

# Java기반 응용SW 개발자 양성 과정



Spring with Mybatis 연습  
온라인 쇼핑몰 구축

# 온라인 쇼핑몰 웹 애플리케이션

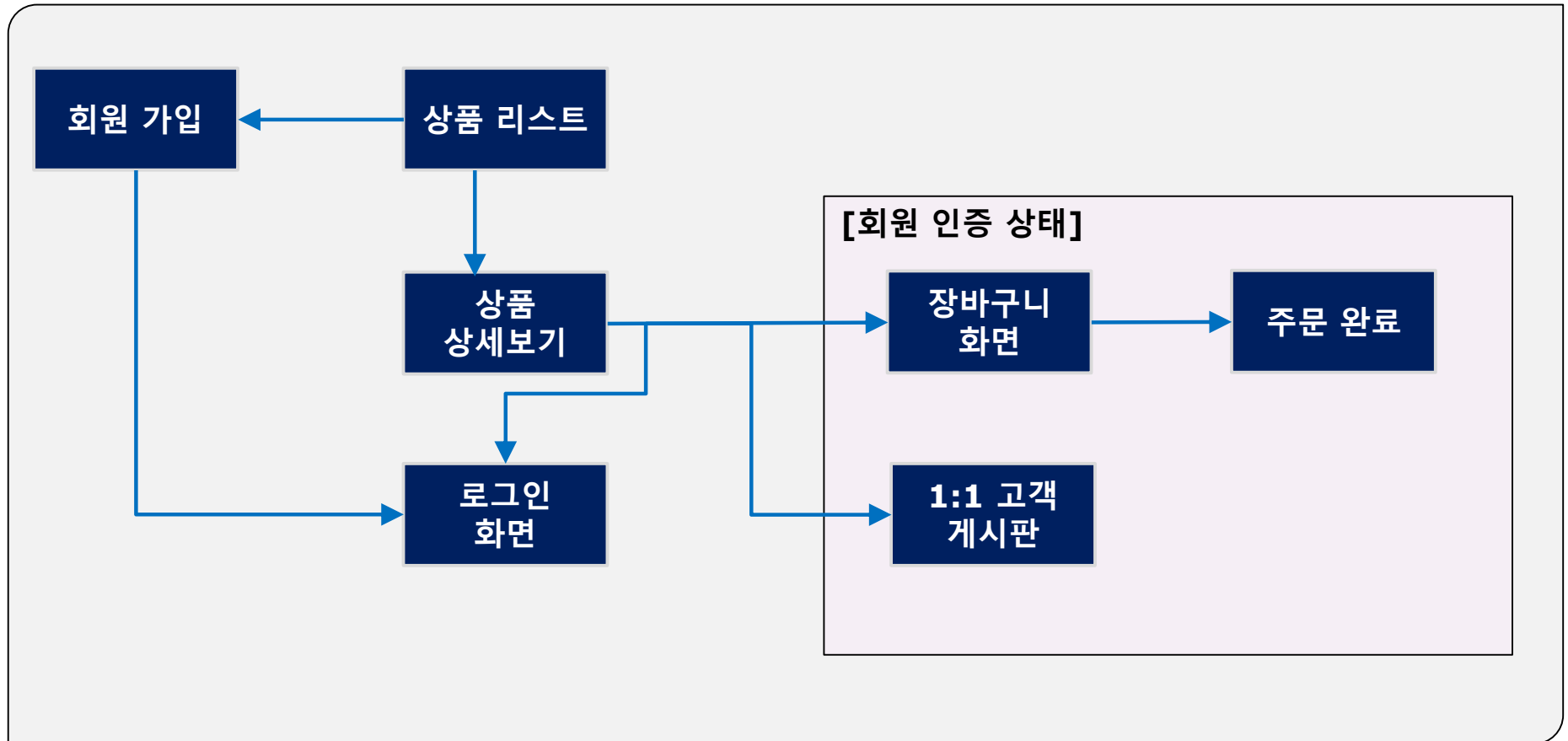
## □ 프로젝트 개요

- 신발(구두)를 판매하는 전문 쇼핑몰
- 쇼핑몰 이름: Nonage Shop
- 프로젝트명: ShopMall

## □ 쇼핑몰의 기능

사용자	관리자
<div>1. 회원 가입하기</div> <div>2. 로그인 하기</div> <div>3. 상품 검색하기</div> <div>4. 상품 상세보기</div> <div>5. 장바구니에 상품 담기</div> <div>6. 장바구니 확인하기</div> <div>7. 주문하기</div> <div>8. 주문 내역보기</div> <div>9. Q&amp;A 게시판 이용하기</div> <div>로그인 없이 사용 가능한 기능</div>	<div>1. 상품 관리하기</div> <div>2. 주문 내역 조회하기</div> <div>3. Q&amp;A 게시판에 답변 달기</div>

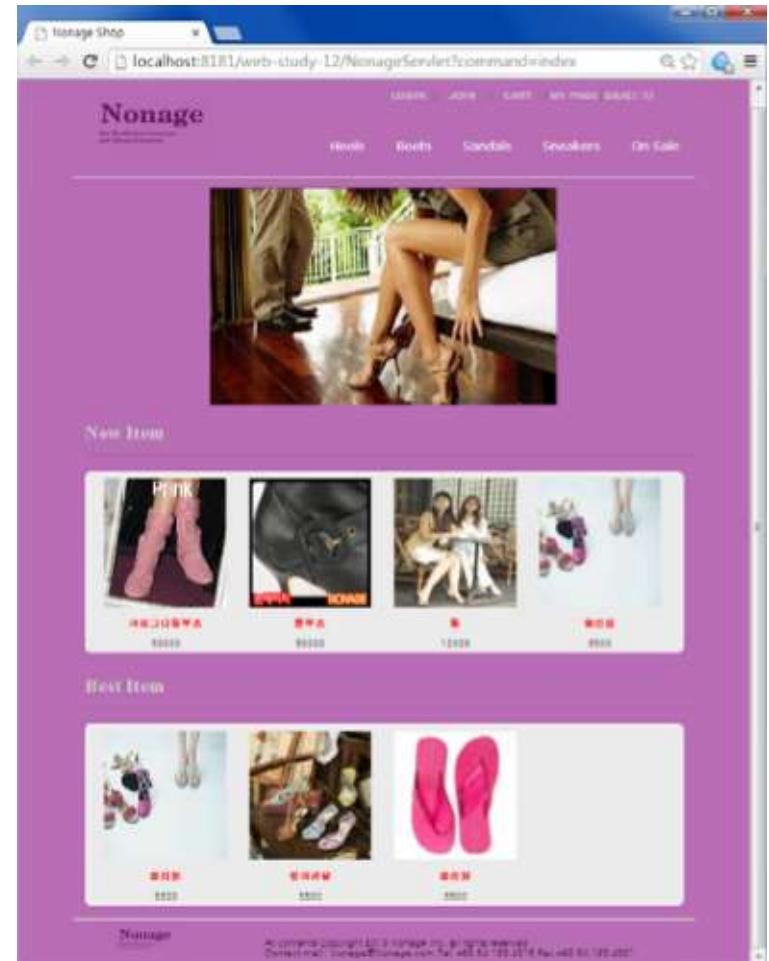
# 쇼핑몰 화면 흐름



# 주요 화면

## □ 메인 화면

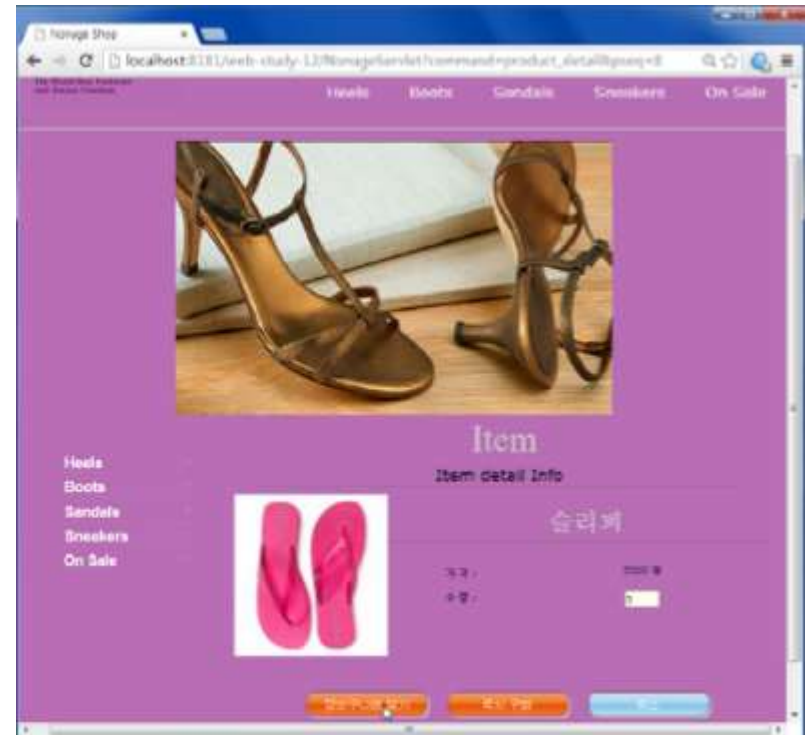
- 신상품 과 인기상품을 나열한다.
- 특정 상품을 클릭하면 상품 상세보기 화면으로 이동한다.



# 주요 화면

## □ 상품 상세 화면

- 상품 이미지와 상품명, 상품 가격이 표시된다.
- 로그인이 되어 있으면 상품을 장바구니에 담거나 즉시 구매 할 수 있다.



