // db넣기
    orderMapper.enrollOrder(OrderDTO);      //vam_order 등록
    for(OrderItemDTO OIdto : OrderDTO.getOrders()) {    //vam_orderItem 등록
        OIdto.setOrderId(orderId);
        orderMapper.enrollOrderItem(OIdto);
    }

    // 4. 사용 포인트 변동 적용
    // 사용 차감 & 변동 돈(money) Member객체 적용
    int calMoney = member.getMoney();
    calMoney -= OrderDTO.getOrderFinalSalePrice();
    member.setMoney(calMoney);

    // 포인트 차감, 포인트 증가 & 변동 포인트(point) Member객체 적용
    int calPoint = member.getPoint();
    calPoint = calPoint - OrderDTO.getUsePoint() + OrderDTO.getOrderSavePoint();  // 기존 포인트 - 사용 포인트 + 획득 포인트
    member.setPoint(calPoint);

    // 변동 돈, 포인트 DB 적용
    orderMapper.deductMoney(member);

    // 5. 재고 변동 적용
    for(OrderItemDTO oit : OrderDTO.getOrders()) {
        /* 변동 재고 값 구하기 */
        BookVO book = bookMapper.getGoodsInfo(oit.getBookId());
        book.setBookStock(book.getBookStock() - oit.getBookCount());
        /* 변동 값 DB 적용 */
        orderMapper.deductStock(book);
    }

    // 6. 장바구니 제거
    for(OrderItemDTO oit : OrderDTO.getOrders()) {
        CartDTO dto = new CartDTO();
        dto.setMemberId(OrderDTO.getMemberId());
        dto.setBookId(oit.getBookId());

        cartMapper.deleteOrderCart(dto);
    }
```

6. 주문하기(&장바구니)

주문 취소시에는 'orderState' 를 '주문취소 ' 로 변경한다.

↓

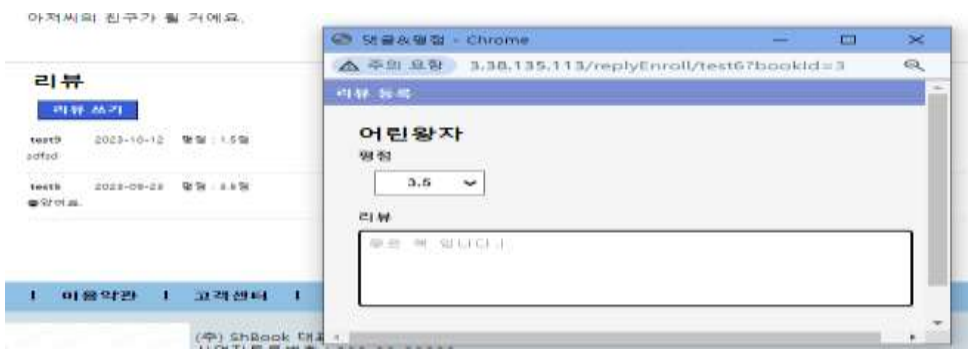
Resources/com/sh/mapper/OrderMapper.xml

```
<!-- 주문 아이템 테이블 등록 -->
<insert id="enrollOrderItem">

   insert into sh_orderItem(orderId, bookId, bookCount, bookPrice, bookDiscount, savePoint)
   values(#{orderId}, #{bookId}, #{bookCount}, #{bookPrice}, #{bookDiscount}, #{savePoint})

</insert>

<!-- 주문 금액 차감 -->
<update id="deductMoney">

   update book_member set money = #{money}, point = #{point} where memberid = #{memberId}

