**백엔드 샘플 프로젝트**

* **프론트 엔드 projectEx를 프론트로 사용**
* **스프링 부트 / MyBatis 기반으로 백엔드 구현**

**프로젝트 구성 및 작성 과정**

**1. 스프링 부트 프로젝트 생성 및 설정**

**2. 프론트 엔드 - projectEx에서 복사해서 구현 (위치)**

**3. index.jsp 분할 : top/bottom/index**

**4. 로그인/로그아웃 기능 : 세션 기능 사용**

**5. 회원 : ID 중복 체크 / 우편번호찾기 (다음 API)**

**6. 상품 조회 및 주문 기능**

* **상품 조회 : 카테고리별 상품 조회 / 상품 상세 정보 조회**
* **장바구니 : 목록보기 / 추가 / 수정 / 삭제**
* **주문 처리 : 주문서 작성 / 주문 완료**

**—--------------------------------------------------------------**

**1. 스프링 부트 프로젝트 생성 및 설정**

* **프로젝트명 : spring\_boot\_project\_ex**
* **패키지명 : com.spring\_boot.projectEx**
* **설정**
  + **application.properties**
    - **포트 번호**
    - **DB 설정**
    - **mapper 위치 지정**
    - **JSP 뷰 설정**
  + **JSP 뷰 설정**
    - **pom.xml에 추가**
    - **WEB-INF 폴더 / views 폴더 생성**

**2. 프론트 엔드 - projectEx에서 복사해서 구현 (위치)**

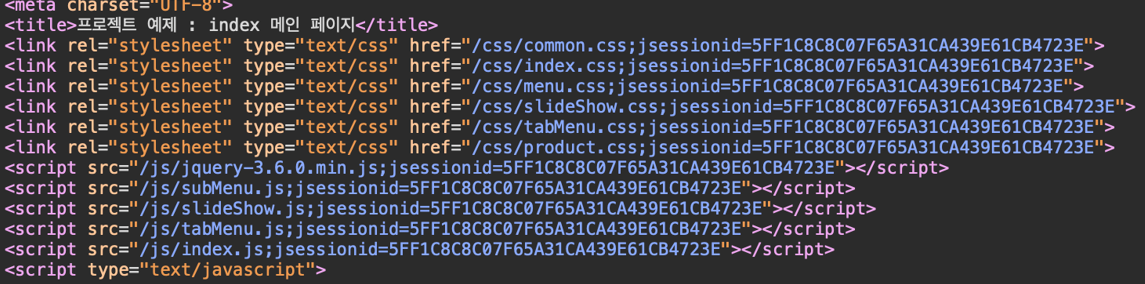
1. **projectEx에서 css/image/js 폴더 복사해서 넣기 (어디에?)**

* **src/main/resources static 폴더에 복사**

1. **views 폴더에 index.jsp 생성하고 index.html에서 전체 내용 복사**
2. **HomeController 생성하고 index 페이지 열고**
3. **실행시켜서 모든 기능이 제대로 작동되는지 확인**

* **서브메뉴**
* **슬라이드쇼**
* **탭 메뉴**
* **탑으로 이동**
* **스크롤 했을 메인메뉴 고정**

****

****

**<c:url> 사용시 jsessionid 붙는 경우**

* **처음 접속 시 cookie 사용하지 못하는 경우를 위해 자동으로 jsessionid를 붙여줌**
* **경로를 못 찾는 경우 발생**
* **jsessionid 제거**
  + **application.properties 파일에서 설정**
  + **server.servlet.session.tracking-modes=cookie**

**3. index.jsp 분할 : top/bottom/index**

* **top.jsp**
  + **<head>**
  + **<header>**
  + **<nav>**
* **bottom.jsp**
  + **<footer>**
* **index.jsp**
  + **<div id=”wrap”>**
    - **<section></section>**

**</div>**

* **index.jsp에 top과 bottom include 시키는 방법**
  + **(1) <jsp:include>**
  + **(2) <c:import>**

**=====================================**

**로그인 처리**

**쿠키와 세션**

* **클라이언트와 서버 간에 정보를 교환하는데**
* **클라이언트 PC 또는 서버의 메모리에 저장해 두고 사용하면**
* **프로그램 속도를 향상시킬 수 있음**