# 주요 화면

## □ 회원가입 화면

Nonage Shop

localhost:8181/web-study-12/NonageServlet?command=join\_form

Nonage Shop

Login

Join us

### Join Us

#### Basic Info

User ID: pinkisung

Password: \*\*\*\*

Retype Password: \*\*\*\*

Name: 성원정

E-Mail: pinkisubin@nate.com

#### Optional

Zip Code: 138-790

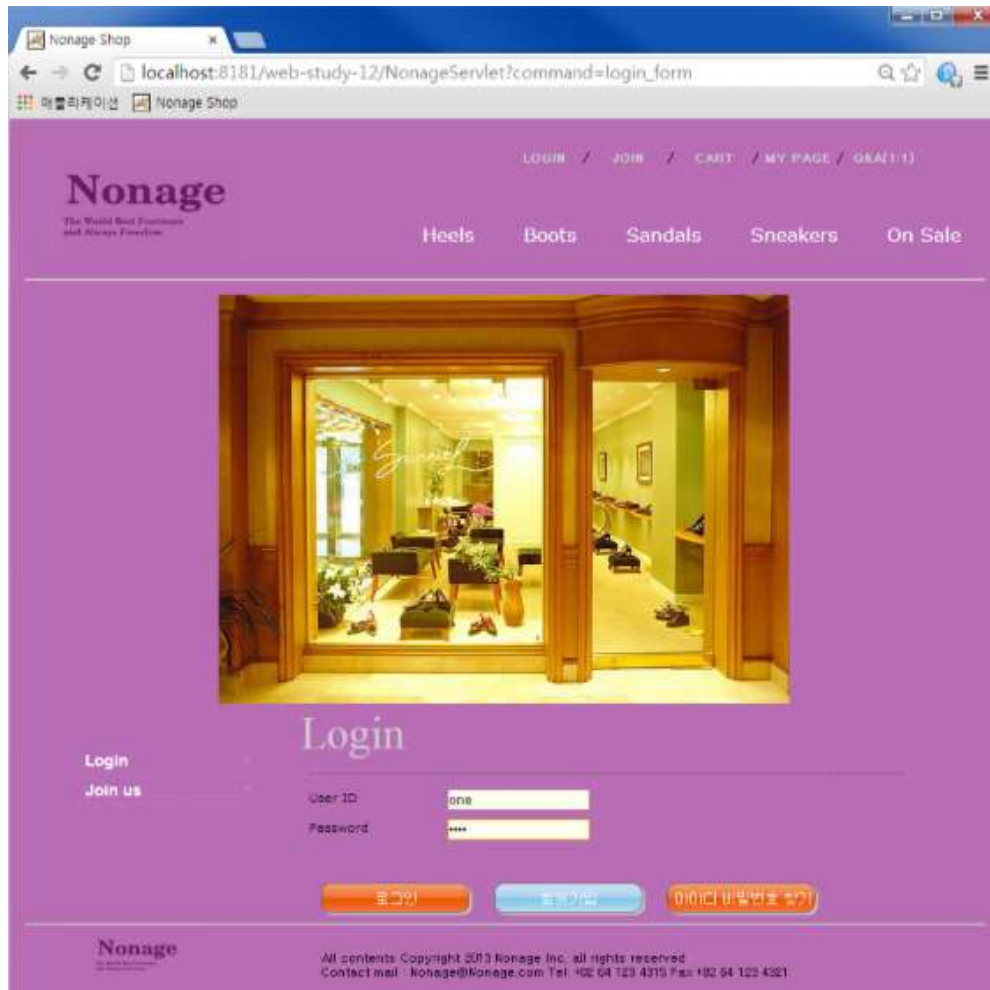
Address: 서울 송파구 잠실3동 트리치움 아파트 100동 1001호

Phone Number: 010-2321-2312

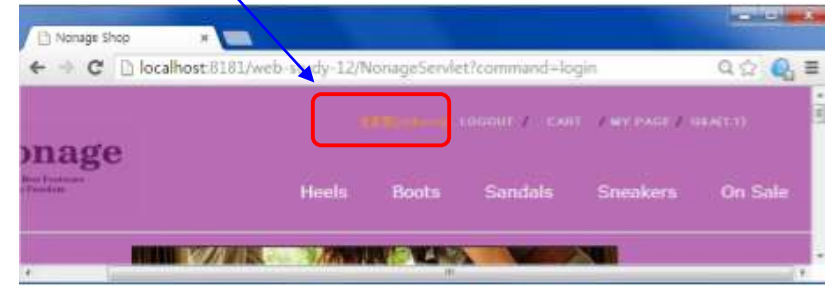
# 주요 화면

## 로그인

- 회원 가입이 완료되면 로그인 화면 출력



로그인 완료 시 사용자명 표시



# 쇼핑몰 데이터베이스

**[주문 상세]**

주문상세번호
주문번호(FK) 상품번호(FK) 주문수량 처리완료여부

**[주문]**

주문번호
주문자 아이디(FK) 주문일

**[Q&A게시판]**

글번호
제목 문의내용 답변내용 작성자아이디 답변유무 작성일

**[상품]**

상품번호
상품명 상품종류 원가 판매가 수익 상품내용 상품이미지 상품삭제여부 베스트 상품 여부 등록일

**[장바구니]**

장바구니 번호
회원아이디(FK) 상품번호(FK) 수량 처리완료여부 등록일

**[회원]**

회원아이디
회원암호 회원이름 회원이메일 우편번호 주소 전화번호 탈퇴여부 가입일

**[주소]**

우편번호 시도 구군 동 우편코드 번지
-------------------------------------

**[관리자]**

관리자
관리자 암호 관리자 이름 전화번호



# 테이블

□ 엔티티명:회원 테이블명: member

NO	속성명	칼럼명	크기	NULL허용	키	디폴트값	비고
1	회원아이디	id	varchar2(20)	N	PK		
2	회원암호	pwd	varchar2(20)				
3	회원이름	name	varchar2(40)				
4	회원이메일	email	varchar2(40)				
5	우편번호	zip_num	varchar2(7)				
6	주소	address	varchar2(100)				
7	전화번호	phone	varchar2(20)				
8	탈퇴여부	useyn	char			y	y:사용가능 n: 탈퇴
9	가입일	regdate	date			sysdate	

# 테이블

□ 엔티티명: 상품    테이블명: product

NO	속성명	칼럼명	크기	NULL허용	키	디폴트값	비고
1	상품번호	pseq	numbr(5)	N	PK		product_seq 시퀀스 객체로 자동 번호 부여
2	상품명	name	varchar2(100)			0	
3	상품종류	kind	char(1)				1:힐, 2:부츠, 3:샌달 4:슬리퍼, 5:스니커즈
4	원가	price1	number(7)			0	
5	판매가	price2	number(7)			0	
6	판매가-원가	price3	number(7)			0	
7	상품내용	content	varchar2(1000)			NULL	
8	상품이미지	image	varchar2(50)			default.jpg	
9	상품삭제여부	useyn	char(1)			y	상품 사용유무 체크 y:사용가능, n:사용불 가능
10	베스트상품여부	bestyn	char(1)			n	y:베스트 상품 n:베스트 상품 아님
11	등록일	regdate	date			sysdate	

# 테이블

□ 엔티티명: 관리자    테이블명: admin

NO	속성명	칼럼명	크기	NULL허용	키	디폴트값	비고
1	아이디	id	varchar2(20)	N	PK		
2	암호	pwd	varchar2(20)				
3	이름	name	varchar2(40)				
4	전화번호	phone	varchar2(20)				

□ 엔티티명: 주소    테이블명: address

NO	속성명	칼럼명	크기	NULL허용	키	디폴트값	비고
1	우편번호	zip_num	varchar2(7)				
2	시도	sido	varchar2(30)				
3	시군	gugun	varchar2(30)				
4	동	dong	varchar2(100)				
5	우편코드	zip_code	varchar2(30)				
6	번지	bunji	varchar2(30)				

# 테이블

□ 엔티티명: 장바구니      테이블명: cart

NO	속성명	칼럼명	크기	NULL허용	키	디폴트값	비고
1	장바구니 번호	cseq	number(10)	N	PK		cart_seq 시퀀스 객체로 자동 일련번호 부여
2	회원 아이디	id	varchar2(20)		FK		member 테이블의 기본 키인 id컬럼
3	상품번호	pseq	number(5)		FK		product 테이블의 기본 키인 pseq 컬럼
4	수량	quantity	number(5)			1	
5	처리완료여부	result	char(1)			1	1:미처리, 2:처리
6	등록일	indate	date			SYSDATE	

# 테이블

□ 엔티티명: 주문      테이블명: orders

NO	속성명	칼럼명	크기	NULL허용	키	디폴트값	비고
1	주문번호	oseq	number(10)	N	PK		orders_seq 시퀀스 객체로 자동 일련번호 부여
2	주문자 아이디	id	varchar2(20)		FK		member 테이블의 기본 키인 id 컬럼
3	주문일	indate	date			SYSDATE	

□ 엔티티명: 주문 상세      테이블명: order\_detail

NO	속성명	칼럼명	크기	NULL허용	키	디폴트값	비고
1	주문상세번호	odseq	number(10)	N	PK		oder_detail_seq 시퀀스 객체로 자동 일련번호 부여
2	주문번호	oseq	number(10)		FK		orders 테이블의 기본 키인 oseq 컬럼
3	상품번호	pseq	number(5)		FK		product 테이블의 기본 키인 pseq 컬럼
4	주문수량	quantity	number(5)				
5	처리여부	result	char(1)			1	1: 미처리, 2:처리

# 테이블

□ 엔티티명: QNA 게시판      테이블명: qna

NO	속성명	칼럼명	크기	NULL허용	키	디폴트값	비고
1	글번호	qseq	number(5)	N	PK		qna_seq 시퀀스 객체로 자동 일련번호
2	제목	subject	varchar2(30)				
3	문의 내용	content	varchar2(1000)				
4	답변 내용	reply	varchar2(1000)				
5	작성자아이디	id	varchar2(20)		FK		member 테이블의 기본키인 id 컬럼
6	답변유무	rep	char(1)			1	1: 답변 무, 2: 답변 유
7	작성일	indate				SYSDATE	

