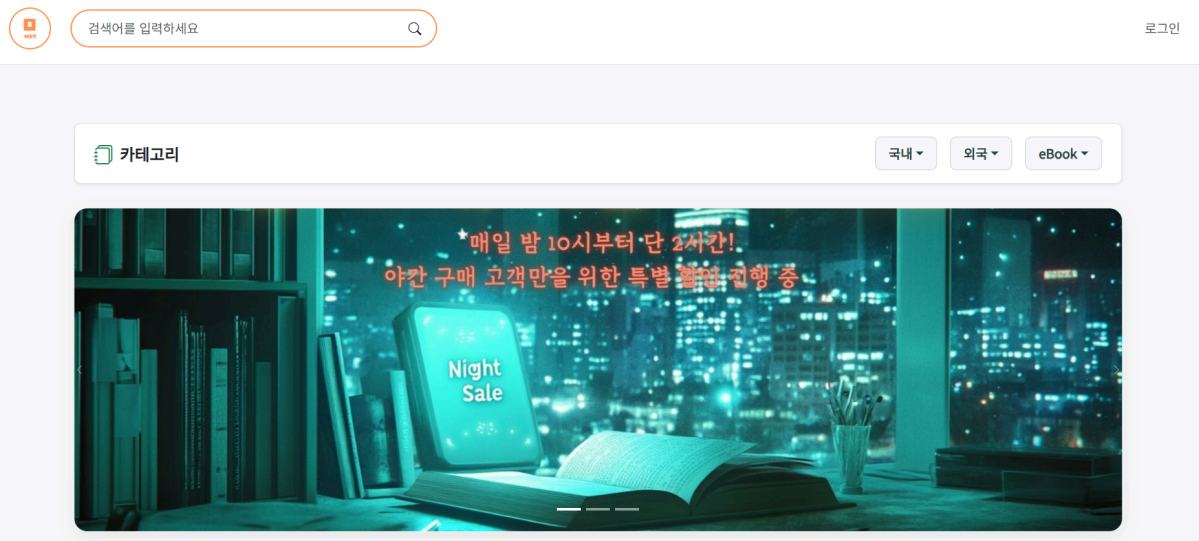


목차

1. 프로젝트 설명
 - a. 프로젝트 개요
 - b. 사용 기술 및 언어
2. **ERD**
3. 메인 페이지 추천 표시
4. 책
 - a. 상세정보
 - b. 검색
5. 장바구니
6. 주문
7. 프로젝트 느낀점

1-a. 프로젝트 개요

설명 : 출판사가 책 등록을 해서 판매하고 사용자가 사는 상황을 고려하여 만들었습니다.