**웹 페이지를 연결하기 위한 방법**

* **(페이지 간에 정보를 교환하거나 데이터를 공유하기 위한 방법)**

1. **<input type=”hidden”> 태그 사용**

* **현재 페이지에 데이터를 숨겨놓고**
* **연결된 다음 페이지로 데이터를 보내는 방법**
* **두 웹 페이지가 데이터를 공유**

1. **URL Rewriting**

* **Get 방식으로 전송할 때 데이터가 URL 뒤에 붙어서**
* **다음 페이지로 전송**

1. **쿠키**

* **클라이언트의 PC의 Cookie 파일에 정보를 저장한 후 웹 페이지가 공유**

1. **세션**

* **서버 메모리에 정보를 저장한 후 웹 페이지 공유**

**쿠키 (Cookie)**

* **서버 측에서 클라이언트 측에 상태 정보를 저장하고 추출하기 위한 메커니즘**
* **서버에서 생성하여, 클라이언트 측에 저장됨**
* **서버에 요청할 때마다 쿠키의 속성 값을 참조하거나 변경**
* **크기는 4kb로 용량 제한**
* **클라이언트에 저장되므로 보안상의 문제 발생**
* **따라서 민감한 정보는 쿠키 내에 저장하지 않음**
* **쿠키는 사용자가 거부할 수 있으며 256문자 이하의 text 데이터만 저장**

**세션 (Session)**

* **클라이언트와 웹 서버 간에 네트워크로 연결이 지속적으로 유지되고 있는 상태**
* **쿠키와 마찬가지로 서버와의 관계를 유지하기 위한 수단**
* **쿠키와 달리 클라이언트에 저장되는 것이 아니라 서버 상에 객체로 존재**
* **따라서 세션은 서버에서만 접근이 가능하여 보안이 좋음**
* **서버에서 사용자의 정보를 유지 관리하는데 사용**
* **사용자 인증 후 여러 페이지에 걸쳐 정보를 공유해서 사용할 수 있도록 해 줌**
* **객체형을 포함한 거의 모든 형태의 데이터 저장도 가능**
* **Session은 서버측에서만 설정이 가능** 
  + **HttpSession session;**
* **세션은 브라우저 당 한개씩 생성**

**세션 ID**

* **클라이언트가 처음 접속하면 서버(컨테이너)로부터 유일한 ID를 부여 받게 되는데 이를 세션 ID라고 하고**
* **클라이언트가 재 접속했을 때 클라이언트 구분하기 위한 수단**

**세션 관련 메소드**

* **setAttribue(이름, 값) : 세션 이름과 값 설정**
  + **session.setAttribute(“sid”, memId);**
* **getAttribute(이름) : 이름에 해당되는 값 반환**
* **invalidate() : 실행 중인 세션 종료. 모든 데이터 삭제**
  + **로그아웃 시 사용**

**세션 설정**

* **컨트롤러에서 session.setAttribute(“sid”, memId);**

**세션 값 알아오기**

* **JSP 페이지에서 ${sessionScope.sid}**

**세션 종료 (무효화)**

* **session.invalidate()**
* **웹 브라우저 종료**
* **또는 설정된 유효 기간이 만료되면 종료**

**—--------------------------------------------------------------**

**로그인 예제**

* **스키마 : projectdb**
* **member 테이블 생성**
* **패키지 생성**
  + **controller / dao / service / model**
* **회원 클래스 및 인터페이스 파일**
  + **MembeVO**
  + **IMemberDAO**
  + **IMemberService / MemberService**
  + **MemberController**
  + **MemberMapper.xml**

**- views 폴더에 jsp 폴더 생성하고**

**- loginForm.jsp**

**- joinForm.jsp**

**- css 폴더에 join.css 저장**

**- top.jsp**

**- 로그인 변경 (링크 설정)**

**- login.js : Ajax**

**- 로그인 결과에 따라 메뉴 항목 다르게 출력 (top)**

**- 로그인 전 메뉴 항목**

**- 로그인 성공 후 메뉴 항목**

**- 로그아웃**

**- 컨트롤러 : session.invalidate();**

**======================================**

**코드 수정 작업한 후 반드시**

* **빌드 다시 / 서버 다시 시작하고**
* **웹 페이지 다시 실행**

**코드 수정 작업한 후 자동 재빌드 서버 다시 시작 설정**

* **자바 코드 수정 시 자동 빌드 및 서버 시작**
  + **pom.xml에 devtools 의존성 라이브러리 추가**
* **웹 브라우저 다시 로드**
  + **크롬 LiveReload 확장 프로그램 설치**
  + **크롬 웹스토어에서 LiveReload 검색하고 설치**