# 프로젝트 만들기

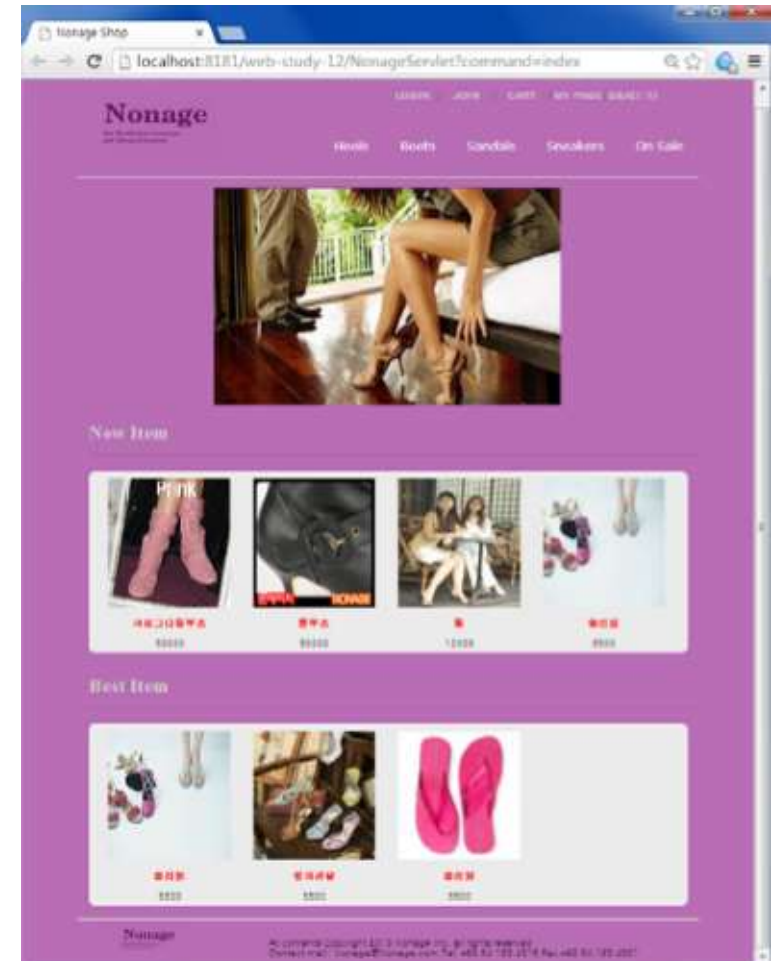
---

- 프로젝트명: ShopMall
  - 프로젝트 설정
    - Java Build Path
    - 프로젝트 Properties
    - pom.xml
    - web.xml
  - 프로젝트에서 사용할 이미지 파일 준비
    - images 폴더
    - product\_images 폴더

# 화면 구현

## □ 메인 화면

- 소스파일명: index.jsp
- 상단 메뉴 구성파일: header.jsp
- 사이트 정보 출력 파일: footer.jsp
- 스타일 시트: shopping.cs





# 상품정보를 저장하는 VO 클래스

---

## □ ProductVO 클래스

- 패키지: com.ezen.biz.dto

```
public class ProductVO {  
    private int pseq;  
    private String name;  
    private String kind;  
    private int price1;  
    private int price2;  
    private int price3;  
    private String content;  
    private String image;  
    private String useyn;  
    private String bestyn;  
    private Timestamp regdate;  
  
    // Getters, Setters  
}
```

# 상품 정보 Mybatis 매핑

---

- Name space: “ProductMapper”

```
// 신상품 목록 얻어오기: “getNewProductList”  
“select * from new_pro_view”;
```

```
// 베스트 상품 목록 얻어오기: “getBestProductList”  
“select * from best_pro_view”;
```

```
// 상품번호로 하나의 상품정보 얻어오기: “getProduct”  
“select * from product where pseq=?”;
```

```
// 상품 종류별 상품 목록 얻어오기: “getProductListByKind”  
“select * from product where kind=?”;
```

# 상품 정보 액세스하는 DAO 클래스

---

## □ ProductDAO 클래스

```
public class ProductDAO {  
    // 신상품 목록 얻어오기  
    public List<ProductVO> getNewProductList(ProductVO vo) {  
    }  
  
    // 베스트 상품 목록 얻어오기  
    public List<ProductVO> getBestProductList(ProductVO vo) {  
    }  
  
    // 상품번호로 하나의 상품정보 얻어오기  
    public ProductVO getProduct(ProductVO vo) {  
    }  
  
    // 상품 종류별 상품 목록 얻어오기  
    public List<ProductVO> getProductListByKind(String kind) {  
    }  
}
```

# 상품 정보 관련 업무 모듈 구현

---

- ProductService 인터페이스
  - ProductVO getProduct(ProductVO vo)
  - List<ProductVO> getProductList();
  - List<ProductVO> getNewProductList();
  - List<ProductVO> getBestProductList();
  - List<ProductVO> getProductListByKind();
  
- ProductServiceImpl 클래스 구현
  - ProductService 인터페이스 구현

# 메인 화면 출력 컨트롤러 구현

---

## □ HomeController 클래스

- 요청 문자열: /index

## □ 액션

- productService를 생성하고 신규상품 조회를 수행하는 `getNewProductList()`를 호출하여 결과 목록을 `ArrayList<ProductVO>` 타입의 변수에 저장한다.
- request 파라미터에 “newProductList”를 키로 위의 조회 결과를 설정한다.
- 베스트 상품 조회를 수행하는 `getBestProductList()`를 호출하여 결과 목록을 `ArrayList<ProductVO>` 타입의 변수에 저장한다.
- request 파라미터에 “bestProductList”를 키로 위의 조회 결과를 저장한다.
- “index.jsp” 화면을 호출한다.

# 제품 상세보기 페이지

---

메인 화면에서 특정 상품을 클릭하면 상품 상세보기 화면으로 이동한다.

- **ProductController** 클래스를 작성하고 **productDetailAction()** 메소드를 추가한다.

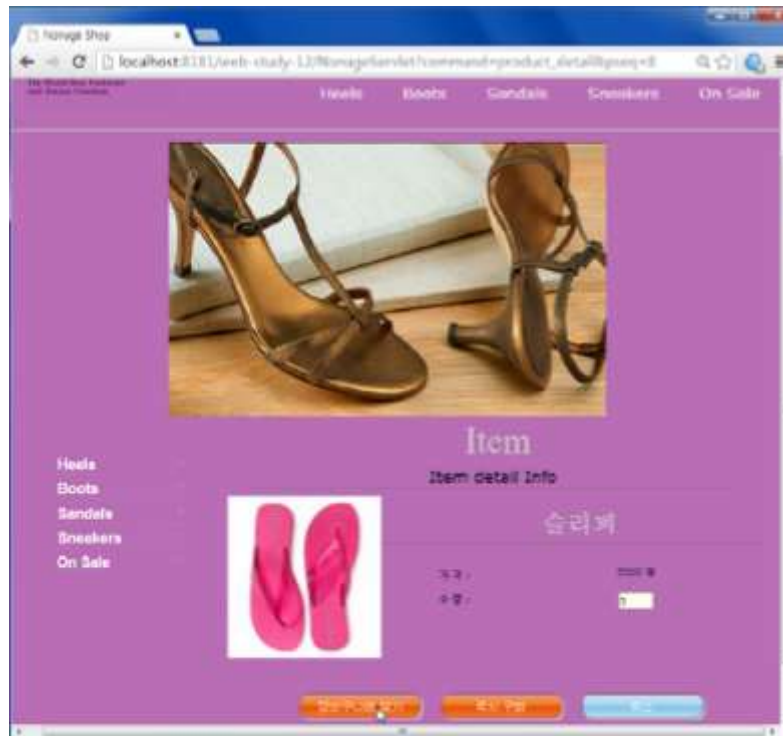
- 요청 문자열: **product\_detail?pseq** (pseq는 상품 일련번호)

- **액션**

- **request** 파라미터에서 “pseq”를 읽어온다.
- **productService**에서 **getProduct()**를 호출하여 상품 정보를 조회한다.
- **request** 파라미터에 “productVO”를 키로 위에서 조회한 상품을 저장한다.
- “product/productDetail.jsp” 화면을 호출한다.

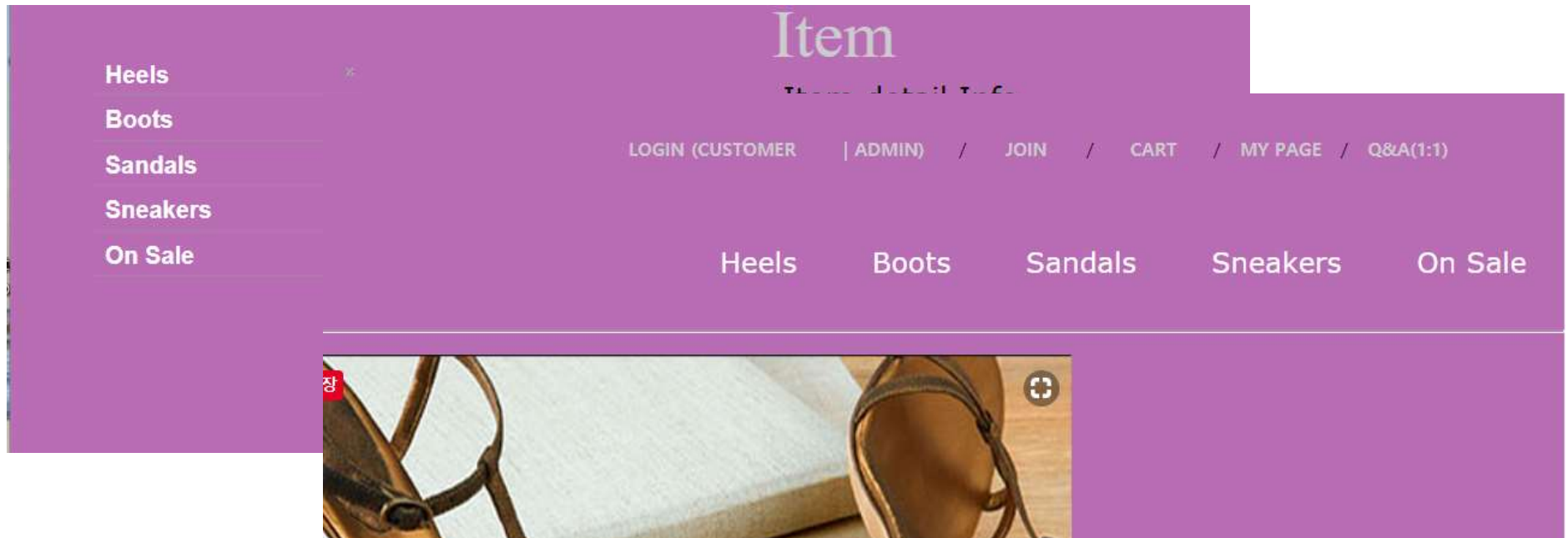
# 화면 구현

## □ 상품 상세 화면



# 메뉴별 페이지 만들기

- 화면 상단에 출력되는 주 메뉴와 제품 상세보기 페이지의 좌측 서브 메뉴에서 카테고리가 나타난다.
- 특정 카테고리를 선택하면 상품 분류별 목록을 출력해주는 화면으로 이동한다.
  - 요청 문자열: category?kind=1





# 분류별 상품 목록 화면 보이기

---

## □ ProductController

### ■ 메소드명: ProductKindAction()

- productService 객체에서 listKindProduct(kind) 메소드를 호출한다.
- 조회 결과를 ArrayList<ProductVO> 타입의 변수에 저장한다.
- 조회 결과는 request 파라미터의 “productKindList”를 키로 저장한다.
- “product/productKind.jsp”화면을 호출한다.

