












# Spring Portpolio

---

고영진

# 사용 툴&라이브러리

---

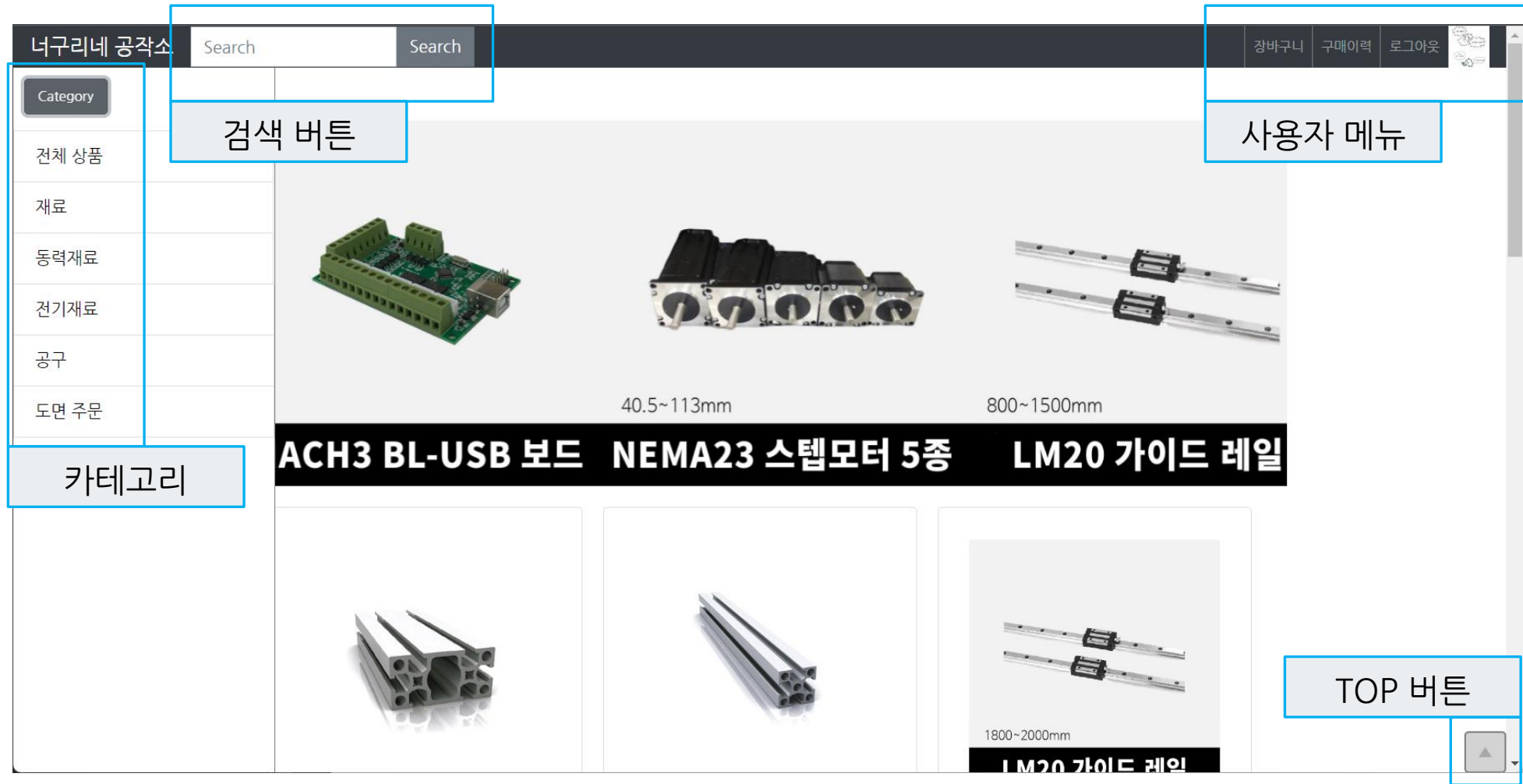
코드 편집기	 VS Code	 IntelliJ	
사용 언어	 Java	 JavaScript	 HTML5
프레임워크	 Spring Boot	 Thymeleaf (스크립트 엔진)	
DataBase	 MySQL	 Hibernate(JP A)	

# 참고 링크

---

GitHub	<a href="https://github.com/kyj9447/shopDwgReader">https://github.com/kyj9447/shopDwgReader</a>
접속 링크	<a href="https://www.kyj9447.kr:444">https://www.kyj9447.kr:444</a> (개인 서버로 연결됩니다.) ADMIN 계정 : admin@admin.com USER 계정 : user@user.com (Password는 둘 다 asdasdasd 입니다.)

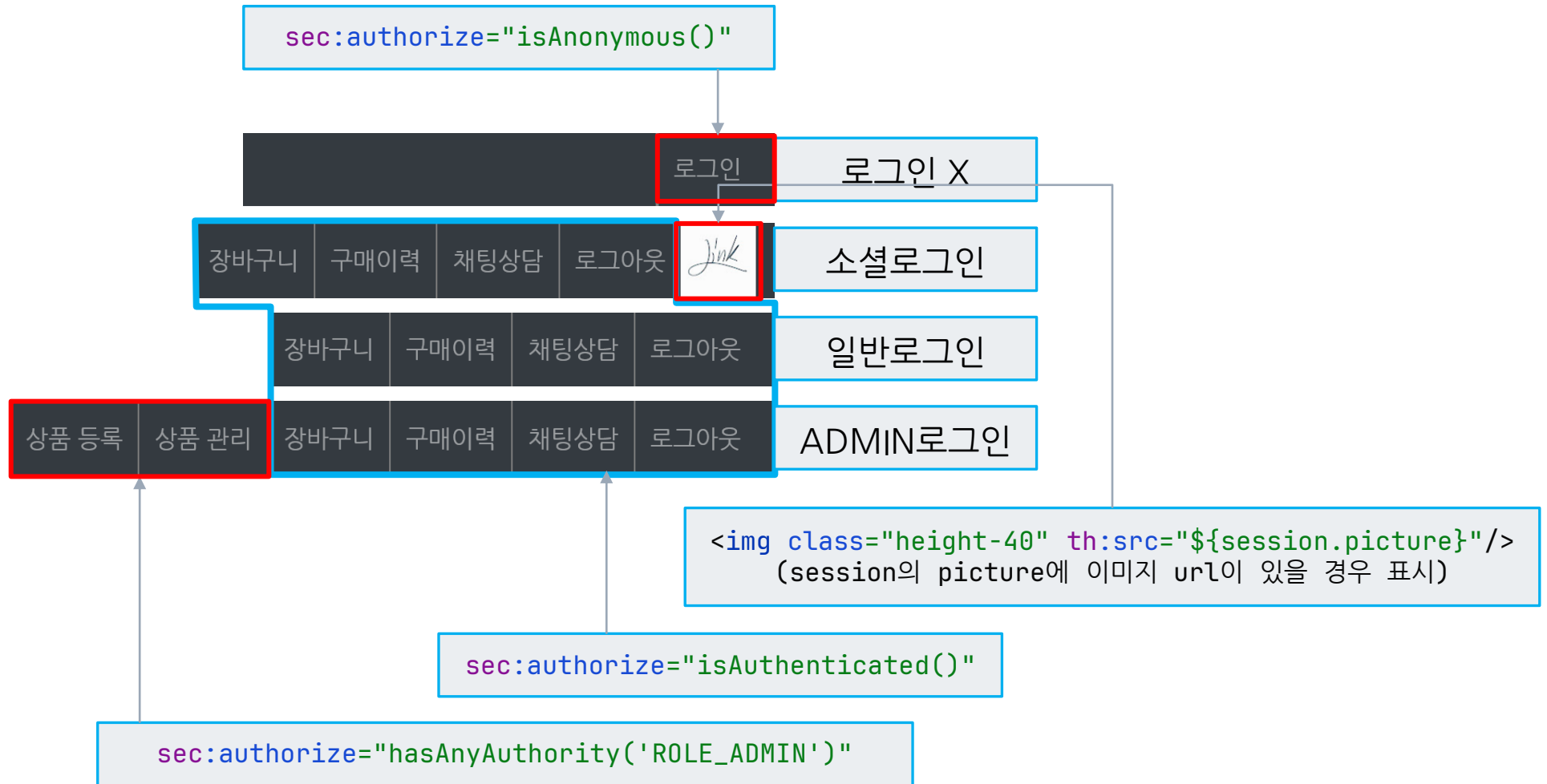
# 메인 화면 구성



# 메인 화면 구성



# 로그인 별 네비게이션 바



# 로그인 화면 구성

너구리네 공작소

Search

Search

로그인

Category

Log In

홈페이지 회원 로그인

Email

Email


Password


Password


로그인

회원가입

OAuth2 회원 가입 후 로그인  
(이미 가입된 회원은 이름과 사진을  
업데이트하고 로그인한다)

 Sign in with Google

 네이버 로그인

 카카오 로그인

2023 Shopping Mall

# 회원가입 화면 구성

```
@Getter
@Setter
public class MemberFormDto {
    @NotBlank(message = "이름은 필수 입력 값입니다.")
    private String name;
    @NotEmpty(message = "이메일은 필수 입력 값입니다.")
    @Email(message = "이메일 형식으로 입력해주세요.")
    private String email;
    @NotEmpty(message = "비밀번호는 필수 입력 값입니다.")
    @Length( min = 8, max = 16, message = "...생략...")
    private String password;
    @NotEmpty(message = "주소는 필수 입력 값입니다.")
    private String address;
    @NotEmpty(message = "전화번호는 필수 입력 값입니다.")
    // @NotEmpty(message = "[테스트용] ADMIN, USER 선택")
    private Role role;
}
```

너구리네 공작소

Search

Search

로그인

Category

이름

이름을 입력해주세요.