**======================================**

**주소 검색**

* **다음 API 사용 실제 주소 사용**
* **joinForm.jsp에 추가 (index.jsp에서 복사)**
  + **<div id="wrap">**
  + **TOP / BOTTOM**
  + **<link rel="stylesheet" type="text/css" href="<c:url value='/css/join.css'/>">**
  + **주소 입력 태그 추가**

**=======================================**

**상품 조회 및 주문 기능 추가**

* **메뉴에서 상품 선택 시 카테고리 별 상품 출력**
* **상세 상품 조회**
* **장바구니에 담기**
* **주문하기**

**현재 사용 중인 스키마 : projectdb**

**테이블**

* **회원 테이블 : member (이미 생성했음)**
* **상품 분류 테이블 : category**
* **상품 테이블 : product**
* **장바구니 테이블 : cart**
* **주문 테이블**
  + **order\_product (주문한 상품에 관련된 정보 저장)**
  + **order\_info (주문 관련 정보)**

**—-------------------------------------------------------------**

**1. 상품 분류 테이블 작성 : category**

**2. 상품 테이블 작성 : product**

**3. 클래스 / 인터페이스 / mapper / 뷰 페이지 작성 내용**

* **ProductVO 클래스**
* **IProductDAO 인터페이스**
* **IProductService 인터페이스 / ProductService 클래스**
* **ProductController 클래스**
* **ProductMapper.xml**
* **productListCtgView.jsp**
* **상품 관리 구현 기능**
  + **카테고리별 상품 조회**
  + **상품 상세 조회 -> 장바구니에 추가**

**3.1 클래스 / 인터페이스 / mapper 파일 추가**

* **DAO / Service 메소드 복사**

**4. top.jsp 메뉴 링크 수정 : 카테고리 메뉴로 변경**

* **subMenuItem1만 변경**

**5. 카테고리별 상품 조회 요청 처리**

* **컨트롤러 / 서비스 / mapper : 코드 작성**
* **productListCtgView.jsp 생성**
* **WebConfig.java에서 productImages 폴더 설정**

**6. 상품 상세 정보 조회**

* **컨트롤러 / 서비스 / mapper**
* **productDetailView.jsp**
  + **로그인 해야지만 주문 가능**
  + **로그인 한 경우 : [장바구니] / [주문하기] 버튼**
  + **로그인 하지 않은 경우 : [로그인]**
  + **자바스크립트**
    - **+/- 버튼으로 수량 조정**
    - **구매예정금액 계산**

**7. 장바구니 테이블 생성**

**8. 장바구니 처리 작업**

* **상품 상세 정보 조회 화면에서 [장바구니에 넣기]**
  + **저장 (insert)**
  + **장바구니에 동일 상품이 존재하면 주문수량만 증가**
  + **동일 상품 없으면 새로 추가**
* **장바구니 목록 보기 (list : select)**
  + **장바구니에서 현재 사용자의 모든 상품 출력**
* **장바구니 정보 수정 (update)**
  + **장바구니 목록에서 주문 수량 변경 시 장바구니 업데이트**
* **장바구니 삭제 (delete)**
  + **선택한 상품 삭제 (전체 선택 가능)**

**9. 장바구니에 필요한 클래스/인터페이스/뷰**

* **CartVO 클래스**
* **ICartDAO 인터페이스**
* **ICartService 인터페이스 / CartService 클래스**
* **CartController 클래스**
* **CartMapper.xml**
* **CartListView.jsp**

**10. 상품 상세 정보 조회 화면에서 [장바구니에 넣기] 클릭했을 때**

* **<input type="submit" id="insertCart" value="장바구니">**
* **<form method="post" action="<c:url value='/product/insertCart' />">**
* **VO로 받는 값 <input>**
  + **prdNo (hidden) / cartQty**
  + **전달받지 않은 memId : sid 가져와서 사용**

**11. CartController**

* **CartVO로 받고**
* **세션 sid 가져오고**
* **(1) 서비스에게 prdNo, memId 전달해서 동일 상품 여부 확인 (COUNT(\*) 값으로 반환)**
* **(2) 0 : 동일 상품 존재하지 않는 경우 - 장바구니에 추가**
* **(3) 아니면 : 존재하면 주문수량만 변경**
* **장바구니 목록(cartList) 화면으로 포워딩**