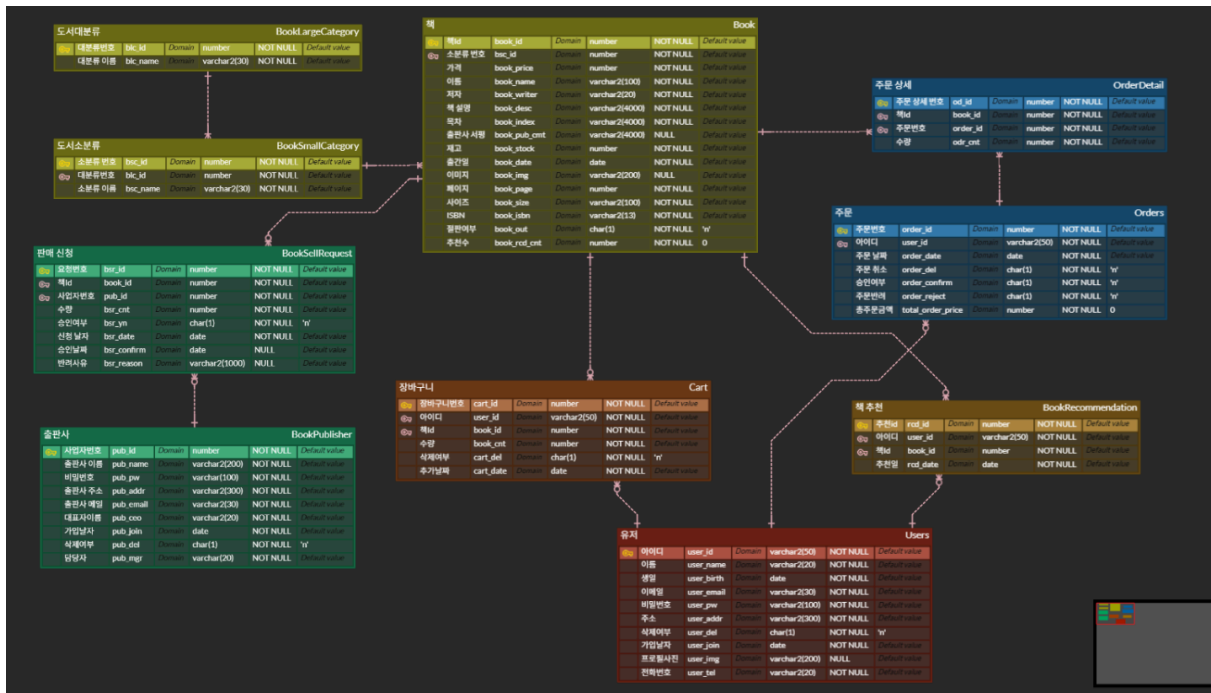


1-b. 사용 기술 및 언어

- **기술** : Spring boot, mybatis, oracle sql, java, ajax, jsp
- **언어** : SQL, JAVA, JS
- **본인역할** : 주문, 장바구니, 책 검색, 책 상세정보, 메인페이지 추천표시

2-a. ERD

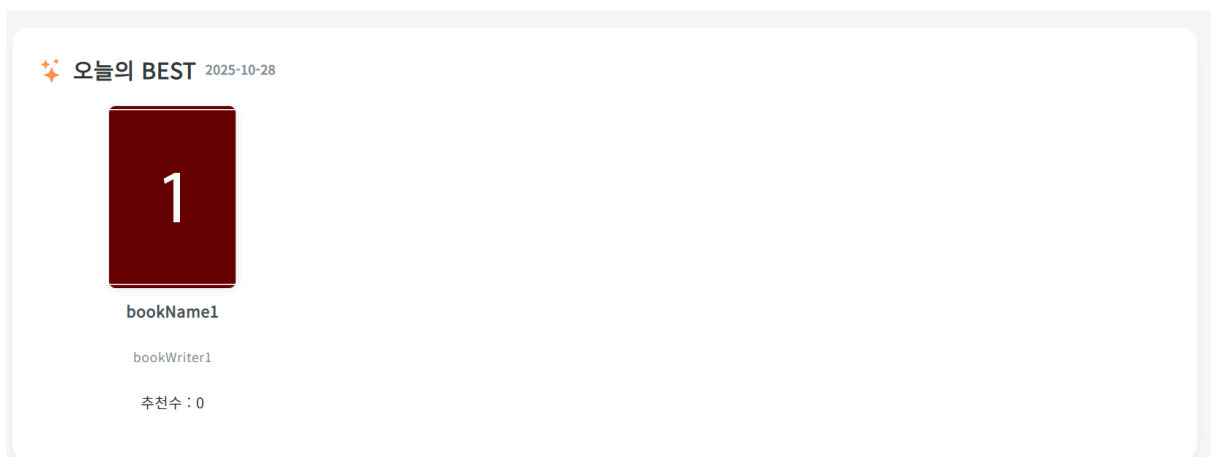
ERD 주소 : <https://www.erdcloud.com/d/CptyXbQ9G42t5ZCfi>



책추천 테이블 : 이전의 커뮤니티 프로젝트에서 게시글 추천기능(1일 1회)에 문제가 있는것을 알게 되었고 **ERD**설계가 잘못되었음을 알게 되었습니다. 그래서 위와 같이 **N:M**관계를 통한 테이블 생성을 통해 해결을 하였습니다.

3. 메인페이지 추천 표시

오늘의 베스트 : 오늘 출간된 책들 중 추천수를 기준으로 상위 5개를 표시합니다.



위의 기능을 위한 쿼리 코드는 다음과 같습니다.