이름은 필수 입력 값입니다.

이메일 주소

이메일을 입력해주세요.

이메일은 필수 입력 값입니다.

비밀번호

비밀번호 입력

비밀번호는 필수 입력 값입니다.

비밀번호는 8자 이상, 16자 이하로 입력해주세요

주소

주소를 입력해주세요.

주소는 필수 입력 값입니다.

전화번호

전화번호를 입력해주세요.

전화번호는 필수 입력 값입니다.

[TEST] Role 선택

☒ADMIN ☐USER

Submit



# 회원 정보 구성

```
@Entity
@Table(name = "member", uniqueConstraints = @UniqueConstraint(columnNames = {"email", "loginType"}))
...생략...
```

email과 loginType이 일치해야 Unique한 멤버임을 설정

```
public class Member extends BaseEntity{
    @Id
    @Column
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    private String name;
    private String email;
    private String password;
    private String address;
    @Enumerated(EnumType.STRING)
    private Role role;
    private String number;
    @Column
    private String picture;
    @Column(nullable = false)
    private String loginType;
    ...builder, getter 등 생략...
```

	id	email	login_type	role
▶	2	kyj[REDACTED]@gmail.com	google	GOOGLE
	4	kyj[REDACTED]@gmail.com	kakao	KAKAO
	91	kyj[REDACTED]@gmail.com	normal	ADMIN
	3	kyj[REDACTED]@naver.com	naver	NAVER
	1	test@test.com	normal	ADMIN
	76	user@user.com	normal	USER
*		NULL	NULL	NULL

email이 동일해도 loginType이 다르면 별도의 멤버로 구분

# 회원 정보 구성

```
@Entity
@Table(name = "member", uniqueConstraints = @UniqueConstraint(columnNames = {"email", "loginType"}))
...생략...
```

```
public class Member extends BaseEntity{
    @Id
    @Column
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
```

```
    private String name;
    private String email;
    private String password;
    private String address;
```

```
    @Enumerated(EnumType.STRING)
```

```
    private Role role;
```

```
    private String number;
```

```
    @Column
```

```
    private String picture;
```

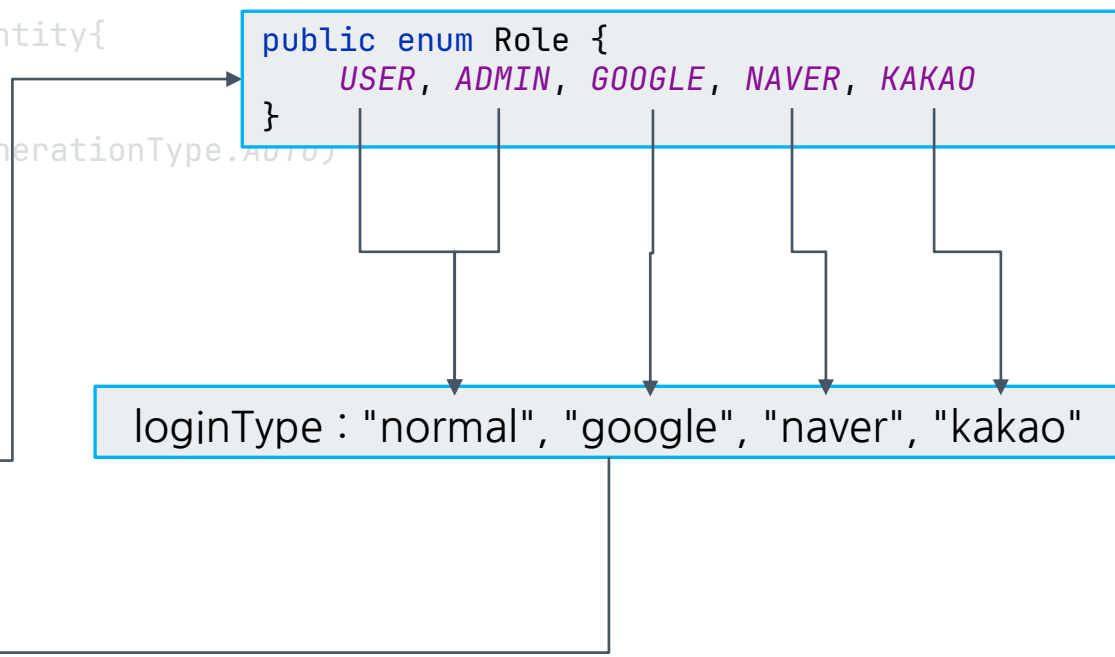
```
    @Column(nullable = false)
```

```
    private String loginType;
```

```
    ...builder, getter 등 생략...
```

```
public enum Role {
    USER, ADMIN, GOOGLE, NAVER, KAKAO
}
```

loginType : "normal", "google", "naver", "kakao"



# 회원 정보 구성

```
@Entity
@Table(name = "member", uniqueConstraints = @UniqueConstraint(columnNames = {"email", "loginType"}))
...생략...
```

```
public class Member extends BaseEntity{
    @Id
    @Column
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
```

```
    private String name;
    private String email;
    private String password;
    private String address;
```

```
    @Enumerated(EnumType.STRING)
```

```
    private Role role;
```

```
    private String number;
```

```
    @Column
```

```
    private String picture;
```

```
    @Column(nullable = false)
```

```
    private String loginType;
```

```
    ...builder, getter 등 생략...
```

```
public enum Role {
    USER, ADMIN, GOOGLE, NAVER, KAKAO
}
```

MemberService를 통해  
홈페이지 일반 가입시

loginType : "normal", "google", "naver", "kakao"

# 회원 정보 구성

```
@Entity
@Table(name = "member", uniqueConstraints = @UniqueConstraint(columnNames = {"email", "loginType"}))
...생략...
```

```
public class Member extends BaseEntity{
    @Id
    @Column
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
```

```
    private String name;
    private String email;
    private String password;
    private String address;
```

```
    @Enumerated(EnumType.STRING)
```

```
    private Role role;
```

```
    private String number;
```

```
    @Column
```

```
    private String picture;
```

```
    @Column(nullable = false)
```

```
    private String loginType;
```

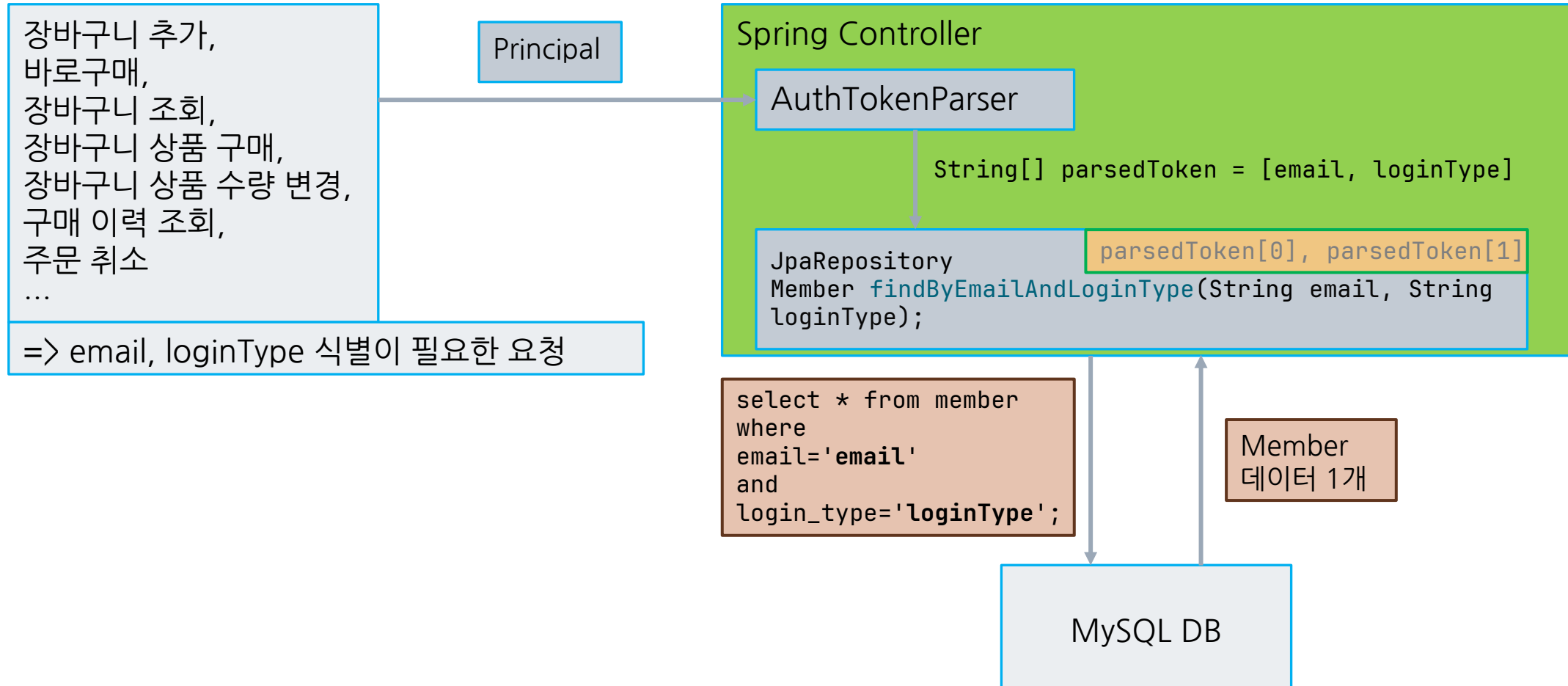
```
    ...builder, getter 등 생략...
```

```
public enum Role {
    USER, ADMIN, GOOGLE, NAVER, KAKAO
}
```