**12. 컨트롤러에 cartList 요청 처리**

**13. cartListView.jsp 생성**

****

* **id=”allCheck”**
* **id=”deleteCartBtn”**

**14. [전체 선택] 체크 박스 처리**

* **cartListView.js**
* **[전체 선택] 체크박스 체크했을 때** 
  + **모든 개별 chkDelete 체크 박스 체크되게 하고**
* **개별 체크박스 해제할 경우 [전체 선택] 체크박스 해제**

**15. 장바구니 목록에서 [삭제] 버튼 눌렀을 때 장바구니에서 해당 memId의 전체 상품 삭제**

* **[삭제] 버튼 클릭 처리 : 자바스크립트**
  + **선택 여부 확인** 
    - **선택이 없는 경우 : 선택된 상품이 없습니다**
    - **하나라도 선택한 경우**
      * **장바구니에서 삭제**
      * **삭제 후 장바구니 목록 다시 출력 (cartList 출력)**
      * **처리 후 오류 없으면 /product/cartList 요청**
      * **Ajax 사용하고 location.href로 이동**
* **컨트롤러에서 삭제 처리 : /product/deleteCart**
  + **컨트롤러에 전달되는 값 : cartNo**
    - **배열로 받아서 처리**
    - **cartNo에 해당되는 deleteCart() 작업**
    - **장바구니에서 삭제 후 return 값 1**
    - **Ajax에서 성공 시 result로 1 받아서 포워딩**
* **(1) [삭제] 버튼 클릭 시 처리**
  + **cartListView.js에 자바스크립트 추가**
  + **Ajax로 전송**
* **(2) 컨트롤러에서 배열로 받음**
  + **ArrayList<String> chkArr**

**—---------------------------------------------------------------**

**주문 처리 작업**

* **장바구니 목록에서 [주문하기] 버튼 눌렀을 때**
  + **주문서 작성 폼**
    - **자동 출력되는 부분**
      * **주문자 정보 : 주문자명(회원명) / 연락처**
      * **수령인 정보**
        + **받는분 : 회원명 (수정 가능)**
        + **연락처 : 회원 전화번호 (수정 가능)**
        + **배송지 주소 : 회원 주소 (수정 가능)**
        + **배송 메시지 : select**
        + **변가능한 입력 폼으로 작성**
      * **주문 상품 목록** 
        + **상품번호, 상품명, 사진, 가격, 주문수량, 주문예정금액, 총구매예정금액**
    - **추가 : 결제 정보 (결제 방법 선택 (라디오버튼))**
  + **주문 완료 (주문 정보 DB 저장)** 
    - **주문번호**
* **주문서 폼에서 [완료] 버튼 눌렀을 때**
  + **DB 저장** 
    - **주문번호 생성** 
      * **오늘날짜시분초\_랜덤숫자4**
    - **order\_info : 모든 주문 정보**
    - **order\_product : 주문 상품 내용 저장**
  + **완료 페이지 출력 : orderCompleteView.jsp**

**—--------------------------------------------------------------**

**주문 테이블 생성**

* **order\_info**
  + **주문번호, 주문일, 회원ID, 수령인, 수령인주소(우편번호, 주소1, 주소2), 수령인 전화, 메시지, 결제방법**
* **order\_product**
  + **주문번호, 상품번호, 주문수량**