---

# 회원 가입 및 로그인

# 회원 정보 VO 클래스

---

## □ MemberVO 클래스

```
public class MemberVO {  
    private String id;  
    private String pwd;  
    private String name;  
    private String email;  
    private String zip_num;  
    private String address;  
    private String phone;  
    private String useyn;  
    private Date regdate;  
  
    // Getters , Setters  
}
```

# 회원 정보 Mybatis 매핑

---

## □ Name space: “MemberMapper”

```
// 회원 id를 조건으로 회원 조회: getMember  
“select * from member where id=?”;
```

```
// 회원 존재 여부 조회:confirmID  
“select pwd from member where id=?”;
```

```
// 회원 등록: insertMember  
insert into member (id, pwd, name, email, zip_num, address, phone)  
+ “values (?, ?, ?, ?, ?, ?, ?)”;
```

# 회원 정보 DAO 클래스

---

## □ MemberDAO

```
public class MemberDAO {  
    // 회원 id를 조건으로 회원 조회  
    public MemberVO getMemeber(String id) {  
    }  
  
    // 회원 존재 여부 조회  
    public int confirmID(String userid) {  
        if (회원이 존재할 경우)  
            return 1;  
        else  
            return -1;  
    }  
  
    public int insertMember(MemberVO memberVO) {  
    }  
}
```

# 회원 정보 관련 업무 모듈 구현

---

- MemberService 인터페이스
  - Void insertMemeber(MemberVO vo);
  - MemberVO getMember(MemberVO vo);
  - List<MemberVO> getMemberList();
  
- MemberServiceImpl 클래스 구현
  - MemberService 인터페이스 구현

# 회원 관련 화면 구현

---

- 서브 페이지에서 이미지 출력
  - member/sub\_img.jsp
- 서브 페이지에서 좌측 메뉴
  - member/sub\_menu.jsp
- 약관 동의 페이지 구현
  - member/contract.jsp
- 회원 가입 화면
  - join.jsp

# 회원 가입 컨트롤러

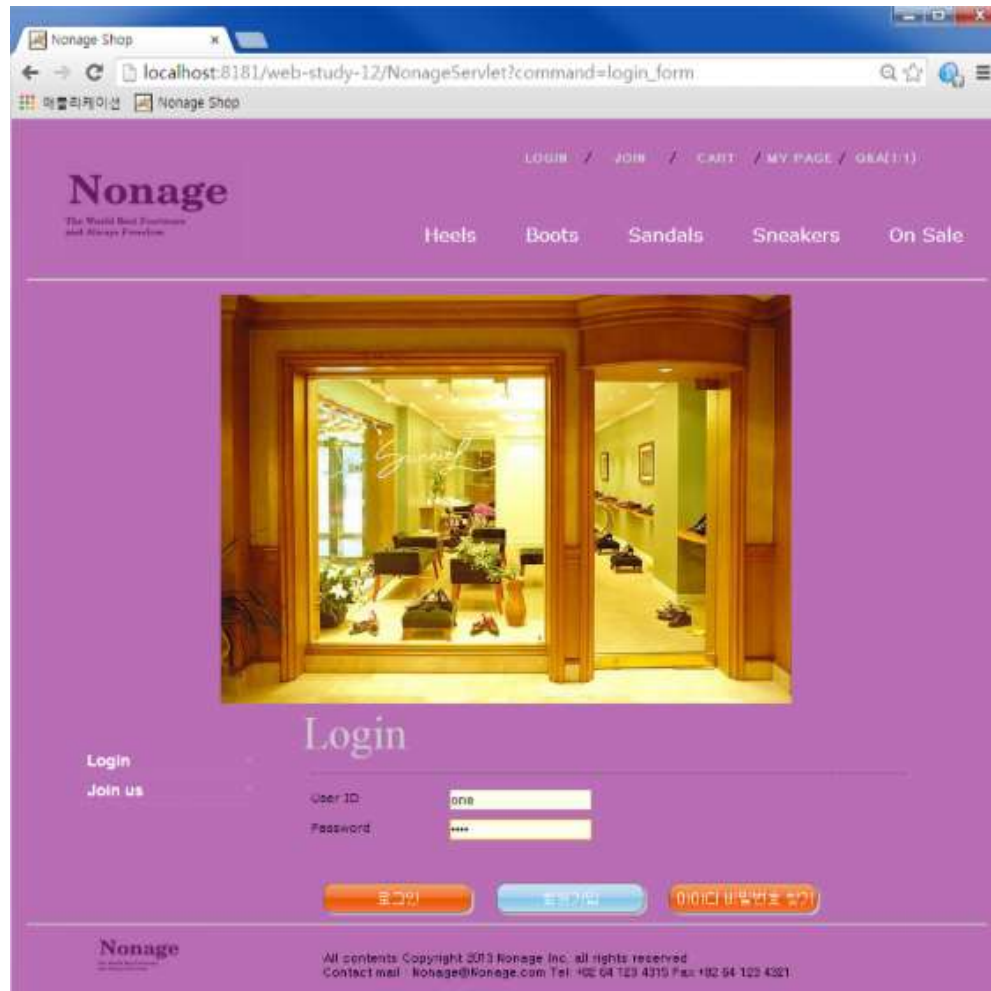
---

- (MemberController)
- 약관 동의 페이지 이동 메소드 작성
  - 메소드명: `contractView()`
  - 요청 문자열: `"/contract"`
  - 출력 화면: `"member/contract.jsp"`
- 회원가입 화면 출력 메소드
  - 메소드명: `joinView()`
  - 요청 문자열: `"/join_form"`
  - 출력 화면: `"join.jsp"`
- 아이디 중복 체크 화면 출력 메소드
  - `idCheckView()`
  - 요청 문자열: `"/id_check_form"`
  - 출력 화면: `"member/idcheck.jsp"`

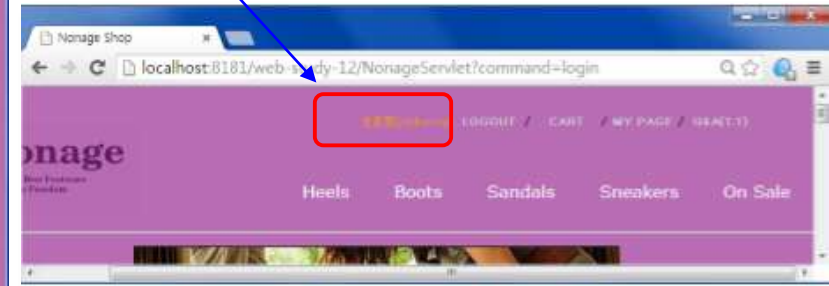


# 화면 구현

## 로그인



로그인 완료 시 사용자명 표시



# 화면 구현

## □ 회원가입 화면

The screenshot shows a web browser window with the title 'Nonage Shop'. The address bar displays 'localhost:8181/web-study-12/NonageServlet?command=join\_form'. The page has a purple background and a navigation menu on the left with 'Login' and 'Join us' (selected). The main content area is titled 'Join Us' and contains a 'Basic Info' section with fields for User ID, Password, Retype Password, Name, and E-Mail. Below this is an 'Optional' section with fields for Zip Code, Address, and Phone Number. At the bottom are two buttons: '회원가입' (Join) and '취소' (Cancel).

Field	Value	Action
User ID	pinkisung	중복 확인
Password	****	
Retype Password	****	
Name	성민정	
E-Mail	pinkisubin@nate.com	
Zip Code	138-790	주소 찾기
Address	서울 송파구 잠실3동 트러치움 아파트	100동 1001호
Phone Number	010-2321-2312	

회원가입    취소

# 주소 찾기

# 주소 정보 VO 클래스

---

## □ AddressVO

```
public class AddressVO {  
    private String zip_num;  
    private String sido;  
    private String gugun;  
    private String dong;  
    private String zipCode;  
    private String bunji;  
  
    // Getters and Setters  
}
```

# 주소 DAO 클래스

---

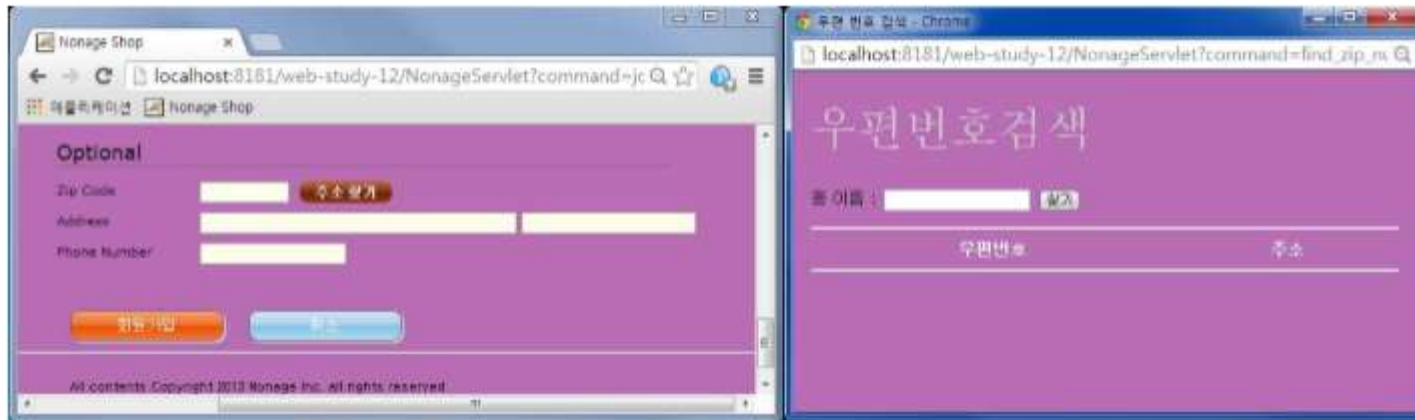
## □ AddressDAO

```
public List<AddressVO> selectAddressByDong(String dong) {  
    sql = "select * from address where dong like '%" + dong + "%'";  
}
```

# 화면 구현

## □ 우편번호 검색

- 회원 가입 시, “주소 찾기” 버튼을 누르면 우편번호를 검색하는 창이 열린다.