OAuth2를 통해 가입시

loginType : "normal", "google", "naver", "kakao"

# loginType 별 사용자 구분



# AuthTokenParser.java - 1

principal로 들어오는 토큰 정보를 확인하여 이메일, 로그인 타입을 길이 2의 배열로 return한다

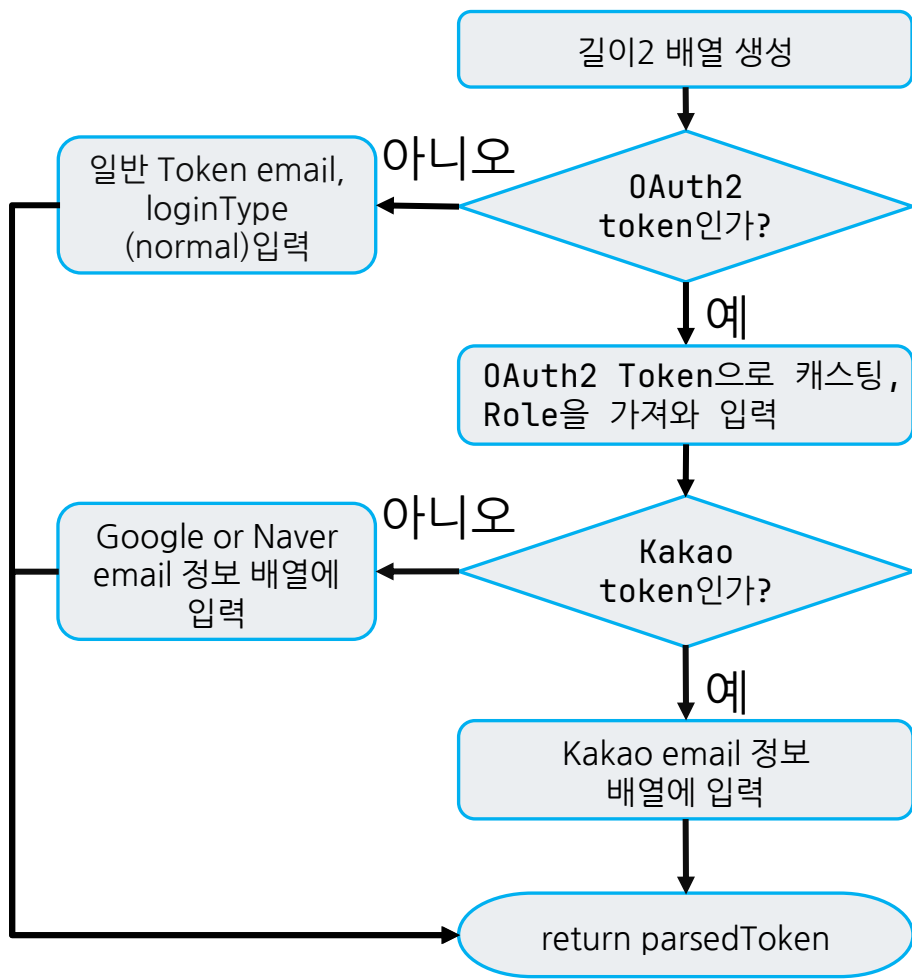
## 일반 로그인 토큰 예시

```
UsernamePasswordAuthenticationToken[
  Principal =
  org.springframework.security.core.userdetails.User
  [
    Username = XXX@gmail.com,
    Password = [PROTECTED],
    Enabled = true,
    AccountNonExpired = true,
    credentialsNonExpired = true,
    AccountNonLocked = true,
    Granted Authorities = [ROLE_ADMIN]
  ],
  Credentials = [PROTECTED],
  Authenticated = true,
  Details = WebAuthenticationDetails
  [
    RemoteIpAddress = 127.0.0.1,
    SessionId = 82F3371953E90195F3A0FB02DFBDBEBC
  ],
  Granted Authorities = [ROLE_ADMIN]
]
```

## OAuth2 로그인 토큰 예시

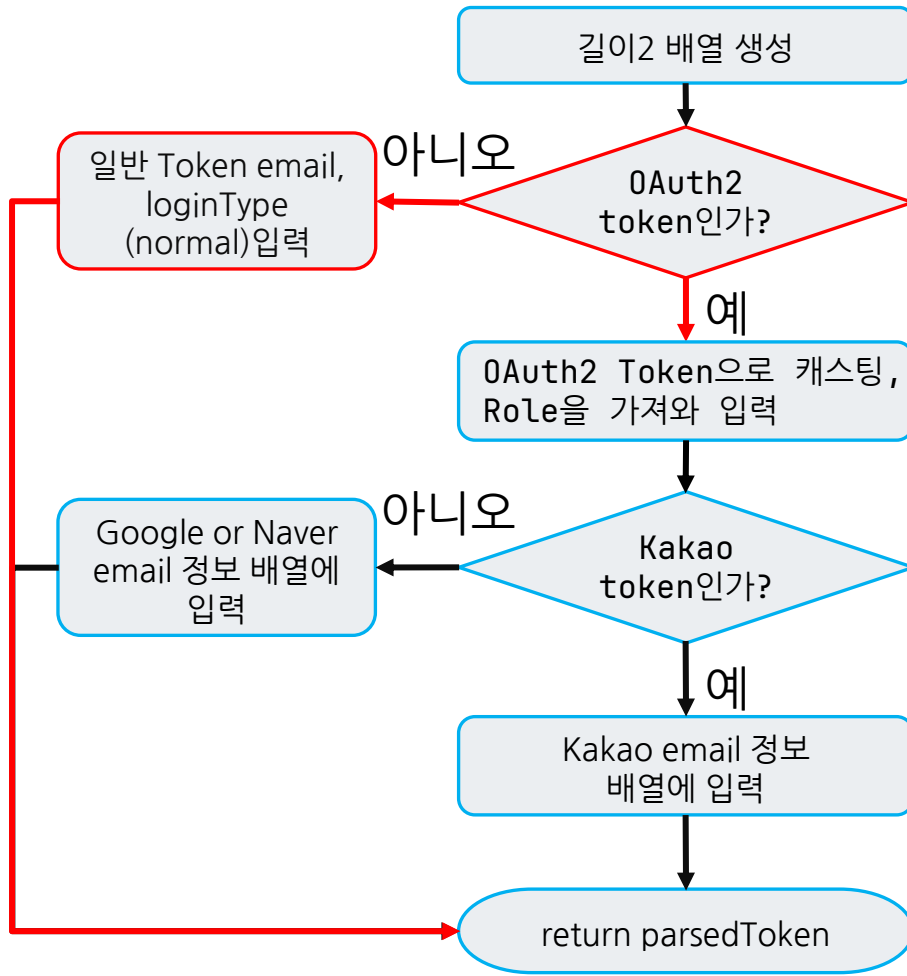
```
OAuth2AuthenticationToken[
  Principal =
  Name: ~~~~~~,
  Granted Authorities: [[GOOGLE]]
  User Attributes: [
    {
      sub= ~~~~~~,
      name=풀네임, given_name=이름, family_name=성,
      picture=https://lh3.googleusercontent.com/a/...(생략)...,
      email=XXX@gmail.com,
      email_verified = true, locale = ko
    }
  ],
  Credentials = [PROTECTED],
  Authenticated = true,
  Details = WebAuthenticationDetails[
    RemoteIpAddress = 127.0.0.1,
    SessionId = F497B406E38AA98BBFC87116E1666134
  ],
  Granted Authorities = [GOOGLE]
]
```

# AuthTokenParser.java - 2



```
public class AuthTokenParser {
    public static String[] getParseToken(Principal principal) {
        String[] parsedToken = new String[2];
        if(principal instanceof OAuth2AuthenticationToken){
            OAuth2AuthenticationToken authToken =
            (OAuth2AuthenticationToken) principal;
            parsedToken[1] =
            authToken.getAuthorities().iterator().next().getAuthority().toLowerCase();
            if (parsedToken[1].equals("kakao")) {
                parsedToken[0] =
                    ((Map<String, Object>)
            authToken.getPrincipal().getAttribute("kakao_account"))
                .get("email").toString();
            }
            else {
                parsedToken[0] = (String)
            authToken.getPrincipal().getAttributes().get("email");
            }
        }
        else {
            UsernamePasswordAuthenticationToken authToken =
                (UsernamePasswordAuthenticationToken) principal;
            parsedToken[1] = "normal";
            parsedToken[0] = (String) authToken.getName();
        }
        return parsedToken;
    }
}
```