</update>

<!-- 주문 재고 차감 -->
<update id="deductStock">

   update sh_book set bookStock = #{bookStock} where bookId = #{bookId}

</update>
```

```
<!-- 주문 취소 -->
<update id="orderCancle">

   update sh_order set orderState= '주문취소' where orderId = #{orderId}

</update>

<!--  주문 상품 정보(주문 취소) -->
<select id="getOrderItemInfo" resultType="com.sh.model.OrderItemDTO">

   select * from sh_orderItem
   where orderId = #{orderId}

</select>

<!--  주문 정보(주문 취소) -->
<select id="getOrder" resultType="com.sh.model.OrderDTO">

   select * from sh_order
   where orderId = #{orderId}

</select>
```

7. 댓글(등록&수정&삭제)

평점과 내용을 입력 후 등록버튼을 누르면
댓글이 작성됨



Views/detail.jsp

리뷰 등록 버튼을 누르면 ajax 실행

com/sh/controller/BookController.java

이미 등록된 리뷰가 있는지 체크하고 없으면 실행

```
/* 댓글 등록 */
@PostMapping("/enroll")
public void enrollReplyPOST(ReplyDTO dto) {
    replyService.enrollReply(dto);
}

/* 댓글 체크 */
/* memberId, bookId 파라미터 */
/* 존재 : 1 / 존재x : 0 */
@PostMapping("/check")
public String replyCheckPOST(ReplyDTO dto) {
    return replyService.checkReply(dto);
}
```

com/sh/service/ReplyService.java

com/sh/service/ReplyServiceImpl.java

```
/* 댓글 등록 */
@Override
public int enrollReply(ReplyDTO dto) {

    int result = replyMapper.enrollReply(dto);

    setRating(dto.getBookId()); // 상품평점 평균값 구하기

    return result;
}

/* 댓글 체크*/
@Override
public String checkReply(ReplyDTO dto) {

    Integer result = replyMapper.checkReply(dto);

    if(result == null) {
        return "0";
    } else {
        return "1";
    }

}
```

7. 댓글(등록&수정&삭제)

com/sh/mapper/ReplyMapper.java

```java
/* 댓글 등록 */
public int enrollReply(ReplyDTO dto);

/* 댓글 존재 체크 */
public Integer checkReply(ReplyDTO dto);

/* 댓글 수정 */
public int updateReply(ReplyDTO dto);

/* 댓글 한개 정보(수정 페이지) */
public ReplyDTO getUpdateReply(int replyId);

/* 댓글 삭제(replyId를 조건으로) */
public int deleteReply(int replyId);
```

Resources/com/sh/mapper/ReplyMapper.xml

```xml
<!-- 댓글등록 -->
<insert id="enrollReply">

  insert into sh_reply(bookId, memberId, content, rating) values(#{bookId}, #{memberId}, #{content}, #{rating})

</insert>

<!-- 댓글 체크(이전 댓글 있는지 확인) -->
<select id="checkReply" resultType="integer">

  select replyId from sh_reply
  where memberId = #{memberId} and bookId = #{bookId}

</select>

<!-- 댓글 수정 -->
<update id="updateReply">

  update sh_reply set content = #{content}, rating = #{rating}
  where replyId = #{replyId}

</update>

<select id="getUpdateReply" resultType="com.sh.model.ReplyDTO">

  select * from sh_reply
  where replyId = #{replyId}

</select>

<!-- 댓글 삭제 -->
<delete id="deleteReply">

  delete from sh_reply
  where replyId = #{replyId}

</delete>
```

8. 마이룸(&비밀번호 변경)

로그인한 회원이 “마이룸”버튼을 클릭하면 해당 회원의 아이디를 조회해서 해당 회원의 아이디,이름,메일,주소,돈,포인트를 model에 담아 가져온다.

com/sh/controller/MemberController.java

```
/* 마이페이지 */
@GetMapping("/myPage/{memberId}")
public String myPageGet(@PathVariable("memberId") String memberId, Model model) {

    model.addAttribute("memberInfo", memberservice.memberInfo(memberId));
    logger.info("model " + model);

    return "member/myPage";
}
```

com/sh/service/MemberService.java

com/sh/service/MemberServiceImpl.java

```
@Override
public MemberVO memberInfo(String memberId) {
    logger.info("(service)membeGetrInfo()...."+ memberId);
    return membermapper.memberInfo(memberId);
}
```

com/sh/mapper/MemberMapper.java