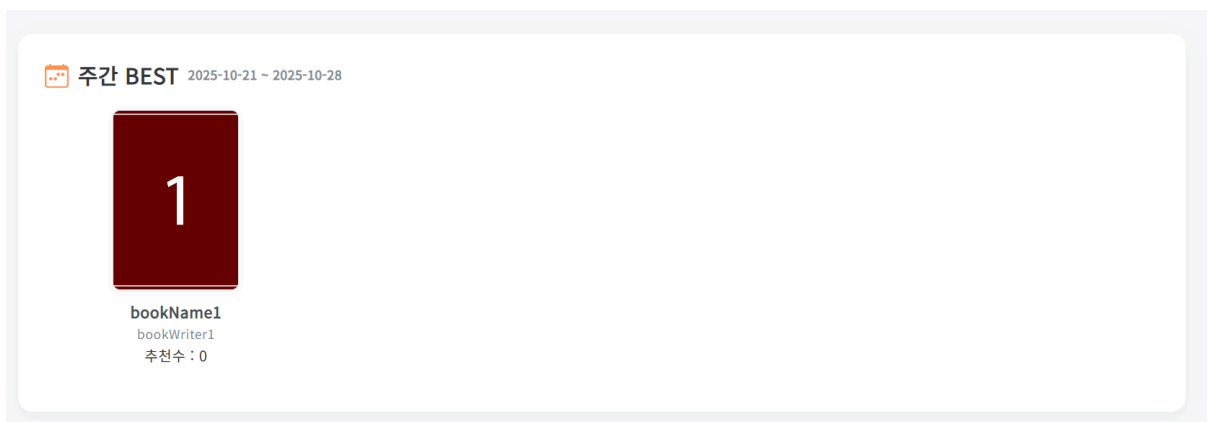
```

<select id="getTodayBestBooks" parameterType="integer" resultType="BookSG">
    select * from (
        select f_qry.*, rownum as rnum from (
            select b.book_id, b.bsc_id, b.book_price, b.book_name,
                   b.book_writer, b.book_desc, b.book_index,
                   b.book_pub_cmt, b.book_stock, b.book_date, b.book_img,
                   b.book_page, b.book_size, b.book_isbn, b.book_out, b.book_rcd_cnt,
                   bsr.bsr_id, bsr.pub_id, bsr.bsr_cnt, bsr.bsr_yn, bsr.bsr_date, bsr.bsr_confirm,
                   bsc.bsc_name,
                   blc.blc_id, blc.blc_name
            from Book b, BookSellRequest bsr, BookLargeCategory blc, BookSmallCategory bsc
            where bsr.bsr_yn = 'y' and b.book_id = bsr.book_id and b.bsc_id = bsc.bsc_id and bsc.blc_id = blc.blc_id
            and b.book_out = 'n' and b.book_stock != 0 and b.book_date like to_date(sysdate, 'YYYY-MM-DD')
            order by b.book_rcd_cnt desc, b.book_date desc, b.book_id desc
        ) f_qry
    ) where rnum between 1 and #{getCnt}
</select>

```

판매신청된 책들중 판매신청완료된 책들만 선정이 됩니다.

주간 베스트 : 현재 날짜 -7 부터 현재 날짜까지 판매신청된 책들중 추천수를 기준으로 상위 **5**개가 표시됩니다.



구현 쿼리는 다음과 같습니다.

```

<select id="getweeklyBestBooks" parameterType="integer" resultType="BookSG">
    select * from (
        select f_qry.*, rownum as rnum from (
            select b.book_id, b.bsc_id, b.book_price, b.book_name,
                   b.book_writer, b.book_desc, b.book_index,
                   b.book_pub_cmt, b.book_stock, b.book_date, b.book_img,
                   b.book_page, b.book_size, b.book_isbn, b.book_out, b.book_rcd_cnt,
                   bsr.bsr_id, bsr.pub_id, bsr.bsr_cnt, bsr.bsr_yn, bsr.bsr_date, bsr.bsr_confirm,
                   bsc.bsc_name,
                   blc.blc_id, blc.blc_name
            from Book b, BookSellRequest bsr, BookLargeCategory blc, BookSmallCategory bsc
            where bsr.bsr_yn = 'y' and b.book_id = bsr.book_id and b.bsc_id = bsc.bsc_id and bsc.blc_id = blc.blc_id
            and b.book_out = 'n' and b.book_stock != 0
            and to_date(b.book_date, 'YYYY-MM-DD') between to_date(sysdate - 7, 'YYYY-MM-DD') and to_date(sysdate, 'YYYY-MM-DD')
            order by b.book_rcd_cnt desc, b.book_date desc, b.book_id desc
        ) f_qry
    ) where rnum between 1 and #{getCnt}
</select>

```

4-a. 책 상세정보

1

book-desc-container

bookId : 1

추천수 : 0

-

수량 : 1

+

1000

원

장바구니

추천하기

book-info-container

설명 : 책에 대한 상세정보를 표시합니다.

- **수량 표시** : 책에 대한 수량을 조절하고 표시합니다. **if문**이 아닌 **효율성**을 위해 **정규식**을 사용하였습니다.

```

<button id="book-req-cnt-dec-btn" onclick="decBookReqCnt(${book.bookPrice})">-</button>
수량 : <input type="text" oninput="refreshBookTotalPrice(${book.bookPrice})" id="book-req-cnt" maxlength="3" value="1" min="1" max="999">
<button id="book-req-cnt-inc-btn" onclick="incBookReqCnt(${book.bookPrice})">+</button>
<input type="text" id="book-total-price" value="${book.bookPrice}" readonly="readonly">원