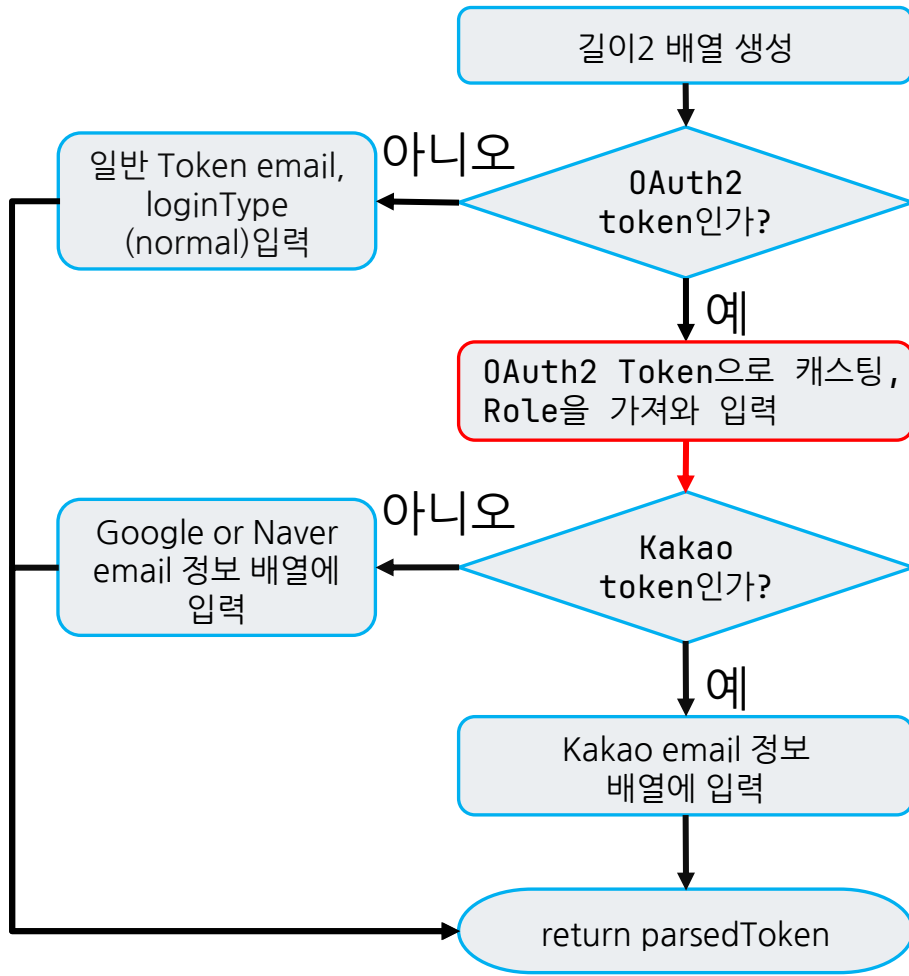
# AuthTokenParser.java - 3



```
public class AuthTokenParser {
    public static String[] getParseToken(Principal principal) {
        String[] parsedToken = new String[2];
        if(principal instanceof OAuth2AuthenticationToken){
            OAuth2AuthenticationToken authToken =
                (OAuth2AuthenticationToken) principal;
            parsedToken[1] =
                authToken.getAuthorities().iterator().next().getAuthority().toLowerCase();
            if (parsedToken[1].equals("kakao")) {
                parsedToken[0] =
                    ((Map<String, Object>)
                    authToken.getPrincipal().getAttribute("kakao_account"))
                    .get("email").toString();
            }
            else {
                parsedToken[0] = (String)
                authToken.getPrincipal().getAttributes().get("email");
            }
        }
        else {
            UsernamePasswordAuthenticationToken authToken =
                (UsernamePasswordAuthenticationToken) principal;
            parsedToken[1] = "normal";
            parsedToken[0] = (String) authToken.getName();
        }
        return parsedToken;
    }
}
```

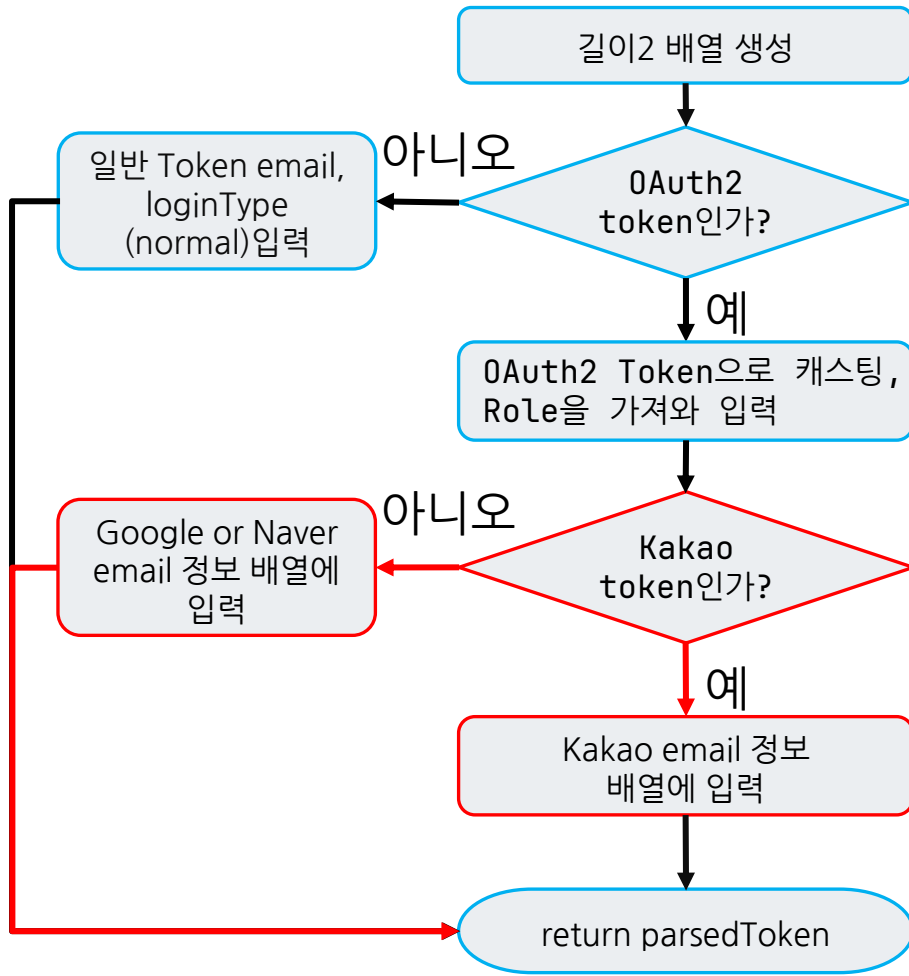


# AuthTokenParser.java - 4



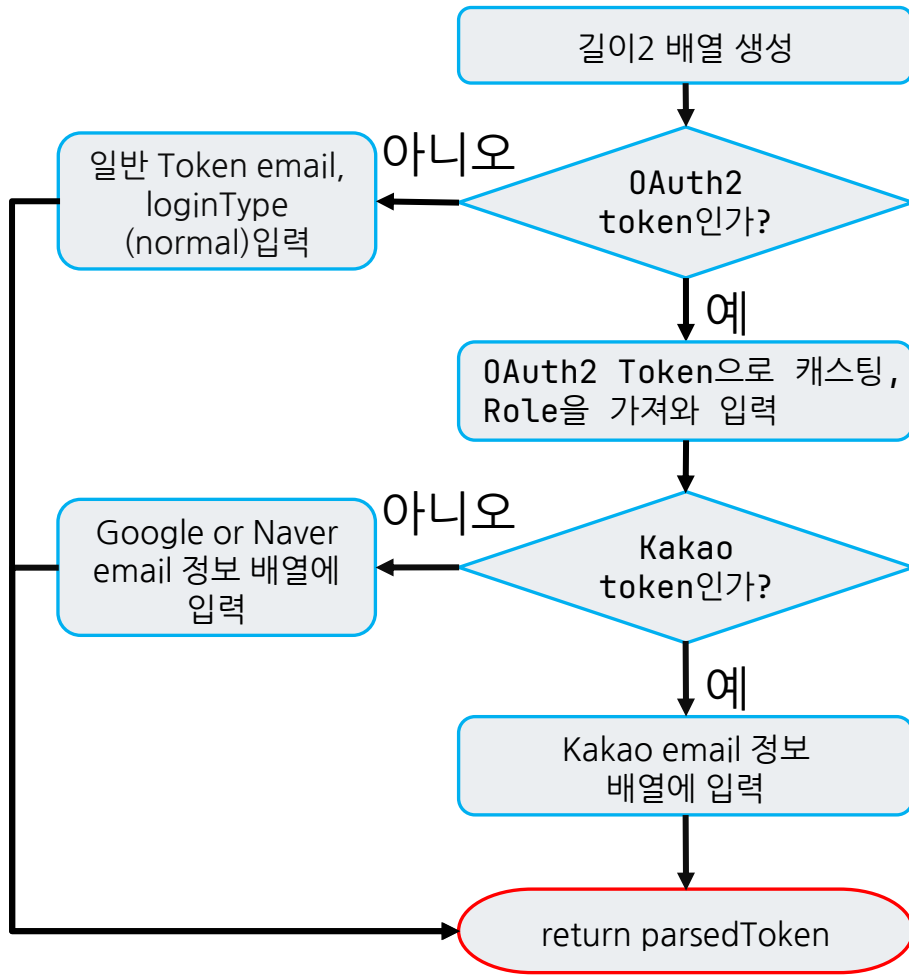
```
public class AuthTokenParser {  
    public static String[] getParseToken(Principal principal) {  
        String[] parsedToken = new String[2];  
        if(principal instanceof OAuth2AuthenticationToken){  
            OAuth2AuthenticationToken authToken =  
                (OAuth2AuthenticationToken) principal;  
            parsedToken[1] =  
                authToken.getAuthorities().iterator().next().getAuthority().toLowerCase();  
            if (parsedToken[1].equals("kakao")) {  
                parsedToken[0] =  
                    ((Map<String, Object>)  
                        authToken.getPrincipal().getAttribute("kakao_account"))  
                        .get("email").toString();  
            }  
            else {  
                parsedToken[0] = (String)  
                    authToken.getPrincipal().getAttributes().get("email");  
            }  
        }  
        else {  
            UsernamePasswordAuthenticationToken authToken =  
                (UsernamePasswordAuthenticationToken) principal;  
            parsedToken[1] = "normal";  
            parsedToken[0] = (String) authToken.getName();  
        }  
        return parsedToken;  
    }  
}
```