---

장바구니 담기

# 장바구니 정보 VO 클래스

---

## □ CartVO

```
public class CartVO {  
    private int cseq;  
    private String id;  
    private int pseq;  
    private String mname;  
    private String pname;  
    private int quantity;  
    private int price2;  
    private Timestamp indate;  
  
    // Getters and Setters  
}
```



# 장바구니 DAO 클래스

---

## □ CartDAO

```
public class CartDAO {  
    // 장바구니에 담기  
    public void insertCart(CartVO cartVO) {  
        sql = "insert into cart(cseq, id, pseq, quantity)" +  
            " values(cart_seq.nextval, ?, ? , ?,)";  
    }  
    // 장바구니 목록  
    public List<CartVO> listCart(String userId) {  
  
    }  
    // 장바구니 취소  
    public void deleteCart(int cseq) {  
        sql = "delete cart where cseq = ?";  
    }  
}
```

# 장바구니에 상품 담기

---

- 상품 상세 보기 페이지에서 [장바구니에 담기] 버튼을 클릭하면 이를 처리하기 위한 요청이 발생
  - 커맨드: `cart_insert`
- 장바구니에 담기 액션
  - 클래스명: `CartInsert`
  - `cardVO`에 사용자 아이디, 상품 일련번호(`pSeq`), 상품 수량(`quantity`)을 저장한다.
  - `cartService`의 `insertCart(cartVO)`를 호출한다.

# 장바구니 리스트 출력하기

---

- 장바구니에 상품을 담고 나면 현재 바구니에 담겨진 목록을 출력한다.
  - 커맨드: `cart_list`
- 장바구니 리스트 액션:
  - 사용자가 로그인 되어 있지 않으면 로그인 화면을 띄워 로그인을 하게 한다.
  - 사용자 아이디를 지정하여 DAO의 `listCart`를 호출한다.

# 장바구니에서 상품 삭제하기

---

- 장바구니에 담긴 상품 중 마음이 바뀌어 주문하지 않는 상품을 장바구니에서 삭제한다.
  - 삭제할 품목에 대해 체크 박스를 체크한 후 [삭제하기] 를 클릭한다.
  - 커맨드: `cart_delete`
- 장바구니 삭제 액션
  - 이전 목록 화면에서 체크박스를 체크한 'cseq' 목록 읽어 배열에 저장한다.
  - 체크한 목록에 대해 `cartDAO`에서 `deleteCart()`를 호출하여 상품을 삭제한다.

# 장바구니 확인 화면

## 장바구니 확인 화면

- 로그인 후 상품 상세 화면에서 장바구니에 상품을 담게 되면 다음의 화면으로 이동한다.
- 장바구니에 담긴 상품의 수량과 가격을 확인한 후에 쇼핑을 계속하거나 장바구니에 담긴 상품을 주문 할 수 있다.
- “주문하기” 버튼을 클릭하면 주문 처리 화면으로 이동한다.



---

# 주문 처리하기

# 주문 정보 VO 클래스

---

## □ OrderVO

```
public class OrderVO {  
    private int odseq;  
    private int oseq;  
    private String id;  
    private Date indate;  
    private String mname;  
    private String zip_num;  
    private String address;  
    private String phone;  
    private int pseq;  
    private String pname;  
    private int quantity;  
    private int price2;  
    private String result;  
  
    // Getters and Setters  
}
```

# 주문 처리 DAO 클래스

## □ OrderDAO

```
public class OrderDAO {  
  
    //장바구니의 상품을 주문 테이블에 넣는다.  
    public int insertOrder(List<CartVO> cartList, String id) {  
        selectMaxOseq = "select max(oseq) from orders";  
  
        // 주문 테이블(orders)에 주문 id를 먼저 넣어준다.  
        insertOrder = "insert into orders(oseq, id) values(?, ?)";  
        // 장바구니에 들어 있는 각 주문에 대해 상세 주문 내역을order_detail에 저장  
        for (CartVO cartVO : cartList) {  
            insertOrderDetail(cartVO, maxOseq);  
        }  
    }  
  
    public void insertOrderDetail(CartVO cartVO, int maxOseq) {  
        insertOrderDetail = "insert into order_detail(odseq, oseq, " +  
            "pseq, quantity) values(order_detail_seq.nextval, ?, ? , ?)";  
        // 상세 주문 insert 후, 장바구니를 업데이트한다.  
        updateCartResult = "update cart set result=2 where cseq=?";  
    }  
}
```



# 주문 처리 DAO 클래스

---

## □ OrderDAO(계속)

// 사용자 별 주문내역을 조회한다.

```
public List<OrderVO> listOrderById(String id, String result, int oseq) {  
    sql = "select * from order_view where id=? " +  
          "and result like '%'||?||'%' and oseq=?";  
}
```

// 사용자 별 주문번호를 조회한다.

```
public ArrayList<Integer> selectSeqOrdering(String id) {  
    sql="select distinct oseq from order_view " +  
        "where id=? and result='1' order by oseq desc";  
}
```

# 주문 처리

---

- 장바구니에 담긴 상품의 주문처리
  - 장바구니 목록 화면에서 <주문하기> 버튼을 클릭한다.
  - 커맨드: `order_insert`
  
- 주문 처리 액션
  - 사용자 로그인이 되어 있지 않으면 로그인 화면으로 이동한다.
  - 카트 목록의 주문 내역을 `orderDAO`의 `insertOrder()`를 호출하여 주문을 테이블에 저장한다.
  - 주문 내역 저장 후, 주문 확인 목록(주문 번호를 파라미터로 전송)을 표시한다.  
(화면 내용은 다음 페이지에)
    - 커맨드: `order_list`

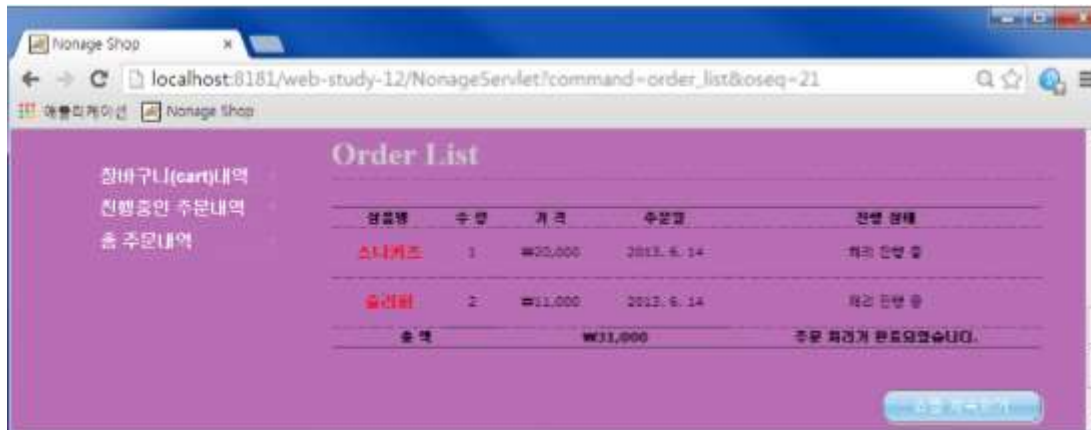
# 주문 내역 리스트 처리

---

- 주문 내역 리스트 액션(OrderListAction)
  - 커맨드: order\_list
  - 화면에서 입력한 주문번호를 받아 orderService의 getListOrderById()를 호출하여 주문 내역을 조회한다.
  - request 객체에 “orderList”에는 주문 내역, “totalPrice”에는 총 주문 금액을 계산하여 “mypage/orderList.jsp”를 호출한다.

# 주문 처리 화면

- 주문 확인 화면
  - 주문 처리가 완료되었음을 표시하는 화면



Order List				
상품명	수량	가격	주문일	주문 상태
스니키즈	1	₩20,000	2012. 6. 14	처리 진행 중
올리브	2	₩11,000	2012. 6. 14	처리 진행 중
총액		₩31,000		주문 처리가 완료되었습니다.

# 진행 중인 주문의 내역 처리

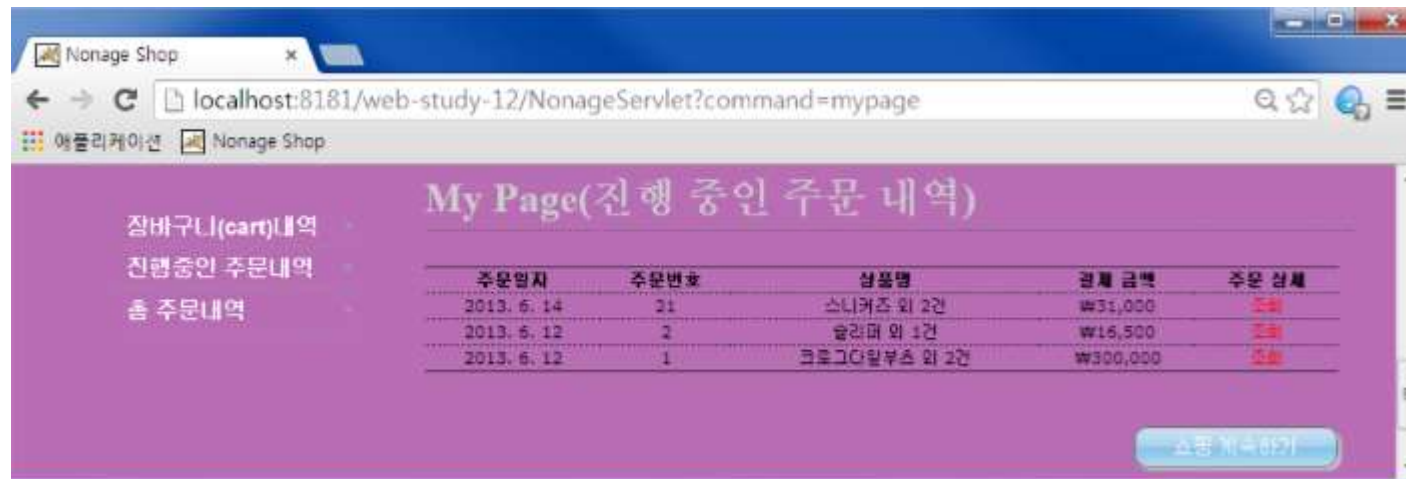
---

- 주문 진행 중, <MyPage>를 클릭하면, “진행 중인 주문내역” 화면으로 이동한다.
- 커맨드: mypage
- 진행 중인 주문 조회 액션
  - orderService에서 getSeqOrdering()을 호출하여 진행 중인 주문의 상세주문 번호 목록을 조회하여 배열에 저장한다.
  - 주문 번호 별로 주문을 조회하여 List 형 변수에 저장하고 총 주문금액(totalPrice)도 계산한다.
  - request 객체에 “title”=“진행 중인 주문 내역”, “orderList”=orderList를 저장하고 “mypage/mypage.jsp”를 호출한다.