```

위의 사진에서 볼 수 있듯이 **3**개의 함수가 해당 기능을 수행합니다. **3**개의 함수는 다음의 **JS** 데이터를 참조합니다.

```

1 const bookReqCntRegex = /^(?:[1-9]|[1-9][0-9]|[1-9][0-9]{2})$/;
2 const currencyFormatter = new Intl.NumberFormat('ko-KR', {
3   style: 'currency',
4   currency: 'KRW',
5 });
6
7 let curBookReqCntInputValue = 1;

```

- **incBookReqCnt, decBookReqCnt** : 해당 버튼을 누르면 수량을 **1**개씩 조절할 수 있습니다.

```
function decBookReqCnt(unitPrice) {
  const bookReqCntInput = document.getElementById('book-req-cnt');
  const bookTotalPriceInput = document.getElementById('book-total-price');

  if (bookReqCntRegex.test(bookReqCntInput.value)) {
    if (bookReqCntInput.value !== bookReqCntInput.min) {
      bookReqCntInput.value = parseInt(bookReqCntInput.value) - 1;
      curBookReqCntInputValue = bookReqCntInput.value;
      bookTotalPriceInput.value = currencyFormatter.format(unitPrice * bookReqCntInput.value).replace('#', '').trim();
    }
  } else {
    bookReqCntInput.value = curBookReqCntInputValue;
    alert('올바른 값을 입력해주세요 최소값: 1, 최대값: 999');
    return;
  }
}

function incBookReqCnt(unitPrice) {
  const bookReqCntInput = document.getElementById('book-req-cnt');
  const bookTotalPriceInput = document.getElementById('book-total-price');

  if (bookReqCntRegex.test(bookReqCntInput.value)) {
    if (bookReqCntInput.value !== bookReqCntInput.max) {
      bookReqCntInput.value = parseInt(bookReqCntInput.value) + 1;
      curBookReqCntInputValue = bookReqCntInput.value;
      bookTotalPriceInput.value = currencyFormatter.format(unitPrice * bookReqCntInput.value).replace('#', '').trim();
    }
  }
}
```

위와 같이 정규식을 참고하여 정규식에 부합하면 해당 값으로 업데이트를 하고 부합하지 않으면 업데이트를 하지 않습니다.

- **refreshBookTotalPrice** : **oninput**이벤트를 통해 값의 변화를 체크합니다.

```
function refreshBookTotalPrice(unitPrice) {
  const bookReqCntInput = document.getElementById('book-req-cnt');
  const bookTotalPriceInput = document.getElementById('book-total-price');

  if (bookReqCntRegex.test(bookReqCntInput.value)) {
    curBookReqCntInputValue = bookReqCntInput.value;
    bookTotalPriceInput.value = currencyFormatter.format(unitPrice * bookReqCntInput.value).replace('#', '').trim();
  } else {
    bookReqCntInput.value = curBookReqCntInputValue;
    alert('올바른 값을 입력해주세요 최소값: 1, 최대값: 999');
    return;
  }
}
```

- **책 이미지** : 책의 이미지 여부를 체크합니다

```
<c:if test="${book.bookImg != null }">
  
</c:if>
<c:if test="${book.bookImg == null }">
  
</c:if>
```

위와 같이 책 이미지가 **null**이 아니면 해당 책이미지를, **null**이면 기본 책이미지를 표시합니다.

- **책 재고 및 절판** : 책 재고 및 절판을 체크합니다.

```
<div id="book-interaction-container">
  <c:if test="${book.bookOut == 'y' || book.bookStock <= 0 }">
    <div id="cant">절판되거나 재고나 없습니다</div>
  </c:if>
  <c:if test="${book.bookOut == 'n' && book.bookStock > 0 }">
    <div id="book-request-control-container">
      <button id="book-req-cnt-dec-btn" onclick="decBookReqCnt(${book.bookPrice})">-</button>
      수량 : <input type="text" oninput="refreshBookTotalPrice(${book.bookPrice })" id="book-req-cnt" maxlength="3" value="
      <button id="book-req-cnt-inc-btn" onclick="incBookReqCnt(${book.bookPrice})">+</button>
      <input type="text" id="book-total-price" value="${book.bookPrice }" readonly="readonly">원
    </div>
    <div id="cant"><button onclick="addBook(${book.bookId})">장바구니</button></div>
  </c:if>
  <div id="recommend"><button onclick="rcdBook(${book.bookId})">추천하기</button></div>
</div>
```

위와 같이 **bookOut**(책절판), **bookStock**(책수량) 등을 체크하여 해당 책에 대한 상호작용을 표시합니다.

