



[스파르타코딩클럽] Spring 심화반 - 2주차



매 주차 강의자료 시작에 PDF파일을 올려두었어요!

▼ PDF 파일

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/bf29e410-d563-44ba-9515-bcd73a3cd80f/_Spring__2_.pdf

[수업 목표]

1. 웹의 인증 및 인가의 개념을 이해한다.
2. 스프링 시큐리티를 이용해 폼 로그인 기능을 구현한다.
3. 스프링 시큐리티 OAuth2 를 이용해 소셜 로그인 기능을 구현한다.

[목차]

- 01. 2주차 오늘 배울 것
- 02. 웹의 인증 및 인가
- 03. 쿠키와 세션
- 04. 회원 관리 기능 요구사항 및 설계
- 05. '스프링 시큐리티' 프레임워크
- 06. 회원 가입 UI 반영
- 07. 회원 가입 기능 구현
- 08. 비밀번호 암호화 구현
- 09. 로그인, 로그아웃 기능 구현
- 10. 회원별 상품 등록 및 조회
- 11. 접근 불가 페이지 만들기
- 12. 소셜 로그인
- 13. 카카오 로그인 설정하기
- 14. 카카오 로그인 구현하기
- 15. 2주차 끝 & 과제 설명
- 16. 2주차 과제 답안 코드



모든 토글을 열고 닫는 단축키

Windows : **ctrl** + **alt** + **t**

Mac : **⌘** + **⌥** + **t**

01. 2주차 오늘 배울 것

▼ 1) 웹의 인증 및 인가의 개념



웹의 인증과 인가라는 개념에 대해서 구분해서 이해해볼게요!

▼ 2) '스프링 시큐리티'를 이용한 인증 및 인가 관리방법

👉 인증과 인가를 직접 구현해보기 위해 스프링 시큐리티 라는 개념에 대해 배워볼게요!

▼ 3) OAuth2 를 이용한 소셜 로그인

👉 여러분 웹사이트나 어플리케이션을 이용하실 때, 카카오나 네이버 아이디로 로그인 하는 거 많이 경험해보셨을 거예요! 스프링을 이용해 직접 구현해볼게요~!

02. 웹의 인증 및 인가

▼ 4) 인증 vs. 인가

💡 인증과 인가는 한글로 보나 영어 (Authentication vs. Authorization) 로 보나 비슷해 보일 뿐 아니라 실제 많이 혼동되어 사용되고 있지만 명확히 다른 의미를 가지고 있습니다.



출처: <https://aboutssl.org/authentication-vs-authorization/>

- **인증 (Authentication):** 사용자 신원을 확인하는 행위
- **인가 (Authorization):** 사용자 권한을 확인하는 행위

예를 들면,

- 인증: 회사 출입을 위한 출입증 확인 혹은 생체정보 (지문, 홍채) 인식
- 인가: 회사 건물 내 접근 권한 관리
 1. 방문자 → 회의실만 접근 가능
 2. 직원 → 회의실, 사무실 접근 가능
 3. 관리자 → 회의실, 사무실, 서버실, 물품보관실 접근 가능

웹에서의 인증 및 인가

- 인증: 로그인을 통해 본인임을 확인 (주로, 아이디와 패스워드 이용)
- 인가: 주로 역할에 따른 사용 권한 관리
 - 예) 웹 카페 사이트에서 회원 랭킹 별 가능한 첨부파일 크기를 다르게 부여

랭킹에 따른 혜택

· 랭킹이 높아질수록 게시를 작성시 한번에 첨부할수 있는 파일 용량이 증가합니다.

- 🍌 씨앗등급 이상 : **10MB** 첨부 가능 🌿 잎새등급 이상 : **15MB** 첨부 가능
- 🍎 열매등급 이상 : **20MB** 첨부 가능 🌳 나무등급 이상 : **25MB** 첨부 가능