# MyPage

## □ 현재 주문 진행 중인 목록

- Command: order\_list
- 화면



## □ 액션

- index.jsp 의 상단 메뉴에서 “마이 페이지” 링크를 클릭하면 MyPage화면이 나타난다.
- 로그인 이 안되어 있을 경우 로그인 화면을 띄워 사용자가 로그인하도록 한다.
- DAO에서 listOrderById를 호출하여 주문 내역을 표시한다.

# 주문 상세 정보 보기

- 주문 목록을 보다가 주문 상세 정보를 보려면 <조회> 를 클릭한다.
  - 커맨드: order\_detail



- 액션
  - 사용자 로그인 되어 있는지 확인한다. 로그인이 안되어 있으면 로그인 화면으로 이동한다.
  - 앞의 화면에서 파라미터로 입력된 주문번호(oseq)를 얻어와 주문 번호별 목록(listOrderById)을 조회하여 ArrayList 타입의 변수(orderList)에 저장한다.

# 주문 상세 정보 보기

---

## □ 액션(계속)

- 총 주문 금액을 계산한다. 반복문을 이용하여 각 주문 상품의 가격과 수량을 곱하여 합산
- request 저장 영역에 아래의 내용을 저장하여 전달한다.
  - “orderDetail” - 화면에서 조회한 주문 목록(orderList)의 첫 번째 데이터를 저장 (주문자 정보를 표시하기 위함)
  - “orderList:” - 조회한 주문 내역
  - “totalPrice” - 총 주문 금액
- “mypage/orderDetail.jsp” 화면을 호출한다.



# 총 주문 내역 보기

- 지금까지 주문한 모든 주문 정보를 보려면 <총 주문내역> 메뉴를 클릭한다.
  - 커맨드: `order_all`

- 액션

- 사용자의 로그인 여부를 확인한다. 로그인이 안되어 있으면 로그인 화면으로 이동한다.
- 사용자 아이디를 조건으로 주문 진행 중인 내역을 조회(`getSeqOrdering`) 하여 주문 번호 목록을 `ArrayList` 타입의 변수에 저장한다.
- 위의 각 주문번호에 대하여 아래와 같이 주문 상품목록을 얻어온다.
  - 사용자 아이디(`id`)와 주문 번호(`oseq`)를 조건으로 상세 주문 목록(`listOrderById`) 을 조회한다.
  - 위의 상세 주문 목록에서 첫 번째 주문 내역을 얻어와 `OrderVO` 객체에 저장한다.
  - 현재의 주문 번호에 여러 건의 주문 상품이 있을 경우 첫 번째 주문 상품인 “ooo 외 x건”으로 메시지를 조립하여 `OrderVO` 객체의 제품명 멤버변수(`pName`)에 저장한다.

# 총 주문 내역 보기

---

## □ 액션 (계속)

- 주문 내역의 총 금액을 계산하여 `price2`에 저장한다.
- 위에서 주문 내역(`OrderVO`) 객체를 `List` 형 변수에 저장한다.

## ■ request 파라미터 설정

- 제목: “title”=> “총 주문 내역”
- 주문 내역: “orderList” => 위에서 처리한 주문 내역

---

## Q&A 게시판

# Q&A 게시판

- 고객이 제품 또는 배송, 반품 등에 대한 전반적인 문의 사항을 기재하고 답변을 듣는 게시판



# Q&A VO 클래스

---

## □ QnaVO 클래스

```
public class QnaVO {  
    private int qseq;  
    private String subject;  
    private String content;  
    private String reply;  
    private String id;  
    private String rep;  
    private Date indate;  
  
    // Getters, Setters  
}
```

# Q&A DAO 클래스

---

## ▣ QnaDAO 클래스

```
public class QnaDAO {  
    // 전체 QnA 목록을 조회  
    public ArrayList<QnaVO> listQna(String id) {  
        sql = "select * from qna where id=? order by qseq desc";  
    }  
    // 일련번호 별 게시글 한 건 조회  
    public QnaVO getQna(int seq) {  
        sql = "select * from qna where qseq=?";  
    }  
    // 게시글 insert  
    public void insertQna(QnaVO vo, String session_id) {  
        sql = "insert into qna(qseq, subject, content, id) " +  
            "values(qna_seq.nextval, ?, ?, ?)";  
    }  
}
```

# QnA 게시물 목록 표시

---

- 메인화면에서 “Q&A(1:1)”링크를 클릭하면 게시물 목록 화면이 나타난다.
  - 커맨드: qna\_list
  
- 액션
  - 세션에서 사용자 정보를 읽어와 사용자가 로그인 상태인지 확인
  - 로그인이 안되어 있으면 로그인 화면으로 이동한다.
  - qnaService를 통하여 listQna()를 호출한다.
  - 조회한 목록을 request 객체의 “qnaList”에 담는다.
  - “qna/qnaList.jsp”화면을 호출한다.

# QnA 게시글 등록 화면 표시

---

- [1:1 질문하기] 버튼을 클릭하면 게시글 등록 화면으로 이동한다.
  - 커맨드: qna\_write\_form(게시글 등록 화면 표시, GET방식)
- 액션: 게시글 등록화면 표시
  - 로그인 되어 있지 않으면 로그인 화면으로 이동한다.
  - “qna/qnaWrite.jsp”화면을 호출한다.



# QnA 게시물 등록

---

- “1:1 고객 게시판”화면에서 <글쓰기> 버튼을 클릭하면 게시글이 등록된다.
  - 커맨드: qna\_write (게시글 등록, POST 방식)
  
- 액션
  - 화면에서 읽어온 파라미터(subject, content) 및 사용자 아이디의 정보를 QnaVO 객체에 저장한다.
  - qnaService 객체에서 insertQna(qnaVO, id)를 호출하여 게시글을 저장한다.
  - 등록이 성공하면 게시글 목록 화면으로 이동(command:qna\_list)한다.

# QnA 게시물 상세보기

---

- 게시물 목록 화면에서 게시물 제목을 클릭하면 게시물 상세보기 화면으로 이동한다.
  - 커맨드: `qna_view`
  
- 액션
  - 로그인에 되어 있지 않으면 로그인 화면으로 이동한다.
  - request 파라미터에서 “qseq”를 읽어온다.
  - `qnaService` 객체에서 `getQna()`를 실행하여 조회한 결과를 “qnaVO”를 키로 `model` 객체에 저장한다.
  - “qna/qnaView.jsp” 화면을 호출한다.

---

관리자 기능 구현

관리자 로그인 인증

# 관리자 기능

## □ 관리자 로그인 인증



The screenshot shows a web browser window with the title "Nonage Admin". The address bar displays the URL "localhost:8181/web-study-12/NonageServlet?command=admin\_login\_form". The browser's taskbar shows "마블리케이션" and "Nonage Shop". The main content area features a blue header bar with the text "Nonage" on the left and "WEB\_OFFICE PRO" on the right. Below the header, there is a login form with two input fields: "아이디" (ID) containing the text "admin" and "비밀번호" (Password) containing five asterisks. A yellow button labeled "업무 로그인" (Business Login) is positioned below the password field.

# 관리자 기능

- 상품관리
  - 상품을 등록하거나 검색하는 기능을 포함

The screenshot shows a web browser window titled 'Nonage Admin' with the URL 'localhost:8181/web-study-12/NonageServlet?command=admin\_product\_list'. The page features a purple header with the 'Nonage' logo and a 'WEB OFFICE PRG' label. A yellow 'logout' button is located in the top right. On the left, there is a sidebar with 'Admin Setting' and a list of links: '상품리스트', '주문리스트', '회원리스트', and 'QA리스트'. The main content area is titled '상품리스트' and includes a search bar with the placeholder '상품명' and three buttons: '검색', '관리보기', and '상품등록'. Below this is a table with the following data:

번호	상품명	원가	판매가	등록일	사양일부
11	스니커즈	15,000	20,000	2013. 6. 15	사용
10	샌달	5,000	5,500	2013. 6. 15	사용
9	스니커즈	15,000	20,000	2013. 6. 15	사용
8	샌달	5,000	5,500	2013. 6. 15	사용
7	스포츠샌달	5,000	5,500	2013. 6. 15	사용

At the bottom of the table, there is a pagination control showing '11 [2] [3]'.

# 관리자 기능

- 회원 관리
  - 등록된 회원을 조회하는 기능



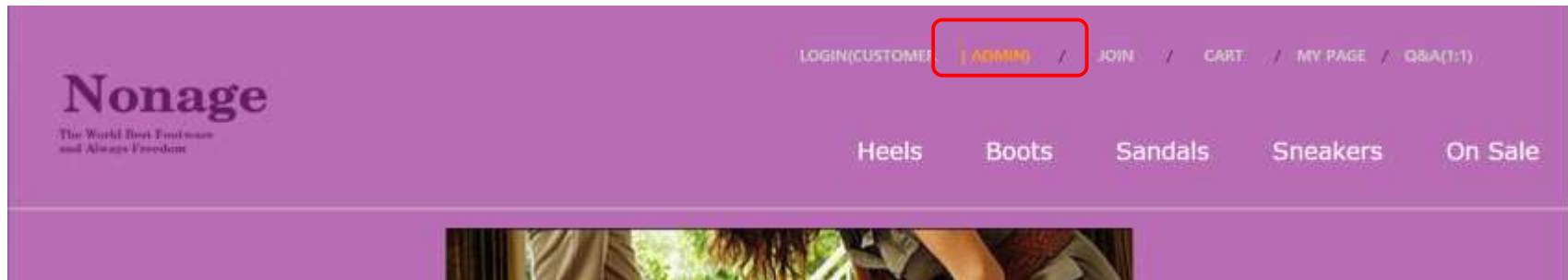
# 관리자 기능

- Q&A 게시판 관리
  - 고객이 올린 게시글을 확인하고 답변하는 기능



# 관리자 로그인 기능 구현

## □ 관리자 로그인 폼 이동



- “ADMIN”을 클릭하면 관리자 로그인 페이지로 이동
  - “header.jsp”를 수정하여 관리자 페이지로 이동하는 링크 추가

```
<a href="admin_login_form" style="width:100px;">| ADMIN)</a>
```

- “AdminController.java”에서 관리자 로그인 페이지를 표시하는 메소드 추가
  - 요청 문자열: /admin\_login\_form
  - 메소드명: adminLoginView()
  - 화면 폼 파일명: admin/main.jsp



# 관리자 로그인 기능 구현

## □ 관리자 인증 DAO 작성

### ■ 매핑 파일명: admin-mapping.xml

#### □ Mapper 명: AdminDAO

#### □ 관리자 인증 매핑: adminCheck: 아이디를 검색조건으로 비밀번호 조회

```
select pwd from admin where id=?
```

#### □ 관리자 정보 조회: getAdmin: 관리자 정보 조회

```
select * from admin where id=?
```

### ■ DAO 클래스 작성

#### □ 클래스명: AdminDAO

#### □ 메소드명: int adminCheck()

- 관리자 인증 매핑 adminCheck를 호출한 결과를 반환처리한다.