- **가격 표시** : 수량 표시와 동일한 원리이며 원화표시 정규식을 사용하였습니다.
- **추천하기** : 해당 책을 추천할 수 있습니다. **1일 1회**만 가능합니다. 다음과 같이 **rcdBook** 함수가 실행됩니다.

```
async function rcdBook(id) {
  const bookReqCntInput = document.getElementById('book-req-cnt');
  let isBookRcdExistResult = null;
  let insertBookRcdResult = null;
  let response = null;
  const data = {
    bookId : id
  };
  let isUserResult = await isUser();

  if (isUserResult === false) {
    alert("먼저 로그인 해주세요");
    return;
  }

  isBookRcdExistResult = await isBookRcdExist(id);
  if (isBookRcdExistResult === true) {
    alert('추천은 1일 1회만 가능합니다');
    return;
  }

  insertBookRcdResult = await insertBookRcd(id);
  if (insertBookRcdResult === 1) {
    alert('추천 성공');
  } else if (insertBookRcdResult === -1) {
    alert('관리자는 추천할 수 없습니다');
    return;
  }
}
```

isUser함수를 통해 비회원을 체크하고, **isBookRcdExist**함수를 통해 책추천 테이블에 해당 책에 대한 추천여부를 체크합니다. 추천여부가 없으면 서버에서 책추천 테이블에 데이터를 넣고 **insertBookRcd**함수를 통해 해당 책의 추천수를 증가시킵니다.

- **장바구니** : 책을 장바구니에 넣는 기능입니다.

```
async function addBook(id) {
  const bookReqCntInput = document.getElementById('book-req-cnt');
  let response = null;
  let isUserResult = await isUser();
  const data = {
    bookId : id,
    bookCnt : bookReqCntInput.value
  };

  if (isUserResult === false) {
    alert("먼저 로그인 해주세요");
    return;
  }

  if (!bookReqCntRegex.test(bookReqCntInput.value)) {
    alert('올바른 값을 입력해주세요. 최소값: 1, 최댓값: 999');
    return;
  }

  try {
    response = await axios.post('/cart/addBook', data);
    if (response.data === 1) {
      alert('장바구니 담기 성공');
      return;
    } else if (response.data === 0) {
      alert('이미 장바구니에 있습니다');
    }
  }
}
```

클라이언트 에서 요청을 하면 서버는 해당 여부를 다음과 같이

체크합니다.

```
@PostMapping("/cart/addBook")
public int addBook(@RequestBody CartSG reqCart, HttpSession session) {
    int result = 0;
    String userId = (String) session.getAttribute("userId");

    Boolean isDupBookExist = cartRestService.isNotDelBookDupExist(reqCart.getBookId(), userId);

    if (isDupBookExist) {
        result = 0;
    } else {
        int total = cartRestService.getNotDelTotalByUserId(userId);
        if (total >= LIMIT_CART_NUM) {
            return -1;
        }
        else if (userId.equals("admin")) {
            return -2;
        }
        else {
            reqCart.setUserId(userId);
            int insertResult = cartRestService.insert(reqCart);

            if (insertResult > 0) {
                result = 1;
            } else {
                result = -3;
            }
        }
    }
}
```

위와 같이 서버는 중복된 책이 장바구니에 있는지, 유저인지, 최대제한 갯수등을 체크해 결과를 리턴합니다.

4-b. 책 검색



검색어를 입력하세요



로그인

검색 결과: book

<div>9</div> <div>bookName9</div> <div>저자: bookWriter9</div> <div>9000원</div> <div>상세보기</div>	<div>8</div> <div>bookName8</div> <div>저자: bookWriter8</div> <div>8000원</div> <div>상세보기</div>	<div>7</div> <div>bookName7</div> <div>저자: bookWriter7</div> <div>7000원</div> <div>상세보기</div>
<div>6</div> <div>bookName6</div> <div>저자: bookWriter6</div> <div>6000원</div> <div>상세보기</div>	<div>5</div> <div>bookName5</div> <div>저자: bookWriter5</div> <div>5000원</div> <div>상세보기</div>	<div>4</div> <div>bookName4</div> <div>저자: bookWriter4</div> <div>4000원</div> <div>상세보기</div>

설명 : 판매신청된 책들중에서 판매신청완료된 책들을 **WHERE %keyword%** 필터링을 통한뒤, 최신순으로 페이징 처리를 하여 보여주도록 하였습니다. 이를 구현한 쿼리는 다음과 같습니다.

```
<select id="getByKwOnSaleTaggedList" parameterType="map" resultType="BookSG">
  select * from (
    select f_gry.*, rownum as rnum from (
      select b.book_id, b.bsc_id, b.book_price,
        b.book_name, b.book_writer, b.book_desc, b.book_index,
        b.book_pub_cmt, b.book_stock, b.book_date,
        b.book_img, b.book_page, b.book_size, b.book_isbn, b.book_out, b.book_rcd_cnt,
        bsr.bsr_id, bsr.pub_id, bsr.bsr_cnt, bsr.bsr_yn, bsr.bsr_date, bsr.bsr_confirm,
        bsc.bsc_name,
        blc.blc_id, blc.blc_name
      from Book b, BookSellRequest bsr, BookLargeCategory blc, BookSmallCategory bsc
      where bsr.bsr_yn = 'y' and b.book_id = bsr.book_id and b.bsc_id = bsc.bsc_id and bsc.blc_id = blc.blc_id
      and b.book_name like '%' || #{keyword} || '%'
      order by b.book_out asc, b.book_date desc, b.book_id desc
    ) f_gry
  ) where rnum between #{startIndex} and #{endIdx}
</select>
```