```
// 회원 정보
public MemberVO memberInfo(String memberId);
```

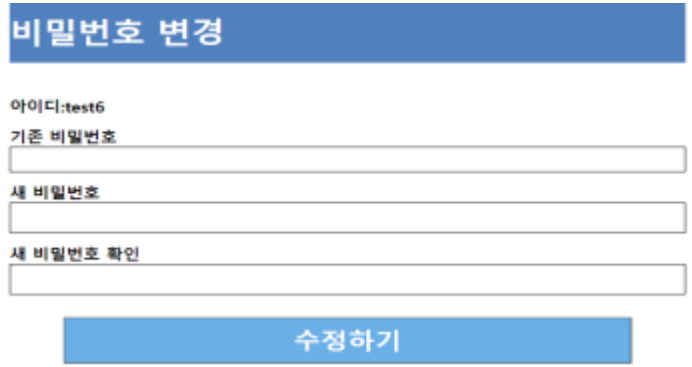
Resources/com/sh/mapper/MemberMapper.xml

SELECT memberId,memberName, memberMail, memberAddr1, memberAddr2, memberAddr3, money, point FROM book_member WHERE memberid = #{memberId}

Views/member/mypage.jsp

8. 마이룸(&비밀번호 변경)

로그인한 회원이 “비밀번호 변경” 버튼을 클릭하면 비밀번호 변경 페이지로 이동한다.
기존 비밀번호, 새 비밀번호, 새 비밀번호 확인을 입력 후 새 비밀번호와 새 비밀번호 확인이 일치하면 수정이 완료된다.
비밀번호 변경도 암호화 된 비밀번호로 저장된다.



com/sh/controller/MemberController.java

```
//비밀번호 변경
@PostMapping(value="/pwUpdate")
public String memberPwUpdate(@RequestParam("currentPw") String currentPw,
      @RequestParam("newPw") String newPw, HttpSession session, RedirectAttributes rttr, Model model) {

   // 세션에서 로그인한 회원의 정보 가져오기
   MemberVO member = (MemberVO) session.getAttribute("member");
   System.out.println("Member : " + member);
   System.out.println("currentPw : " + currentPw);
   System.out.println("newPw: " + newPw);

   // 서비스에 비밀번호 변경 요청 전달
   boolean result = memberservice.memberPwUpdate(member, currentPw, newPw);

   //결과에 따라 다른 페이지로 리다이렉트
   if(result) {
      session.setAttribute("msg1", "비밀번호 변경성공!");
      return "redirect:/member/pwUpdateForm";
   }else {
       session.setAttribute("msg1", "기존 비밀번호가 틀립니다.");
      return "redirect:/member/pwUpdateForm";
   }
}
```

com/sh/service/MemberService.java

com/sh/service/MemberServiceImpl.java

```
/* 비밀번호 변경 */
@Override
public boolean memberPwUpdate(MemberVO member, String currentPw, String newPw) {
   // 일치하는 아이디 있는지 확인
   MemberVO Ivo = membermapper.memberLogin(member);
   System.out.println("getmemberpwww:  " + Ivo.getMemberPw());
   System.out.println("memberpwwwwwwww:  " +currentPw);
   // 일치하는 아이디 있으면
   if(Ivo != null) {
      if(pwEncoder.matches(currentPw, Ivo.getMemberPw())) { //현재의 비밀번호가 일치하면
         String encodePw = pwEncoder.encode(newPw); // 변경할 비밀번호 인코딩
         member.setMemberPw(encodePw); // 인코딩 된 비밀번호 member객체에 다시 저장
         membermapper.memberPwUpdate(member); // dao 에 비밀번호 변경 요청 전달
         return true;      //비밀번호 변경 성공을 반환
      } else {
         return false;      //비밀번호 변경 실패를 반환
      }
   }
   return false;  //비밀번호 변경 실패를 반환
}
```

com/sh/mapper/MemberMapper.java

```
public void memberPwUpdate(MemberVO member);
```

Resources/com/sh/mapper/MemberMapper.xml