#### □ 메소드명: AdminVO getAdmin()

# 관리자 로그인 기능 구현

## □ Service 클래스 작성

### ■ 클래스명: AdminServiceImpl

#### □ 메소드명: adminCheck()

- AdminDAO의 adminCheck()를 호출한 결과에 대하여 다음을 처리한다.
- 관리자가 존재하지 않으면 -1을 반환하고,
- 비밀번호가 맞지 않으면 0을 반환
- 아이디가 존재하고 비밀번호가 일치하면 1을 반환

#### □ 메소드명: getAdmin()

- Id로 조회한 결과를 AdminVO 타입으로 반환한다.

## □ 관리자 인증 AdminController 구현

### ■ 요청 문자열: “/admin\_login”

### ■ 요청 방식: POST

### ■ 구현 내용

- adminService의 checkAdmin()를 호출하여 관리자 아이디 확인
- 관리자 아이디 조회결과가 1이면 “admin\_product\_list”url을 요청한다.
- 조회결과가 0또는 -1이면 message 속성에 “비밀번호를 확인하세요.” 또는 “아이디를 확인하세요 ” 를 저장하여 “admin/main” 화면을 호출한다.

# 관리자 로그아웃 기능 구현

---

- 관리자페이지의 각 화면에서 “logout“ 버튼을 누르면 “admin\_logout” url이 요청된다.
- AdminController에서 logout 구현
  - 요청 URL: admin\_logout
  - 관리자 세션을 지우고, admin/main.jsp를 호출한다.

---

관리자 기능 구현

## 상품관리 기능

# 상품조회 기능 구현

## □ 화면 준비

- 화면 상단: admin/header.jsp
- 화면 하단: admin/footer.jsp
- 메뉴 파일: admin/sub\_menu.jsp

## □ 상품 목록 조회 매핑, DAO 구현, Service 구현

- 총 상품 목록의 개수 조회: countProductList

```
select count(*) from product where name like '%'||?||'%' ;
```

- 상품 목록 조회: listProduct

```
SELECT pseq, regdate, name, price1, price2, useyn, bestyn  
FROM product WHERE name LIKE '%'||?||'%'  
ORDER BY pseq desc;
```

- 상품 추가: insertProduct

```
INSERT INTO product(  
    pseq, kind, name, price1, price2, price3, content, image)  
VALUES(product_seq.nextval, ?, ?, ?, ?, ?, ?, ?);
```

# 상품조회 기능 구현

## □ 상품 목록 조회 매핑, DAO 구현, Service 구현 (계속)

### ■ 상품 수정

```
UPDATE product SET kind=?, useyn=?, name=?, price1=?, price2=?,  
price3=?, content=?, image=?, bestyn=? where pseq=?;
```

## □ 자바 스크립트 구현: product.js

### ■ 폼에 입력된 정보가 올바른지 판단하는 스크립트

- 함수명: go\_save()
  - 확인 필드: kind(상품 분류), name(상품명), price1(원가), price2(판매가), content(상품 상세), image(상품 이미지)
  - 모든 필드가 확인되었으면 submit을 수행
    - encoding, price1.value, price2.value, price3.value에서逗를 제거하여 저장한다.
    - action: "admin\_product\_write"

### ■逗 제거 함수

- 함수명: removeComma(input)
- 내용: input.value.replace(/,/gi, "");

# 상품조회 기능 구현

---

## □ 자바 스크립트 구현: product.js

- 상품 리스트로 이동 함수
  - 함수명: go\_mov()
  - action: admin\_product\_list
- 상품 검색 함수
  - 함수명: go\_search()
  - action: admin\_product\_list

## □ AdminController 구현

- adminService의 제품 목록을 조회하는 listProduct()을 호출하여 상품 목록을 조회하고
- Model 객체에 “productList”로 저장한다.
- “admin/product/productList”화면을 호출한다.

# 상품등록 기능 구현

---

- 상품 등록 화면 페이지 이동
  - “productList.jsp” 페이지에서 ‘상품등록’ 을 클릭하면 상품 등록이 요청된다.
  - 요청 URL: admin\_product\_write\_form
  - 이미지 저장 디렉토리: product\_images
  
- AdminController에서 상품등록 화면 이동 메소드 구현
  - 메소드명: adminProductWriteView()
  - 요청 URL: admin\_product\_write\_form
    - kindList[] 배열에 아래의 내용을 저장한다.
    - {“Heels”, “Boots”, “Sandals”, “Slippers”, “Sneakers”, “Sale”}
    - Model 객체의 kindList 속성에 kindList[] 내용을 저장
    - “admin/product/productWrite.jsp” 저장



# 상품등록 기능 구현

---

## □ presentation-layer.xml에 파일 업로드 설정

```
<bean id="multipartResolver"  
class="org.springframework.web.multipart.commons.CommonsMultipartResolver">  
<property name="maxUploadSize" value="5000000"/>  
</bean>
```

## □ AdminController에서 상품 등록 메소드 구현

- 메소드명: adminProductWrite
- 이미지 업로드 디렉토리: product\_images
- adminService의 insertProduct()를 호출하여 상품 정보 저장

# 상품 상세정보 구현

---

- 상품 상세정보 페이지 이동
  - productList.jsp(상품 목록 페이지)에서 상품명을 클릭하면 상품 상세보기 페이지로 이동: go\_detail() 자바 스크립트 호출
    - action: “admin\_product\_detail?pseq="+pseq
- AdminController 에서 상품 상세보기 구현
  - 요청 문자열: admin\_product\_detail
  - 메소드명: adminProductDetail()
  - pseq를 조건으로 productService의 getProduct()를 호출하여 상품 정보를 조회
  - Model 객체에 “productVO” 속성으로 상품 정보 저장
  - Model 객체의 “kind” 속성에 신발의 종류를 카테고리명을 변환하여 저장한다.
    - {“o”, “Heels”, “Boots”, “Sandals”, “Slippers”, “Sneakers”, “Sale”}
  - 상세보기 화면을 호출한다.
    - “admin/product/productDetail.jsp”

# 상품정보 수정 구현

---

- 상품 정보의 수정 페이지 이동
  - 상품 상세보기에서 “수정“ 버튼을 클릭하면 자바스크립트의 `go_mod()`를 호출하여 수정화면으로 이동
    - action: “admin\_product\_update\_form?pseq="+pseq
  
- AdminController에서 상품정보 수정 구현
  - 요청 문자열: `admin_product_update_form`
  - 메소드명: `adminProductUpdateView()`
  - 상품 상세보기와 마찬가지로 흐름
  - 상세보기 화면을 호출
    - “admin/product/productUpdate.jsp” 페이지 호출

# 상품정보 수정 구현

---

## □ 상품 정보 수정 구현

- 상품 정보 수정 페이지에서 “수정“ 버튼을 클릭하면 자바스크립트의 `go_mod_save()`를 호출하여 수정정보 수정 요청
  - action: `admin_product_update`
  - 확인 필드: `kind`(상품 분류), `name`(상품명), `price1`(원가), `price2`(판매가), `content`(상품 상세), `image`(상품 이미지)
  - 모든 필드가 확인되었으면 `submit`을 수행
    - `encoding`, `price1.value`, `price2.value`, `price3.value`에서 콤마를 제거하여 저장한다.
    - action: “`admin_product_update`”

## □ AdminController 구현:

- 화면에서 입력된 값을 가지고 `adminService`의 `updateProduct()`을 호출하여 데이터를 갱신한다.
- 수정된 데이터의 확인을 위해 “`admin_product_list`”를 `redirect`로 요청한다.

---

관리자 기능 구현

## 주문 관리 기능

# 주문 관리 개요

- 회원이 상품에 대한 결제 완료 후, 배송 확인 후 주문처리를 완료해야 한다.
  - 관리자가 주문처리를 완료하면, 사용자 페이지 에서도 주문처리 완료된 것으로 표시된다.