페이징 객체화 변경 : 이전 프로젝트에선 다음과 같이 페이징 데이터를 생성한뒤에 **Dao** 쪽에서 값을 넣는방식으로 하였습니다.

```
public PagingData getPagingData(int page, int total) {
    PagingData pgData = new PagingData();
    int totalBoardNum = total;

    pgData.setTotalBoardNum(totalBoardNum);
    pgData.setLimitBoardNum(limitBoardNum);
    pgData.setLimitPageNum(limitPageNum);

    if (totalBoardNum <= 0) {
        pgData.setMaxPage(1);
    } else if (totalBoardNum % limitBoardNum == 0) {
        pgData.setMaxPage(totalBoardNum / limitBoardNum);
    } else {
        pgData.setMaxPage(totalBoardNum / limitBoardNum + 1);
    }

    if (page < 1) {
        pgData.setPage(1);
    } else if (page > pgData.getMaxPage()) {
        pgData.setPage(pgData.getMaxPage());
    }
    pgData.setPage(page);

    pgData.setStartPage(limitPageNum * ((page - 1) / limitPageNum) + 1);
    pgData.setEndPage(pgData.getStartPage() + limitPageNum - 1);
    if (pgData.getMaxPage() <= pgData.getEndPage()) {
        pgData.setEndPage(pgData.getMaxPage());
    }
}
```

하지만 이런 방식이 좋지 않다 생각을 하여 다음과 같이 생성자를 통해 객체를 생성하도록 하였고 변수명을 좀 더 포괄적인 이름으로 지었습니다.

```
@Data
public class PagingData {
    private int totalNum;
    private int limitColNum;
    private int limitRowNum;

    private int page;
    private int startPage;
    private int endPage;
    private int nextPage;
    private int prevPage;
    private int maxPage;

    private int startIdx;
    private int endIdx;

    public PagingData() {
    }

    public PagingData(int totalNum, int limitColNum, int limitRowNum, int page) {
        this.totalNum = totalNum;
        this.limitColNum = limitColNum;
        this.limitRowNum = limitRowNum;

        if (this.totalNum <= 0) {
            this.maxPage = 1;
        }
    }
}
```

그리고 다음과 같이 페이징 객체 생성을 한 줄로만 가능하도록 하였습니다.

```
PagingData pgData = new PagingData(byKwOnSaleTaggedBookTotal, 15, 10, page);
```

5. 장바구니

3	bookName3	1	단가 : 3,000원 총 합계 : 3,000원	삭제
2	bookName2	1	단가 : 2,000원 총 합계 : 2,000원	삭제
1	bookName1	1	단가 : 1,000원 총 합계 : 1,000원	삭제
1				
총 주문 금액 : 6,000원				
주문하기				

설명 : 담긴 책을 확인할 수 있는 장바구니 입니다. 장바구니 **jsp**코드는 다음과 같습니다.