**테이블을 나누는 이유 : 데이터 중복 때문**

**1개의 테이블로 한 경우**

**1 1001 5 2022-01-01 abcd 홍길동 12345 서울 강남 1번지 010-1111-1111 신용카드**

**1 1002 1 2022-01-01 abcd 홍길동 12345 서울 강남 1번지 010-1111-1111 신용카드**

**1 1003 2 2022-01-01 abcd 홍길동 12345 서울 강남 1번지 010-1111-1111 신용카드**

**1 1007 7 2022-01-01 abcd 홍길동 12345 서울 강남 1번지 010-1111-1111 신용카드**

**문제 : 중복 데이터 발생**

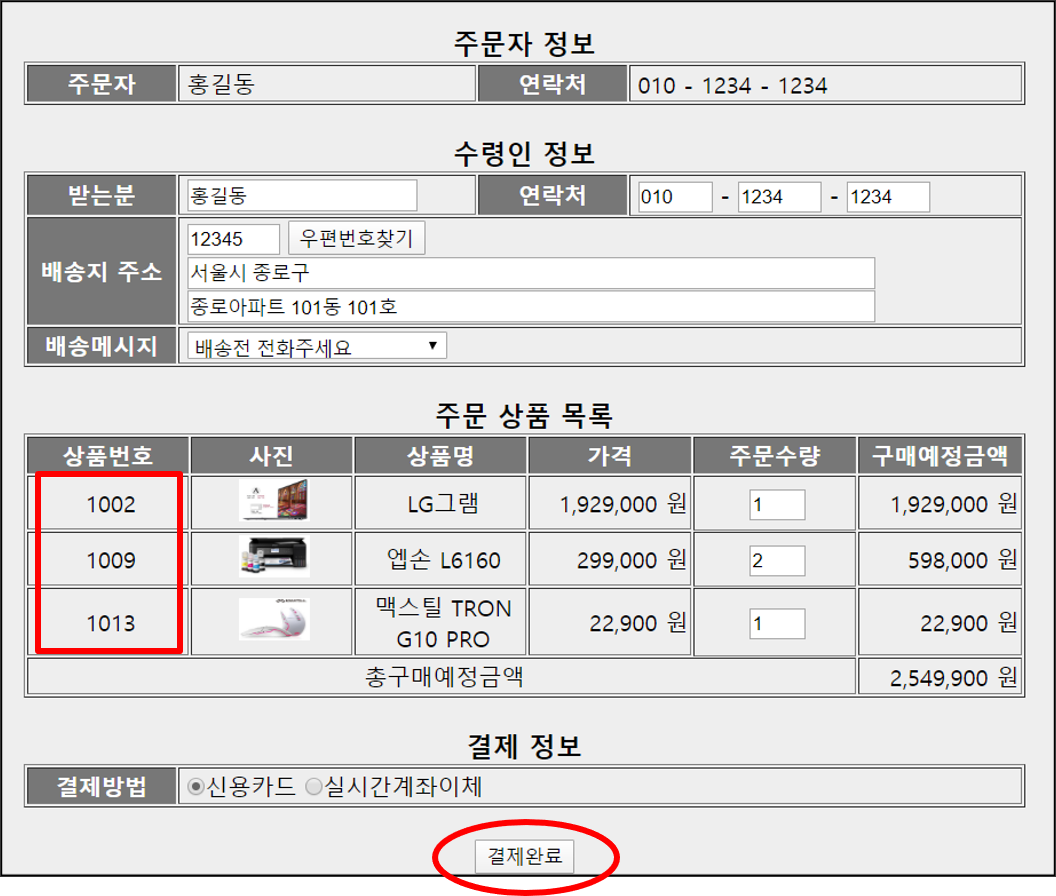
**해결 : 중복 발생 최소화하면서 테이블 분리 (정규화)**

* **1 1001 5**
* **1 1002 1**
* **1 1003 2**
* **1 1007 7**
* **1 2022-01-01 abcd 홍길동 12345 서울 강남 1번지 010-1111-1111 신용카드**

**—------------------------------------------------------------**

**주문 테이블 생성**

**—------------------------------------------------------------**

****

**view 페이지 구조**

**<body>**

**<div id=”wrap”>**

**TOP**

**<secion>**

**모든 내용**

**</section>**

**BOTTOM**

**</div>**

**</body>**

**$(document).ready(function(){**

**// [전체 선택] 체크박스 체크했을 때**

**$('#allCheck').on('click', function(){**

**// [전체 선택] 체크박스 상태 확인**

**var chk = $('#allCheck').prop("checked");**

**if(chk) { // true : 체크되었다면**

**$('.chkDelete').prop("checked", true);**

**} else {**

**$('.chkDelete').prop("checked", false);**

**}**

**});**

**// 개별 체크박스 (.chkDelete) 하나라도 해제할 경우 [전체선택] 체크박스 해제**

**// 모두 체크되었을 때 [전체선택] 체크**

**/\***

**$('.chkDelete').on("click", function(){**

**var count = 0;**

**$('.chkDelete').each(function(){**

**if($(this).is(':checked'))**

**count++;**

**});**

**if(count == $('.chkDelete').length) // 체크된 수가 전체 개수라면**

**$('#allCheck').prop("checked", true);**

**else**

**$('#allCheck').prop("checked", false);**

**});**

**\*/**

**$(".chkDelete").click(function(){**

**var total = $(".chkDelete").length; // 전체 체크박수 개수**

**var checked = $(".chkDelete:checked").length; // 체크된 체크박스 개수**

**if(total != checked) // 두 수가 같다면 (모두 선택된 상태가 아니라면)**

**$("#allCheck").prop("checked", false); // 해제**

**else // 모두 선택했으면**

**$("#allCheck").prop("checked", true); // 체크**

**});**

**});**