Nonage WEB OFFICE PRG

logout

Admin Setting

주문리스트

주문자 이름  검색

주문번호(외래대부)	주문자	상품명	수량	우편번호	배송지	전화	주문일
4 (이 처리)	이백합	힐	1	130-120	서울시송파구잠실2동 리센스 아파트 201동 505호	011-123-4567	2013. 6. 15
7 (이 처리)	이백합	크로크다일부츠	2	130-120	서울시송파구잠실2동 리센스 아파트 201동 505호	011-123-4567	2013. 6. 15
3 (이 처리)	김나리	슬리퍼	3	133-110	서울시성동구성수동1가 1번지21호	017-777-7777	2013. 6. 15
1 (이 처리)	김나리	크로크다일부츠	1	133-110	서울시성동구성수동1가 1번지21호	017-777-7777	2013. 6. 15
2 (이 처리)	김나리	롤부츠	5	133-110	서울시성동구성수동1가 1번지21호	017-777-7777	2013. 6. 15
5 (이 처리완료)	이백합	롤부츠	1	130-120	서울시송파구잠실2동 리센스 아파트 201동 505호	011-123-4567	2013. 6. 15
6 (이 처리완료)	이백합	여성부츠	2	130-120	서울시송파구잠실2동 리센스 아파트 201동 505호	011-123-4567	2013. 6. 15

주문처리(입금확인)

181/web-study-12/NonageServlet?command=order\_detail&oseq=3

My Page(주문 상세 정보)

주문자 정보

주문일자	주문번호	주문지	주문 총액
2013. 6. 17	3	이백합	₩198,000

주문 상품 정보

상품명	상품별주문번호	수량	가격	처리 상태
힐	4	1	₩12,000	진행중
롤부츠	5	1	₩50,000	처리완료
여성부츠	6	2	₩36,000	처리완료
크로크다일부츠	7	2	₩100,000	진행중

# 주문 관리 매핑 과 DAO 추가

## □ 주문 조회 매핑, DAO 구현, Service 구현

### ■ 주문 전체 조회 매핑: listOrder(String member\_name)

#### □ “order-mapping.xml”에 추가

```
SELECT * FROM order_view WHERE mname LIKE '%'||?||'%'
ORDER BY result, oseq desc
```

### ■ ProductDAO에 주문 전체조회 메소드 추가

#### □ 메소드명: List<OrderVO> listOrder(String member\_name)

## □ 주문 상태 갱신 매핑, DAO, Service 구현

### ■ 주문 상태 갱신 매핑: updateOrderResult(int odseq)

```
UPDATE order_detail SET result='2' WHERE odseq=?
```

### ■ ProductDAO에 주문 상태 갱신 메소드 추가

#### □ 메소드명: void updateOrderResult(int odseq)

# 주문 목록 출력

---

- 화면 왼쪽의 서브메뉴에서 “주문 리스트” 링크를 클릭하면 “admin\_order\_list” 요청이 발생한다.
- 주문 목록 출력을 위한 AdminController 구현
  - 메소드명: adminOrderList
  - 요청 URL: admin\_order\_list
  - 처리 내용
    - 화면에서 사용자명이 입력되었으면 String 타입의 key변수에 사용자명 저장.
    - 입력되지 않았으면 key는 “”로 초기화
    - adminService의 listOrder를 호출하여 주문 목록을 조회해 온다.
    - Model 객체의 “orderList” 속성에 조회한 주문 목록을 설정한다.
    - “admin/order/orderList.jsp” 화면을 호출한다.



# 주문 완료 처리

---

- 주문 리스트에서 주문 처리할 항목에 대해 체크박스를 체크한 후 ‘주문처리(입금확인)’ 버튼을 클릭하여 주문 처리를 완료한다.
  
- 주문 처리완료를 위한 AdminController
  - 메소드명: AdminOrderSave
  - 요청 URL: admin\_order\_save
  - 처리 내용:
    - 화면에서 전송한 주문 처리 체크 항목을 oseq의 배열에 저장한다.
    - 배열에 저장된 주문 일련번호의 각 항목에 대해 반복문을 사용하여 업데이트 처리
    - adminService의 updateOrderResult()를 호출하여 처리

---

관리자 기능 구현

## 회원 관리

# 회원관리 매핑과 DAO 구현

---

- 회원 조회 매핑, DAO 구현, Service 구현
  - 회원 전체 조회 매핑: `listMember(String member_name)`
    - `member_mapping.xml`에 추가

```
SELECT * FROM member WHERE mname LIKE '%'||?||'%'  
ORDER BY regdate desc
```

# 회원 목록 출력

---

- 관리자 화면의 '회원 리스트' 를 클릭하면 회원 목록 출력 요청(admin\_member\_list)이 발생한다.
  
- 회원 목록 출력을 위한 AdminController 구현
  - 메소드명: adminMemberList()
  - 요청 URL: admin\_member\_list
  - 처리 내용:
    - 화면에서 사용자명이 입력되었으면 String 타입의 key변수에 사용자명 저장.
    - 입력되지 않았으면 key는 ""로 초기화
    - adminService의 listMember()를 호출하여 회원 목록을 조회해 온다.
    - 조회한 내역을 Model 객체의 "memberList"에 저장
    - 회원 목록 페이지인 "admin/member/memberList.jsp"를 호출한다.

---

관리자 기능 구현

## Q&A 게시판 관리

# 게시판 관리 매핑과 DAO 구현

- 사용자가 올린 질문에 대한 답변을 달기 위해 게시판을 관리한다.
- 게시판 관리 매핑, DAO 구현, Service 구현

- 게시판 전체 조회 매핑: `listAllQna`

- rep 컬럼의 값: 1-게시물, 2-답변
  - qna\_mapping.xml에 추가

```
SELECT * FROM qna ORDER BY indate desc
```

- 게시판 답변 처리 매핑: `updateQna(QnaVO)`

```
UPDATE qna SET reply=?, rep='2' WHERE qseq=?
```

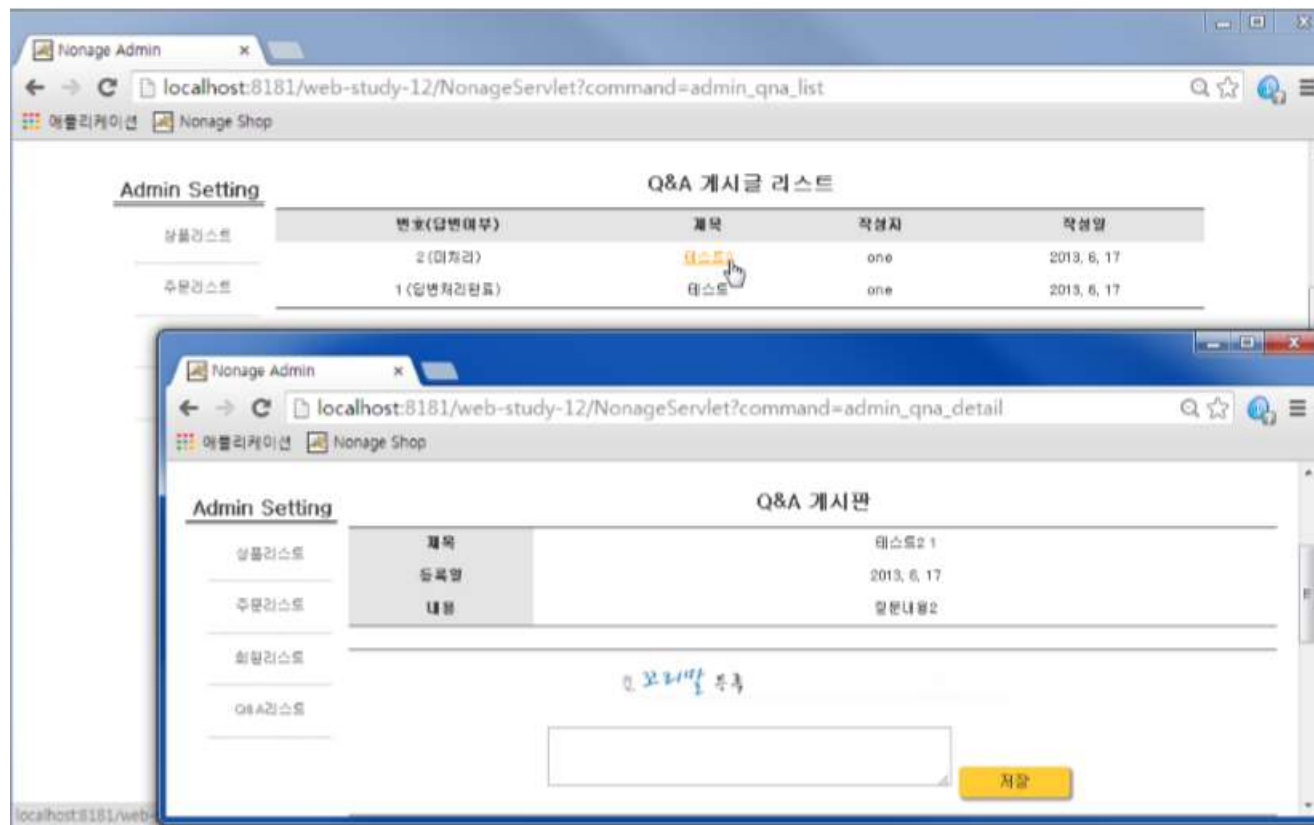
# 게시판 목록 출력

---

- 관리자 화면의 좌측 서브 메뉴에서 '게시판 리스트' 를 클릭하면 Q&A 게시판 목록 출력 요청(admin\_qna\_list)가 발생한다.
  
- 게시판 목록 출력을 위한 AdminController 구현
  - 메소드명: adminQnaList()
  - 요청 URL: admin\_qna\_list
  - 처리 내용:
    - adminService의 listAllQna()를 호출하여 Q&A게시판 목록을 조회하여 List<QnaVO> 변수에 저장한다.
    - 조회한 내용을 Model 객체의 “qnaList” 속성에 저장한다.
    - 게시판 출력 페이지를 호출한다: “admin/qna/qnaList.jsp”

# Q&A 답변 처리

- “게시글 리스트”에서 답변 처리가 되지 않은 게시글에 답변을 달기 위해서는 제목을 클릭한다.
  - 답변을 달 수 있는 게시글 상세보기 화면으로 이동한다.





# Q&A 답변 처리

---

- 게시물 상세 보기 AdminController 구현
  - 메소드명: adminQnaDetail
  - 요청 URL: admin\_qna\_detail
  - 처리 내용
    - 게시물 목록 화면에서 전송한 qseq 파라미터를 저장한다.
    - qnaService의 getQna()를 호출하여 게시물 정보를 조회한다.
    - Model 객체의 “qnaVO”속성에 게시물 정보를 저장한다.
    - 게시물 상세보기 페이지를 호출한다: admin/qna/qnaDetail.jsp

# Q&A 답변 처리

---

## □ 게시물 답변 처리

- 답변을 달고 ‘저장’ 버튼을 클릭하면 게시물 리스트로 이동한다.
- 답변을 단 게시물 번호 앞에 “답변 처리 완료” 메시지가 추가된다.
- 회원의 게시물 상세보기 페이지에서도 답변을 확인할 수 있다.

## □ 답변 처리 AdminController 구현

- 메소드명: adminQnaRepSave()
- 요청 URL: admin\_qna\_repsave
  - 화면에서 전송한 qseq(게시글 일련번호), reply(답글) 파라미터를 변수에 저장한다.
  - QnaVO 객체에 qseq와 reply를 설정하고 adminService의 updateQna() 메소드를 호출하여 답글을 갱신한다.
  - 상품 목록 페이지 출력: “redirect:admin\_product\_list”