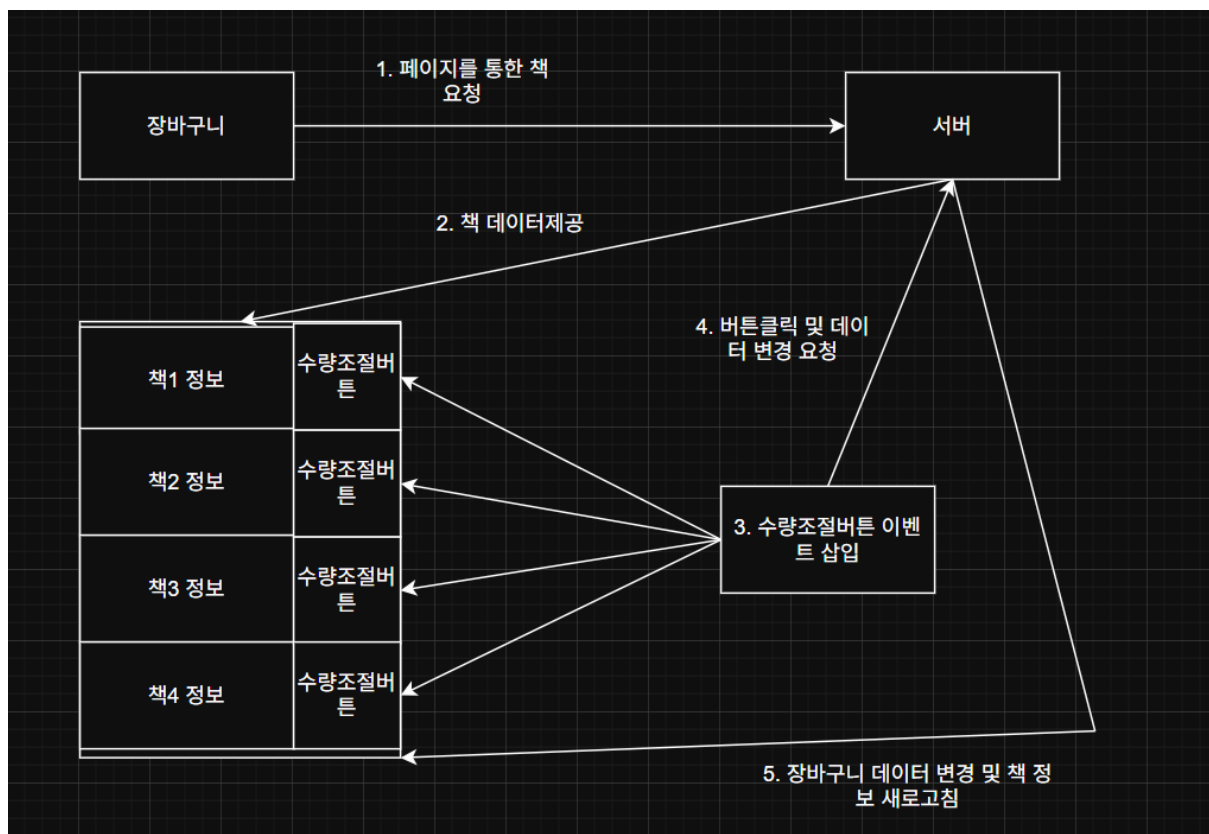
```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Insert title here</title>
    <script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
    <link rel="stylesheet" href="../../resources/sg/cart/css/style.css">
    <%@ include file="sessionChk.jsp" %>
</head>
<body>
    <div id="cart-container">
        <!-- cart-container -->
    </div>
    <div id="page-container">
        <!-- page-container -->
    </div>
    <form id="order-form-container" action="/cart/orderBooks" method="post">
        <!-- order-form-container -->
    </form>
    <script type="text/javascript" src="../../resources/sg/cart/js/script.js"></script>
</body>
</html>

```

위와 같이 비어있는데, 정적으로 처리하는게 아닌 **AJAX**을 통한 동적으로 처리하기 위해 다음과 같이 하였습니다.

장바구니 프로세스 : 아래와 같은 프로세스를 진행하게 됩니다.



간략하게 설명하자면, 특정 페이지를 요청하면 해당 페이지에 있는 책데이터를 **AJAX**을 통해 데이터를 반영합니다. 구현 코드는 다음과 같습니다.

```

• async function reqPage(page) {
•   const data = {
      page: page,
    };

    try {
      reqPageResponse = await axios.post('/cart/cartData', data);

      if(reqPageResponse.data.isSuccess !== true || reqPageResponse.data === null ||
      reqPageResponse.data === undefined || reqPageResponse.data === '') {
        await clearCartContainer();
        await clearPageContainer();
        alert('장바구니 페이지 요청 오류');
        return;
      }

      curBookReqInputValues = [];

      await clearCartContainer();
      await clearPageContainer();
      await clearOrderFormContainer();
      await insertBooks();
      await insertPages();
      await insertOrders();
    } catch (error) {
      await clearCartContainer();
      await clearPageContainer();
    }
  }

```

위와 같이 장바구니 책 컨테이너와 페이지 컨테이너를 초기화 한뒤에 책정보나 해당 버튼 및 이벤트를 삽입하게 됩니다.

6. 주문

[메인](#)
[주문신청목록](#)
[주문완료목록](#)
[주문반려목록](#)