# AuthTokenParser.java - 4



```
public class AuthTokenParser {
    public static String[] getParseToken(Principal principal) {
        String[] parsedToken = new String[2];
        if(principal instanceof OAuth2AuthenticationToken){
            OAuth2AuthenticationToken authToken =
                (OAuth2AuthenticationToken) principal;
            parsedToken[1] =
                authToken.getAuthorities().iterator().next().getAuthority().toLowerCase();
            if (parsedToken[1].equals("kakao")) {
                parsedToken[0] =
                    ((Map<String, Object>)
                    authToken.getPrincipal().getAttribute("kakao_account"))
                    .get("email").toString();
            }
            else {
                parsedToken[0] = (String)
                authToken.getPrincipal().getAttributes().get("email");
            }
        }
        else {
            UsernamePasswordAuthenticationToken authToken =
                (UsernamePasswordAuthenticationToken) principal;
            parsedToken[1] = "normal";
            parsedToken[0] = (String) authToken.getName();
        }
        return parsedToken;
    }
}
```

# AuthTokenParser.java - 5



```
public class AuthTokenParser {  
    public static String[] getParseToken(Principal principal) {  
        String[] parsedToken = new String[2];  
        if(principal instanceof OAuth2AuthenticationToken){  
            OAuth2AuthenticationToken authToken =  
                (OAuth2AuthenticationToken) principal;  
            parsedToken[1] =  
                authToken.getAuthorities().iterator().next().getAuthority().toLowerCase();  
            if (parsedToken[1].equals("kakao")) {  
                parsedToken[0] =  
                    ((Map<String, Object>)  
                    authToken.getPrincipal().getAttribute("kakao_account"))  
                        .get("email").toString();  
            }  
            else {  
                parsedToken[0] = (String)  
                    authToken.getPrincipal().getAttributes().get("email");  
            }  
        }  
        else {  
            UsernamePasswordAuthenticationToken authToken =  
                (UsernamePasswordAuthenticationToken) principal;  
            parsedToken[1] = "normal";  
            parsedToken[0] = (String) authToken.getName();  
        }  
        return parsedToken;  
    }  
}
```

# 상품 등록 화면 구성

```
@Getter
@Setter
public class ItemFormDto {
    private Long id;
    @NotBlank(message = "상품명은 필수 입력 값입니다.")
    private String itemName;
    @NotBlank(message = "분류를 선택해주세요")
    private String itemCategory;
    @NotNull(message = "가격은 필수 입력 값입니다.")
    private Integer price;
    @NotBlank(message = "상품 설명은 필수 입력 값입니다.")
    private String itemDetail;
    private String itemRequest;
    @NotNull(message = "재고는 필수 입력 값입니다.")
    private Integer stockNumber;
    private ItemSellStatus itemSellStatus;
    private Integer thick;
    private List<ItemImgDto> itemImgDtoList = new ArrayList<>();
    private List<Long> itemImgIds = new ArrayList<>();
    private static ModelMapper modelMapper = new ModelMapper();

    public Item createItem() { return modelMapper.map(this, Item.class); }
    public CustomItem createCustomItem() { return modelMapper.map(this, CustomItem.class); }
    public static ItemFormDto of(Item item) { return modelMapper.map(item, ItemFormDto.class); }
}
```

The screenshot shows a web application interface for product registration. At the top is a navigation bar with links: '너구리네 공작소', 'Search', '상품 등록', '상품 관리', '장바구니', '구매이력', and '로그아웃'. Below the navigation bar is a 'Category' dropdown menu. The main heading is '상품 등록'. The form contains several input fields and buttons, with annotations in blue and red boxes:

- 판매중** (Status): A dropdown menu.
- 상품명** (Product Name): A text input field with the placeholder '상품명을 입력해주세요.'
- 분류** (Category): A dropdown menu with the placeholder '--분류를 선택하세요--'.
- 가격** (Price): A text input field with the placeholder '상품의 가격을 입력해주세요.'
- 재고** (Stock): A text input field with the placeholder '상품의 재고를 입력해주세요.'
- 상품 상세** (Product Description): A text area with the placeholder '상품 상세' and a note 'html 입력시 반영됨' (Reflected when HTML is entered).
- 요청 사항** (Request): A text area.
- 상품이미지1** (Product Image 1): A text input field with a 'Browse' button.
- 상품이미지2** (Product Image 2): A text input field with a 'Browse' button.
- 상품이미지3** (Product Image 3): A text input field with a 'Browse' button.
- 상품이미지4** (Product Image 4): A text input field with a 'Browse' button.

Annotations:

- material, motor, electirc, tools 중 택 1** (Choose 1 from material, motor, electirc, tools): A blue box pointing to the '분류' dropdown.
- 상품 상세 html 입력시 반영됨** (Product description is reflected when HTML is entered): A blue box pointing to the '상품 상세' text area.
- List<MultipartFile> 상품이미지1 은 대표이미지로 사용됨** (List<MultipartFile> Product Image 1 is used as the representative image): A blue box pointing to the '상품이미지1' input field.

# 상품 상세 화면 구성

너구리네 공작소 Search Search 상품 등록 상품 관리 장바구니 구매이력 로그아웃

Category

tool

판매중

전자 버니어 캘리퍼스

9000원

수량 1

수량 변경시 총 가격 계산

결제 금액 9000원

장바구니 담기 주문하기

전자 버니어 캘리퍼스

상품 상세 설명

!건전지 별도!

html 입력시 반영됨

상품이미지 1

상품이미지 나머지

사용자(Principal) + OrderDto (상품id, 수량) ajax로 POST 보냄

```
@Setter
public class OrderDto {
    @NotNull
    private Long itemId;
    @Min(value = 1, message = "최소 주문 수량은 1개입니다.")
    @Max(value = 999, message = "최대 주문 수량은 999개 입니다.")
    private int count;
}
```

# 카테고리 화면 구성 1

너구리네 공작소

Search

Search

상품 등록

상품 관리


장바구니

구매이력

로그아웃

Category

카테고리 : tool




작업 고글 / 보안경 3종

보안경 모음

10000원

주문하기

상품 이름 클릭시  
해당 상품의 나머지 이미지 url을  
ajax로 요청 후 상세화면으로 변경



조각날 (6mm)


6mm 조각날

12000원

장바구니담기

주문하기

해당상품 장바구니 담기 or 구매  
(수량은 1로 자동 설정)



전자 버니어 캘리퍼스

전자 버니어 캘리퍼스

장바구니담기

주문하기

Prev 1 Next

페이지 버튼 (Pageable 사용)

# 카테고리 화면 구성 2

너구리네 공작소

Search

Search

상품 등록

상품 관리

장바구니

구매이력

로그아웃

Category

카테고리 : tool



PVC 작업용 고글

ajax 응답으로 받은  
해당 상품의 나머지 이미지 url  
수량만큼 bootstrap carousel 추가

판매중

X

닫기 버튼

보안경 모음

10000원

수량 1

총 가격

장바구니담기

주문하기

아이템 설명

!레이저 보안경 구매시 **보호범위** 확인해주

해당상품 장바구니 담기 or 구매  
(수량 변경 가능)



조각날 (6mm)

6mm 조각날

12000원

장바구니담기

주문하기

https://ec2-13-209-198-40.ap-northeast-2.compute.amazonaws.com/category/tool#carouselExampleIndicators0

# 도면 주문 화면 구성 1

너구리네 공작소 Search Search 상품 등록 상품 관리 장바구니 구매이력 로그아웃

Category

도면 주문



※.dwg 파일 분석기능에 대한 자세한 내용은 동봉된 Java Portfolio.pptx 파일을 확인해주세요!