```
UPDATE book_member SET
memberPw= #{memberPw} WHERE
memberId= #{memberId}
```

9. 인터셉터 적용

회원이 로그인 하지 않고 기능(마이페이지,장바구니,고객센터)를 이용하려 하거나 관리자페이지에 접근할 경우 인터셉터에서 막고, 경고창을 띄운다.

com/sh/interceptor/AdminInterceptor.java

```
<mvc:interceptors>

    <!-- 로그인 -->
    <mvc:interceptor>
        <mvc:mapping path="/member/login" />
        <bean id="loginIntreceptor" class="com.sh.interceptor.LoginInterceptor" />
    </mvc:interceptor>

    <!-- 관리자페이지 -->
    <mvc:interceptor>
        <mvc:mapping path="/admin/**"/>
        <bean id="AdminIntreceptor" class="com.sh.interceptor.AdminInterceptor" />
    </mvc:interceptor>

    <!-- 장바구니페이지 -->
    <mvc:interceptor>
        <mvc:mapping path="/cart/**"/>
        <mvc:exclude-mapping path="/cart/add"/>
        <bean id="CartIntreceptor" class="com.sh.interceptor.CartInterceptor"/>
    </mvc:interceptor>


    <!-- 고객센터페이지 -->
    <mvc:interceptor>
        <mvc:mapping path="/member/customer/list"/>
        <bean id="CustomerInterceptor" class="com.sh.interceptor.CustomerInterceptor"/>
    </mvc:interceptor>

    <!-- 마이룸페이지 -->
    <mvc:interceptor>
        <mvc:mapping path="/member/myPage/**"/>
        <bean id="MyPageInterceptor" class="com.sh.interceptor.MyPageInterceptor"/>
    </mvc:interceptor>
</mvc:interceptors>
</beans>
```

com/sh/interceptor/AdminInterceptor.java

```
@Override
public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)
        throws Exception {

    HttpSession session = request.getSession();

    MemberVO lvo = (MemberVO)session.getAttribute("member");

    if(lvo == null || lvo.getAdminCk() == 0) {    // 관리자 계정 아닌 경우

        // response.sendRedirect("/main");    // 메인페이지로 리다이렉트
        response.setContentType("text/html; charset=UTF-8");
        PrintWriter printwriter = response.getWriter();
        printwriter.print("<script>alert('로그인이 필요한 서비스입니다.'); location.href='/'</script>");
        printwriter.flush();
        printwriter.close();
        return false;

    } else {

        return true;    // 관리자 계정 로그인 경우(lvo != null && lvo.getAdminCk() == 1)
    }
  }
}
```

회원의 adminCk==1 인 경우(관리자인 경우)에만 관리자 페이지 이용 가능

↓

관리자 페이지 이용

9. 인터셉터 적용

com/sh/interceptor/LoginInterceptor.java

```
@Override
public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)
        throws Exception {

    System.out.println("LoginInterceptor preHandle 작동");

    HttpSession session = request.getSession();

    session.invalidate();

    return true;
}
}
```

새로운 회원 로그인 시 세션 제거

새로운 회원 로그인 완료

com/sh/interceptor/CustomerInterceptor.java

com/sh/interceptor/CartInterceptor.java

com/sh/interceptor/MypageInterceptor.java

```
@Override
public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)
        throws Exception {

    HttpSession session = request.getSession();

    MemberVO mvo = (MemberVO)session.getAttribute("member");

    if(mvo == null) {
        //response.sendRedirect("/main");
        response.setContentType("text/html; charset=UTF-8");
        PrintWriter printwriter = response.getWriter();
        printwriter.print("<script>alert('로그인이 필요한 서비스입니다.'); location.href='/'</script>");
        printwriter.flush();
        printwriter.close();
        return false;
    } else {
        return true;
    }

}
}
```

DB에 저장되어있는 회원이 로그인 되어있을경우

마이페이지,장바구니,고객센터 이용가능

10.1 알림

웹소켓을 사용하기 위한 path값을 설정하고, EchoHandler에서 오버라이딩한 메서드를 설정한다.

WEB-INF/spring/appServlet/servlet-context.xml