주문번호: 23
2025-05-11

주문 도서 목록

}
bookName3
주문수량 : 1권
단가 : 3,000원
총 합계 : 3,000원

2
bookName2
주문수량 : 1권
단가 : 2,000원
총 합계 : 2,000원

1
bookName1
주문수량 : 1권
단가 : 1,000원
총 합계 : 1,000원

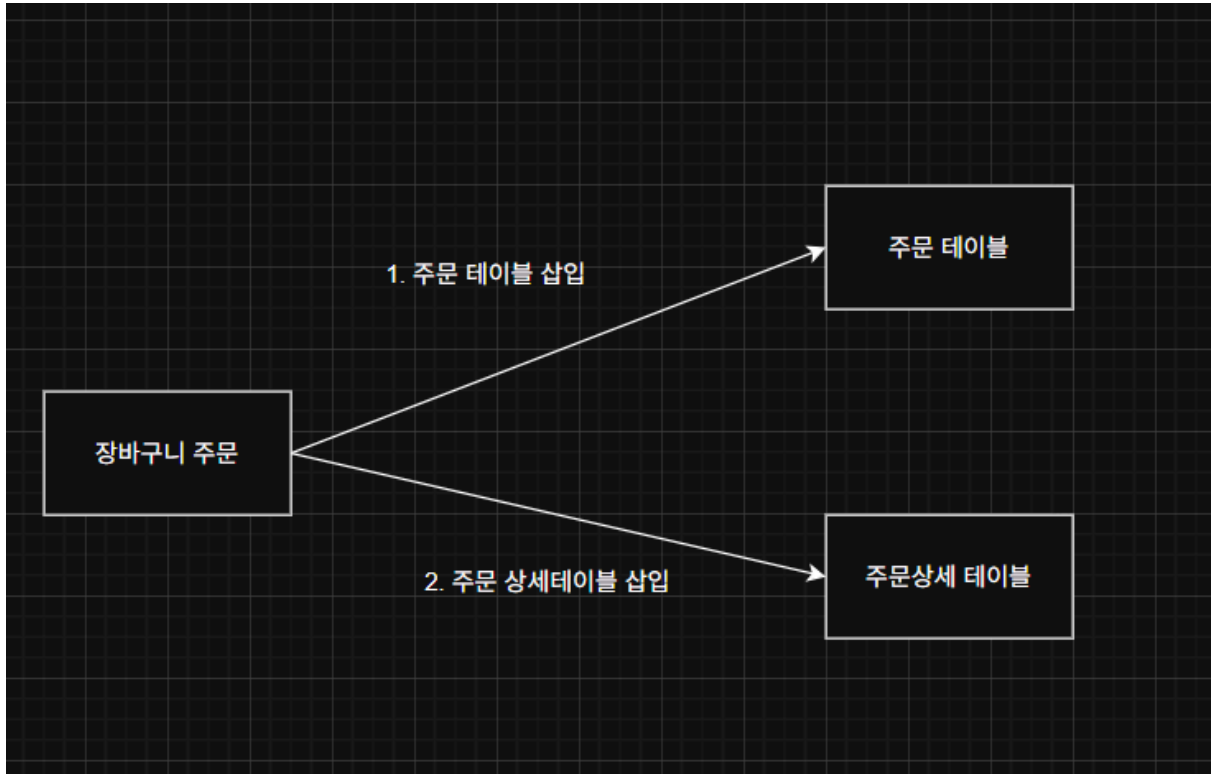
총 주문신청금액 : 6,000원

주문신청취소

주문번호: 22
2025-05-11

주문 도서 목록

주문처리과정 : 장바구니에서 주문하기 버튼을 누르면 주문이 됩니다.
이 주문버튼을 누르면 다음과 같은 프로세스를 진행합니다.



하지만 여기서 주문 **1번** 프로세스 진행 후 에러가 나서 **2번** 프로세스가 진행이 되지 않으면 데이터 정합성이 깨지게 됩니다. 따라서, **2개의 프로세스를 하나의 트랜잭션으로 처리하여 부분 오류에 대한 롤백처리를 해야한다 생각했습니다. 아래는 해당 구현 쿼리입니다.**

```
<insert id= "insertbooks" parameterType= "map">
  insert all
    into Orders values (order_seq.NEXTVAL, #{userId}, sysdate, 'n', 'n', 'n',
      (select nvl(sum(c.book_cnt * b.book_price), 0) from Cart c, Book b where b.book_id = c.book_id and user_id = #{userId} and cart_
    <foreach collection="allBookList" item="book" close="select * from dual" separator=" "
      into OrderDetail values (get_od_seq_nextval(), #{book.bookId}, order_seq.CURRVAL, #{book.bookCnt})
    </foreach>
  </insert>
```

위의 사진에서 보듯이 **INSERT ALL**문을 사용하여 주문과 주문상세 처리를 한번에 처리하도록 하였습니다.

7. 프로젝트 느낀점

설명 : 초기 **1차** 프로젝트(커뮤니티 프로젝트)보다는 보다 원활히 진행할 수 있었습니다. 초기 **1차** 프로젝트는 제가 아무래도 코딩을 해본 경험이 있어 제 역할이 많았지만 팀원분들이 **1차** 프로젝트를

진행하면서 동시에 실력이 많이 올라 저에게 많은 도움을 주어서
고맙습니다.