.dwg 파일 선택 Browse

--재료를 선택하세요--

업로드

.dwg파일, 두께 선택 후 submit

5T [원판단가: 50000원/m2(=0.5원/mm2), 가공단가: 50원/mm, 드릴단가: 1300원/개]

--재료를 선택하세요--

2T [원판단가: 20000원/m2(=0.2원/mm2), 가공단가: 20원/mm, 드릴단가: 500원/개]

5T [원판단가: 50000원/m2(=0.5원/mm2), 가공단가: 50원/mm, 드릴단가: 1300원/개]

10T [원판단가: 100000원/m2(=1원/mm2), 가공단가: 100원/mm, 드릴단가: 2800원/개]

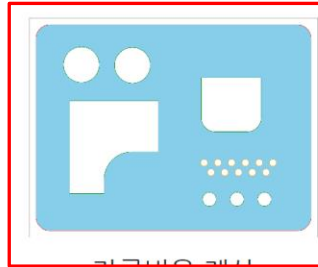


# 도면 주문 화면 구성 2

너구리네 공작소 Search 상품 등록 상품 관리 장바구니 구매이력 로그아웃  
Category

※.dwg 파일 분석기능에 대한 자세한 내용은 동봉된 Java Portfolio.pptx 파일을 확인해주세요!

업로드 결과



가공비용 계산

- 원판 가격 [회색] : (420mm x 320mm) 67200원
- 원판 가공 [빨간색] : (1366mm) 68284원
- 일반 가공 [초록색] : (511mm) 25534원
- 일반 가공 [초록색] : (314mm) 15693원
- 일반 가공 [초록색] : (163mm) 8151원
- 일반 가공 [초록색] : (163mm) 8151원
- 일반 가공 [초록색] : (63mm) 3142원
- 일반 가공 [초록색] : (63mm) 3142원
- 일반 가공 [초록색] : (63mm) 3142원
- 드릴 가공 ø10 [주황색] : 1300원
- 드릴 가공 ø10 [주황색] : 1300원
- 드릴 가공 ø10 [주황색] : 1300원
- 드릴 가공 ø10 [주황색] : 1300원
- 드릴 가공 ø10 [주황색] : 1300원
- 드릴 가공 ø10 [주황색] : 1300원
- 드릴 가공 ø10 [주황색] : 1300원
- 드릴 가공 ø10 [주황색] : 1300원
- 드릴 가공 ø10 [주황색] : 1300원
- 드릴 가공 ø10 [주황색] : 1300원

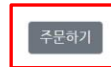
3.가격

총 합 216739원

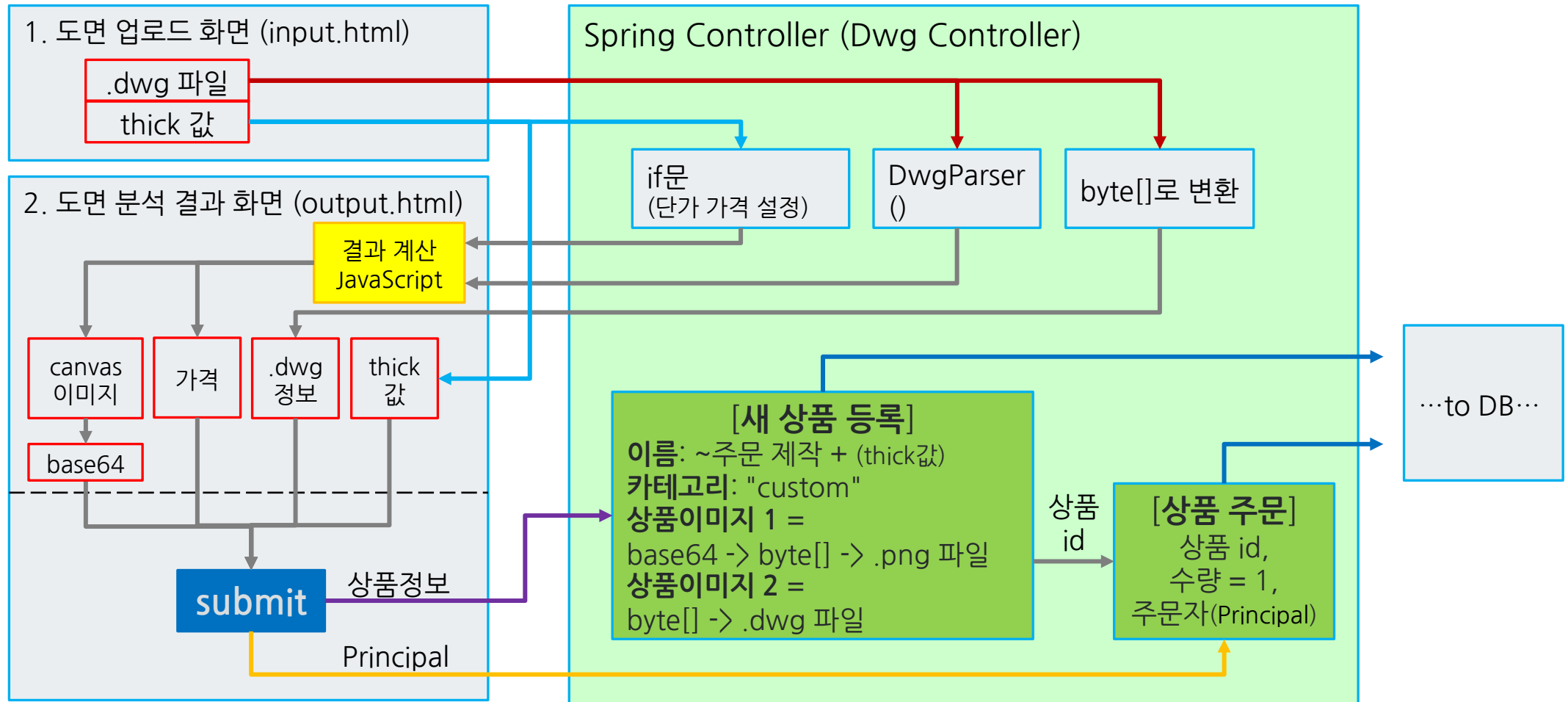
주문하기

1.html canvas 이미지 -> base64 문자열  
(canvas.toDataURL() 사용)

2.숨겨진 .dwg파일  
데이터 (byte[])



# 도면 주문 동작 구성



# 주문 상품 구성 - entity.CustomItem

```
@Entity
@Table(name = "customItem")
@Getter
@Setter
@ToString
public class CustomItem extends Item {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false, length = 50)
    private String itemName;

    @Column(nullable = false)
    private int price;

    @Column
    private int thick;

    @OneToMany
    @JoinTable(name = "member_item", joinColumns = @JoinColumn(name = "member_id"), inverseJoinColumns =
    @JoinColumn(name = "item_id"))
    CustomItem은 생성한 Member에게 귀속 -> 1:N 관계

}
```

Item을 상속받음

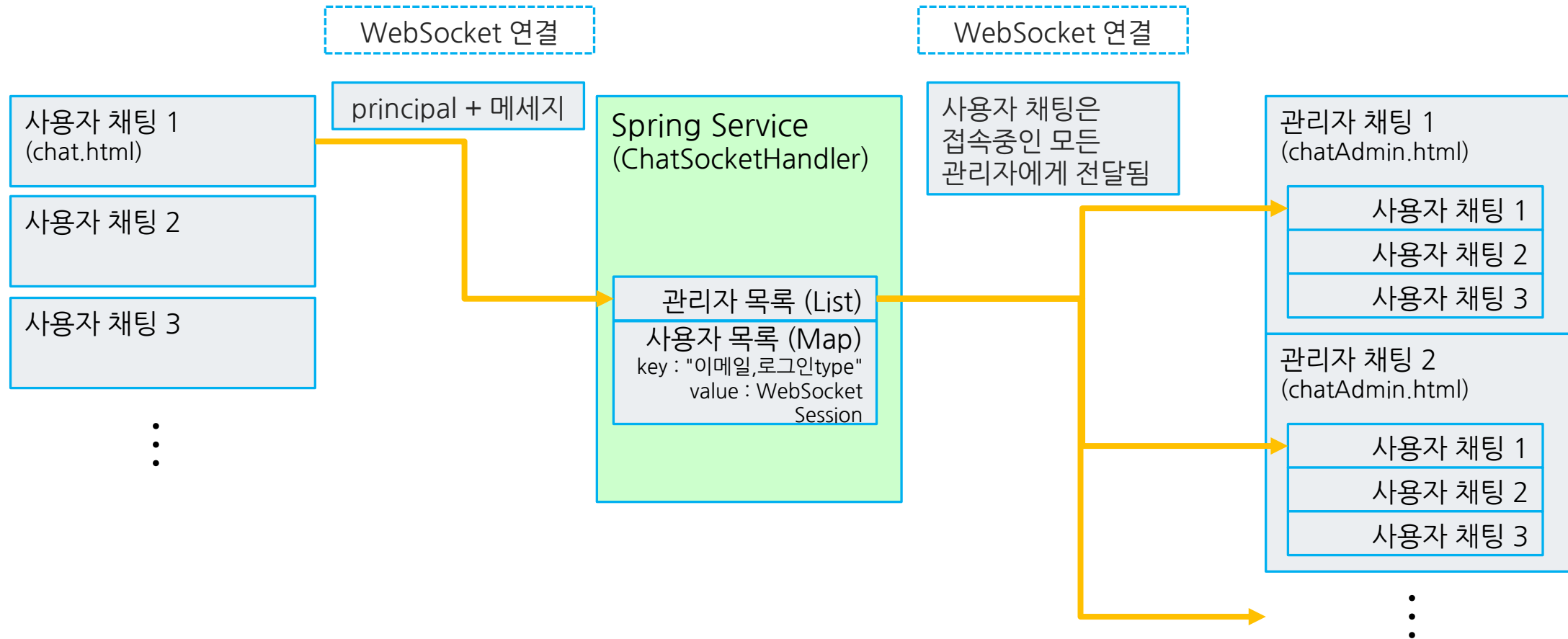
MySQL table(item)

dtype	item_id	category	item_name	price	stock_number	thick
CustomItem	95	custom	5T 철판 주문 제작	216739	0	5
Item	5	tool	전자 버니어 캘리퍼스	9000	200	NULL
Item	9	tool	6mm 조각날	12000	200	NULL
Item	12	tool	보안경 모음	10000	300	NULL
Item	18	electric	1.5kw VFD (LS산전)	600000	110	NULL