```
<!-- websocket handler -->

<beans:bean id="myHandler" class="com.sh.handler.EchoHandler" />
<websocket:handlers>
    <websocket:mapping handler="myHandler" path="/replyEcho" />
    <websocket:handshake-interceptors>
        <beans:bean class="org.springframework.web.socket.server.support.HttpSessionHandshakeInterceptor"/>
     </websocket:handshake-interceptors>
    <websocket:sockjs websocket-enabled="true"/>
</websocket:handlers>
```

com/sh/handler/EchoHandler.java

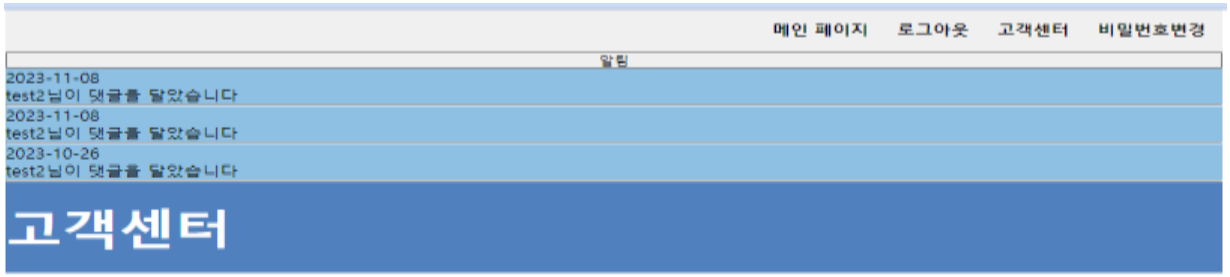
```
@Override
protected void handleTextMessage(WebSocketSession session, TextMessage message) throws Exception {
    System.out.println("handleTextMessage: " + session + " : " + message );
    logger.info("ssesion!!!!!!"+currentUserId(session));
    String msg = message.getPayload();
    logger.info("msg="+msg);

    if(StringUtils.isNotEmpty(msg)) {
        logger.info("if문 들어옴?");
        String[] strs = msg.split(",");
        if(strs != null && strs.length == 4) {
            //protocol : cmd,댓글작성자,게시글작성자,bno   (ex:reply,user2,user1,글번호(ex:12)
            String cmd = strs[0];
            String replyWriter = strs[1]; //replyWriter 댓글작성자
            String boardWriter = strs[2]; //boardWriter 글 작성자
            //String receiverEmail = strs[3];
            String bno = strs[3];

            WebSocketSession replyWriterSession = userSessionMap.get(replyWriter);
            WebSocketSession boardWriterSession = userSessionMap.get(boardWriter);
            logger.info("boardWriterSession="+userSessionMap.get(boardWriter));
            logger.info("boardWriterSession"+boardWriterSession);

            //작성자가 로그인해서 있다면
            //WebSocketSession boardWriterSession = userSessionMap.get(boardWriter);
            if ("reply".equals(cmd) && boardWriterSession != null) {
                TextMessage tmpMsg = new TextMessage(replyWriter+"님이 "
                        + "고객센터 게시판의 "+"<a href='/member/customer/get?postNo=" +bno + "'>"  + bno +"</a> 번 게시글에 댓글을 달았습니다!");
                boardWriterSession.sendMessage(tmpMsg);
```

Views/member/customer/list.jsp

```
function connetWs(){
    console.log("ttttt")
    var ws = new SockJS("/replyEcho");
    socket = ws;

    ws.onopen = function() {
        console.log('Info: connection opened. ');
    };

    ws.onmessage = function (event) {
        console.log("ReceiveMessage:", event.data+ '\n');

            let $socketAlert = $('div#socketAlert');
            $socketAlert.html(event.data);
            $socketAlert.css('display', 'block');

            setTimeout(function(){
                $socketAlert.css('display','none');

            }, 5000);

    };


    ws.onclose= function (event) {
        console.log('Info: connection closed. ');
    };

    ws.onerror = function (err) {
        console.log('Error: ', err);
    };