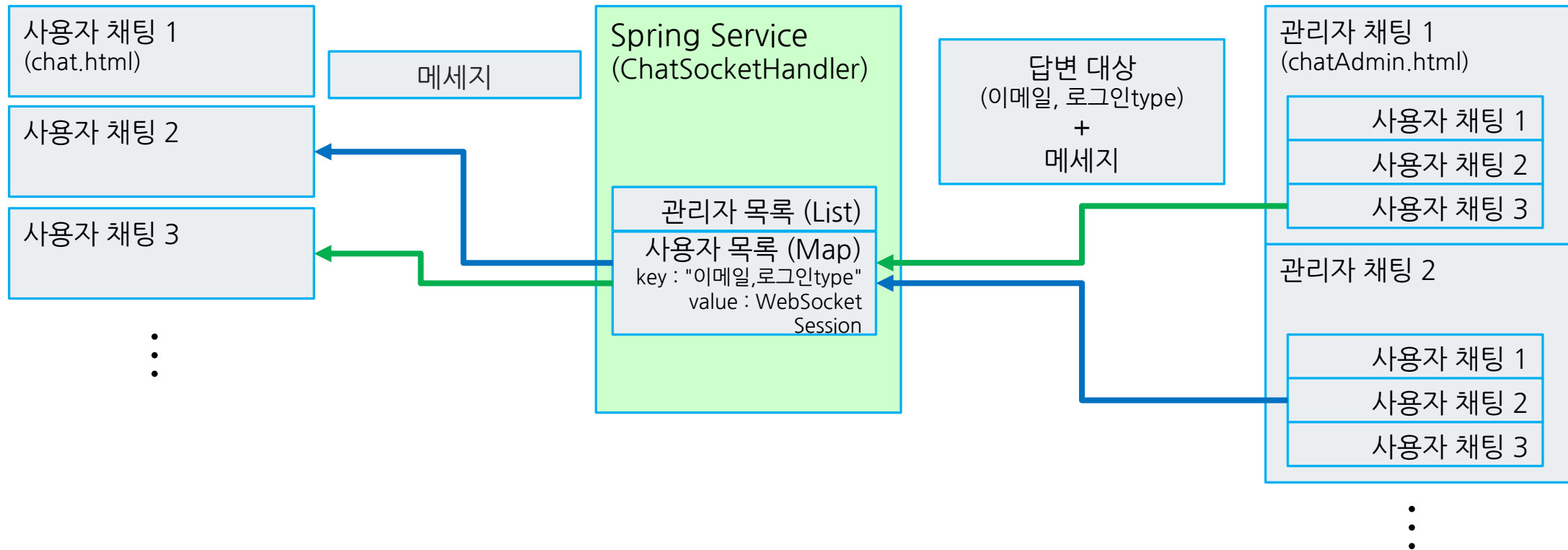
한 테이블에서  
dtype으로 구분함

CustomItem만  
가지고있는 값

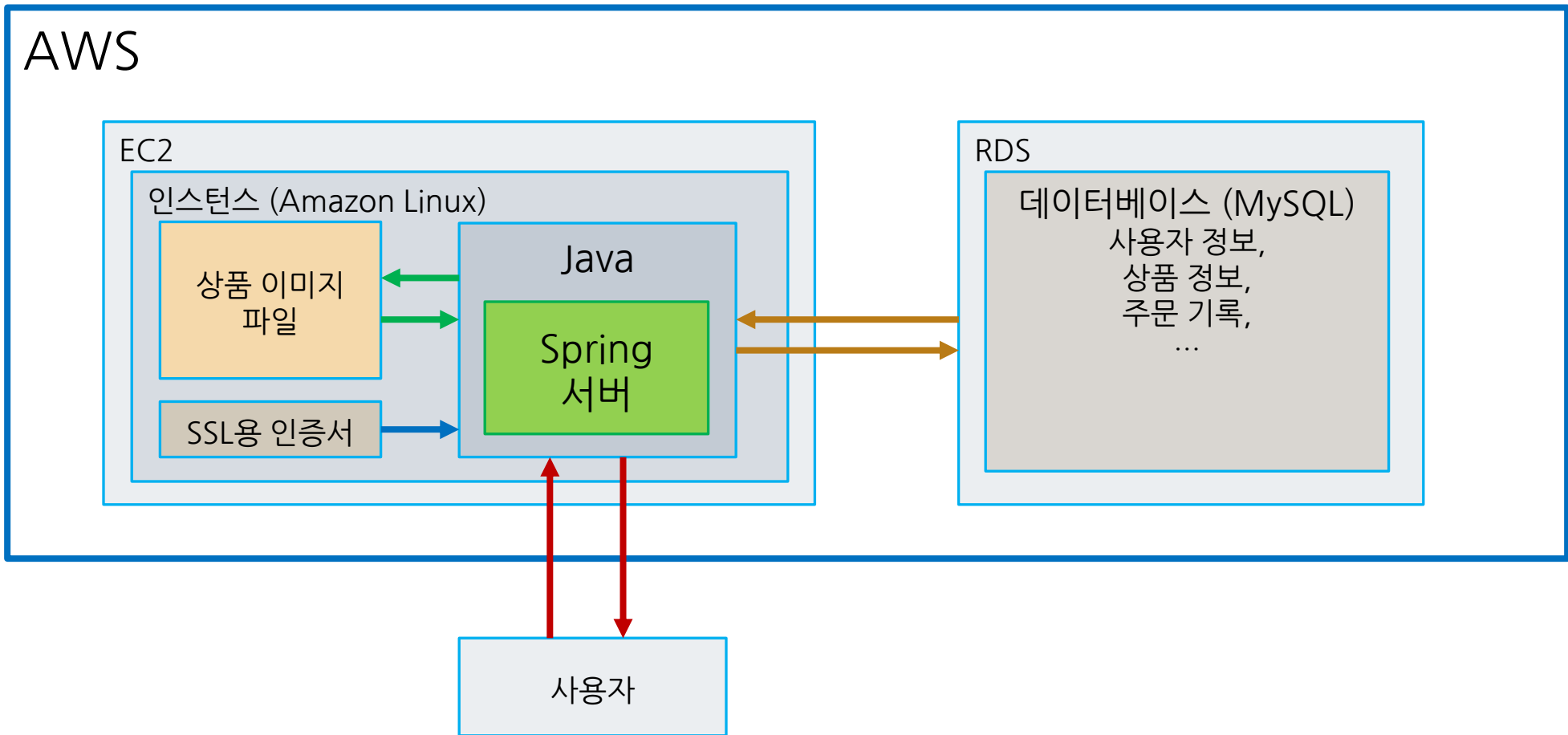
# 고객상담 채팅 1 (WebSocket 사용)