};
```

10.2 알림 목록(&DB저장)

알림 테이블을 만들고, 알림이 생성될 때 테이블에 알림이 저장된다.
알림을 클릭하면 알림 테이블의 read_status 컬럼의 값을 “x”에서 “o”로 변경돼서 알림 목록에서는 사라진다.



com/sh/handler/EchoHandler.java

```
//알람 데이터 생성
AlarmVO alarm = new AlarmVO();
alarm.setToId(boardWriter); //글쓴이
alarm.setFromId(replyWriter); //댓쓴이
alarm.setPostNo(Integer.parseInt(bno));
alarm.setCategory("reply");
// read_status라는 변수에 'x'라는 값을 할당
String read_status = "x";
alarm.setRead_status(read_status );

//알람 데이터 db에 저장
int result = alarmservice.insertAlarm(alarm);
if (result == 1) {
    System.out.println("알람 입력 성공");
} else {
    System.out.println("알람 입력 실패");
}
}
```

com/sh/controller/AlarmController.java

```
//알람수
@ResponseBody
@GetMapping(value = "/alarmCount")
public int alarmCount (String memberId) throws Exception{

    int alarm = alarmService.alarmCount(memberId);

    return alarm;
}


//알람목록
@ResponseBody
@GetMapping(value = "/alarmList")
public List<AlarmVO> alarmList(String memberId) throws Exception{
    System.out.println("controller:::::::::::::::::::: alarmList");
    return alarmService.alarmList(memberId);
}


//알람클릭
@ResponseBody
@PostMapping(value = "/alarmClick")
public String alarmClick(String memberId, int postNo) throws Exception{
    logger.info("알람클릭");
    alarmService.alarmClick(memberId, postNo);
     return"redirect:/member/customer/list";
}
```

10.2 알림 목록(&DB저장)

com/sh/service/AlarmService.java

com/sh/service/AlarmServiceImpl.java

```
//알람 개수 조회
public int alarmCount(String memberId) throws Exception;

//알람 목록 조회
public List<AlarmVO> alarmList(String memberId) throws Exception;

//알람 클릭
void alarmClick(String memberId, int postNo) throws Exception;

//알람 삽입 2
public int insertAlarm(AlarmVO alarm) throws Exception;
```

com/sh/mapper/AlarmMapper.java

```
int alarmCount(String memberId);

List<AlarmVO> alarmList(String memberId);

void alarmClick(String memberId, int postNo);

//알람 삽입2
void insertAlarm(AlarmVO alarm);
```

알림목록은 read_status 값이 "x"만 보이도록 한다.

alarmId	toId	fromId	postNo	category	alarmDate	read_status
23	test1	test2	44	reply	2023-10-26 15:53:37	x
25	admin	test3	45	reply	2023-10-26 17:50:59	x
26	test1	test2	46	reply	2023-11-08 21:34:48	x
27	test1	test2	47	reply	2023-11-08 21:35:42	x
18	test1	test10	43	reply	2023-10-25 19:52:14	o

Resources/com/sh/mapper/AlarmMapper.xml

```
<select id="alarmCount" resultType="int">
  select count(*) from sh_alarm where fromId = #{memberId}
</select>

<select id="alarmList" resultType="com.sh.model.AlarmVO">
  select * from sh_alarm where toId = #{memberId} order by alarmDate DESC LIMIT 6
</select>

<update id="alarmClick">
 update sh_alarm set read_status = 'o' where fromId = #{memberId} and postNo = #{postNo}
</update>
```

Views/member/customer/get.jsp

```
if(value.read_status == 'x') {
        var category = value.category ;
        a += '<div>';
        a += '<div class="small text-gray-500">'+value.alarmDate+'</div>';
        if(category == "reply"){
        a += '<span class="font-weight-bold"><a href="#"  onclick="alarmClick('+value.postNo+'
        }else if(category == "questionCheck"){
        a += '<span class="font-weight-bold"><a href="#" onclick="alarmClick('+value.postNo+',

        }
        a += '</div><hr/>';

}
```

스프링 포트폴리오 끝

감사합니다.