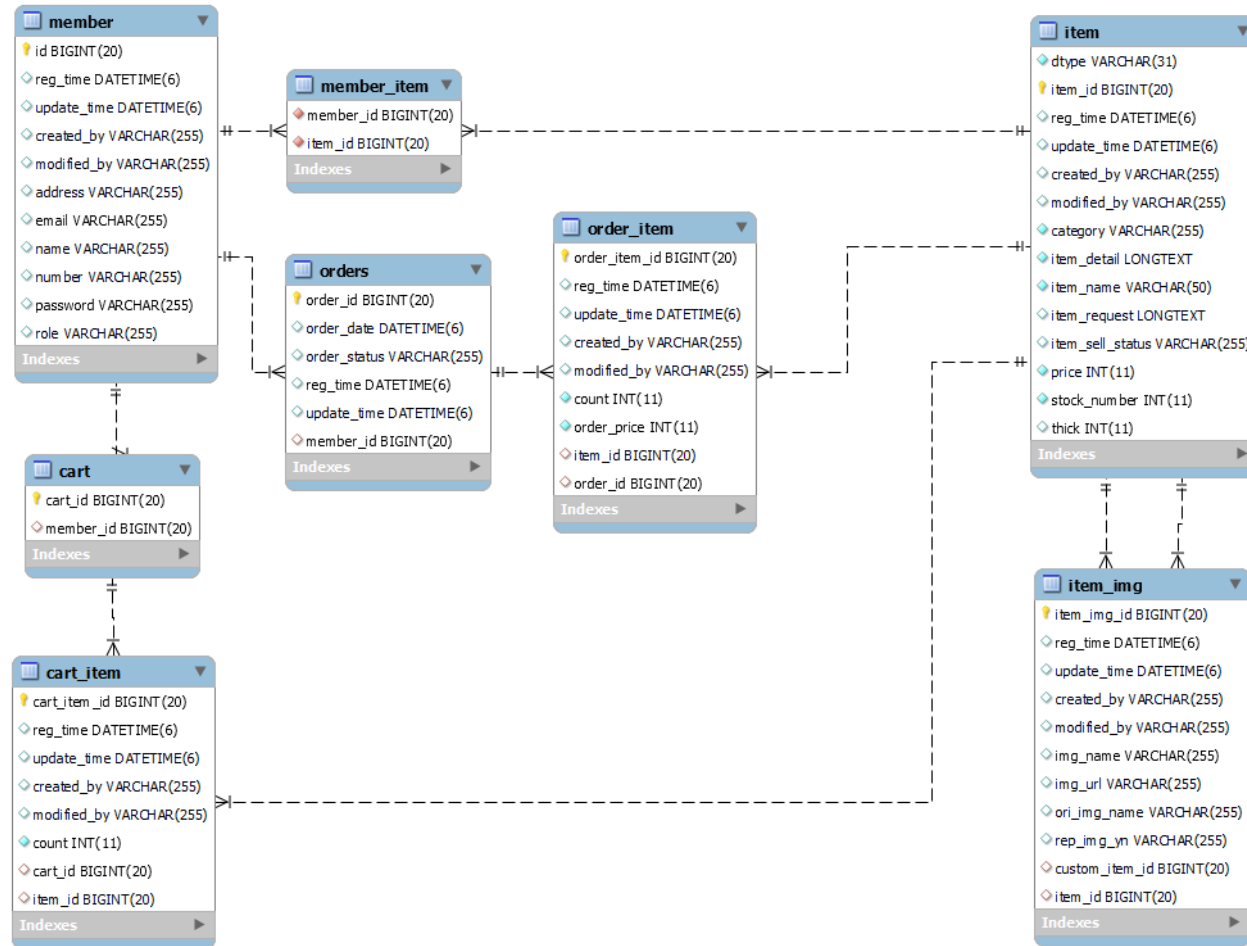
# 고객상담 채팅 2 (WebSocket 사용)



# AWS 게시 요약



# 데이터베이스 상세



# 테이블 상세 - cart, cart item

---

Table	테이블 내용	Column	데이터 타입	컬럼 타입	컬럼 키	null 허용	컬럼 내용
cart	장바구니 테이블	cart_id	bigint	bigint(20)	PRI	NO	장바구니 id
cart	장바구니 테이블	member_id	bigint	bigint(20)	MUL	YES	장바구니 회원 id

Table	테이블 내용	Column	데이터 타입	컬럼 타입	컬럼 키	null 허용	컬럼 내용
cart_item	장바구니 아이템	cart_item_id	bigint	bigint(20)	PRI	NO	상품 id
cart_item	장바구니 아이템	reg_time	datetime	datetime(6)		YES	등록일
cart_item	장바구니 아이템	update_time	datetime	datetime(6)		YES	수정일
cart_item	장바구니 아이템	created_by	varchar	varchar(255)		YES	상품 등록자
cart_item	장바구니 아이템	modified_by	varchar	varchar(255)		YES	상품 수정자
cart_item	장바구니 아이템	count	int	int(11)		NO	상품 개수
cart_item	장바구니 아이템	cart_id	bigint	bigint(20)	MUL	YES	장바구니 id
cart_item	장바구니 아이템	item_id	bigint	bigint(20)	MUL	YES	장바구니 상품 id



# 테이블 상세 - item

---

Table	테이블 내용	Column	데이터 타입	컬럼 타입	컬럼 키	null 허용	컬럼 내용
item	상품	dtype	varchar	varchar(31)		NO	상품 타입 (Item, CustomItem)
item	상품	item_id	bigint	bigint(20)	PRI	NO	상품 id
item	상품	reg_time	datetime	datetime(6)		YES	등록일
item	상품	update_time	datetime	datetime(6)		YES	수정일
item	상품	created_by	varchar	varchar(255)		YES	작성자
item	상품	modified_by	varchar	varchar(255)		YES	수정자
item	상품	category	varchar	varchar(255)		NO	카테고리
item	상품	item_detail	longtext	longtext		NO	상품 상세
item	상품	item_name	varchar	varchar(50)		NO	상품 이름
item	상품	item_request	longtext	longtext		YES	상품 요청사항
item	상품	item_sell_status	varchar	varchar(255)		YES	상품 판매상태
item	상품	price	int	int(11)		NO	상품 가격
item	상품	stock_number	int	int(11)		NO	상품 재고
item	상품	thick	int	int(11)		YES	두께(CustomItem일 경우)

# 테이블 상세 - item image

---

Table	테이블 내용	Column	데이터 타입	컬럼 타입	컬럼 키	null 허용	컬럼 내용
item_img	상품 이미지	item_img_id	bigint	bigint(20)	PRI	NO	이미지 id
item_img	상품 이미지	reg_time	datetime	datetime(6)		YES	등록일
item_img	상품 이미지	update_time	datetime	datetime(6)		YES	수정일
item_img	상품 이미지	created_by	varchar	varchar(255)		YES	작성자
item_img	상품 이미지	modified_by	varchar	varchar(255)		YES	수정자
item_img	상품 이미지	img_name	varchar	varchar(255)		YES	이미지 이름 (UUID 형식)
item_img	상품 이미지	img_url	varchar	varchar(255)		YES	이미지 URL
item_img	상품 이미지	ori_img_name	varchar	varchar(255)		YES	이미지 파일 원본 이름
item_img	상품 이미지	rep_img_yn	varchar	varchar(255)		YES	대표이미지 여부
item_img	상품 이미지	custom_item_id	bigint	bigint(20)	MUL	YES	커스텀 상품 id
item_img	상품 이미지	item_id	bigint	bigint(20)	MUL	YES	상품 id

# 테이블 상세 - member, member item

---

Table	테이블 내용	Column	데이터 타입	컬럼 타입	컬럼 키	null 허용	컬럼 내용
member	회원	id	bigint	bigint(20)	PRi	NO	회원 id
member	회원	reg_time	datetime	datetime(6)		YES	가입일
member	회원	update_time	datetime	datetime(6)		YES	수정일
member	회원	created_by	varchar	varchar(255)		YES	작성자
member	회원	modified_by	varchar	varchar(255)		YES	수정자
member	회원	address	varchar	varchar(255)		YES	주소
member	회원	email	varchar	varchar(255)	MUL	YES	이메일
member	회원	login_type	varchar	varchar(255)		NO	로그인 형식 (일반, 소셜)
member	회원	name	varchar	varchar(255)		YES	이름
member	회원	number	varchar	varchar(255)		YES	전화번호
member	회원	password	varchar	varchar(255)		YES	암호 (Password Encoded)
member	회원	picture	varchar	varchar(255)		YES	프로필 사진 URL (소셜 로그인)
member	회원	role	varchar	varchar(255)		YES	권한 (ADMIN, USER, GOOGLE...)

Table	테이블 내용	Column	데이터 타입	컬럼 타입	컬럼 키	null 허용	컬럼 내용
member_item	회원-상품 join용	member_id	bigint	bigint(20)	MUL	NO	회원 id
member_item	회원-상품 join용	item_id	bigint	bigint(20)	MUL	NO	상품 id

# 테이블 상세 - orders, order item

---

Table	테이블 내용	Column	데이터 타입	컬럼 타입	컬럼 키	null 허용	컬럼 내용
orders	주문	order_id	bigint	bigint(20)	PRI	NO	주문서 id
orders	주문	order_date	datetime	datetime(6)		YES	주문일
orders	주문	order_status	varchar	varchar(255)		YES	주문 상태
orders	주문	reg_time	datetime	datetime(6)		YES	등록일
orders	주문	update_time	datetime	datetime(6)		YES	수정일
orders	주문	member_id	bigint	bigint(20)	MUL	YES	회원 id

Table	테이블 내용	Column	데이터 타입	컬럼 타입	컬럼 키	null 허용	컬럼 내용
order_item	주문된 상품	order_item_id	bigint	bigint(20)	PRI	NO	주문상품 id
order_item	주문된 상품	reg_time	datetime	datetime(6)		YES	등록일
order_item	주문된 상품	update_time	datetime	datetime(6)		YES	수정일
order_item	주문된 상품	created_by	varchar	varchar(255)		YES	작성자
order_item	주문된 상품	modified_by	varchar	varchar(255)		YES	수정자
order_item	주문된 상품	count	int	int(11)		NO	수량
order_item	주문된 상품	order_price	int	int(11)		NO	가격
order_item	주문된 상품	item_id	bigint	bigint(20)	MUL	YES	아이템 id
order_item	주문된 상품	order_id	bigint	bigint(20)	MUL	YES	주문서 id

감사합니다