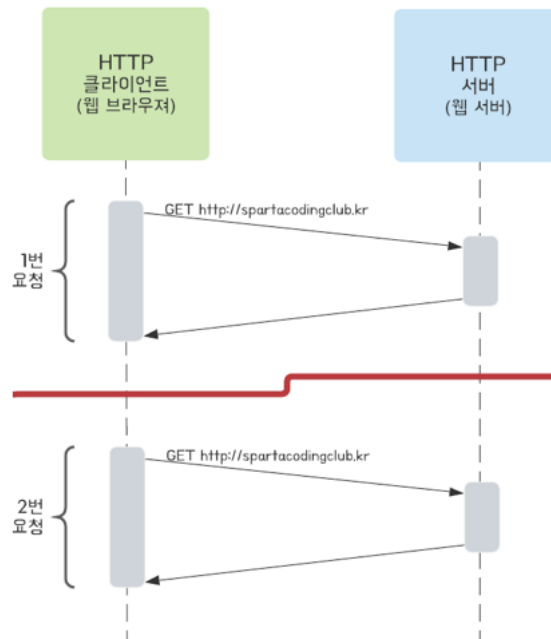
· 🌿 가지등급 이상은 '정보'를 지원합니다. 자격조건이 되는 카페는 카페 지원센터 공식카페에 지원을 신청하세요.

[카페지원센터 가기](#)

03. 쿠키와 세션

▼ 5) 사용자를 구별하지 못 하는 HTTP

- HTTP 는 상태를 저장하지 않습니다. ('Stateless' 하다)
- 아래 그림에서 클라이언트의 요청 (GET http://spartacodingclub.kr)을 서버에게 보낸 후 응답을 받을 때까지가 하나의 HTTP 요청입니다. 하지만 HTTP 상태는 기억되지 않기 때문에 웹 서버에서는 1번과 2번이 같은 클라이언트의 요청인지 알 수 없습니다.



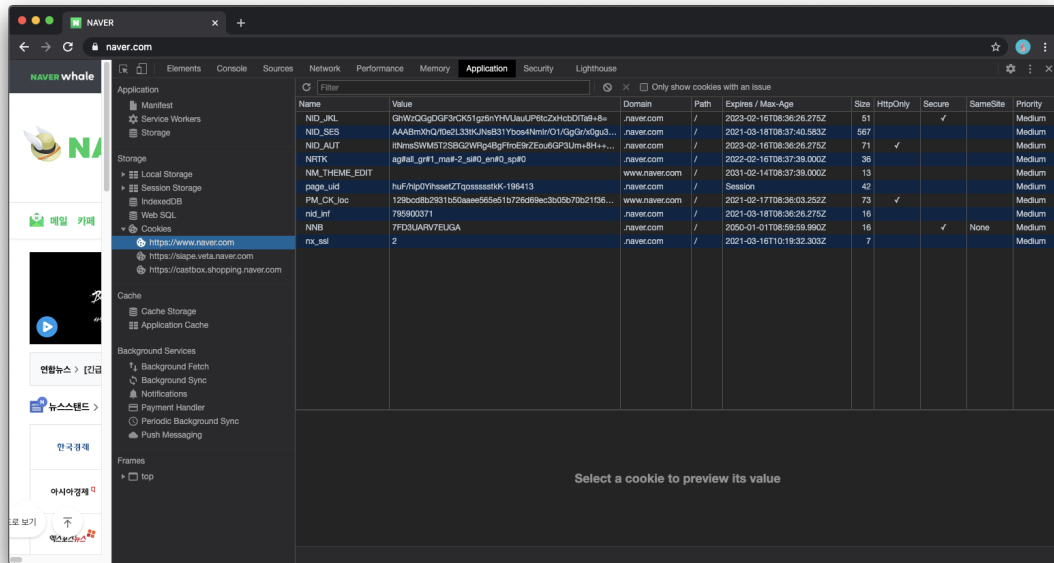
▼ 6) 쿠키와 세션



쿠키와 세션 모두 HTTP 에 상태 정보를 유지(Stateful)하기 위해 사용됩니다. 즉, 쿠키와 세션을 통해 서버에서는 클라이언트 별로 인증 및 인가를 할 수 있게 됩니다.

1. 쿠키

- 클라이언트에 저장될 목적으로 생성한 작은 정보를 담은 파일 입니다.
- 클라이언트인 웹 브라우저에 저장된 '쿠키' 를 확인해 보죠.
 - 크롬 브라우저 기준으로 '개발자도구' 를 열어 보세요.
 - Application - Storage - Cookies 에 도메인 별로 저장되어 있는게 확인 됩니다.

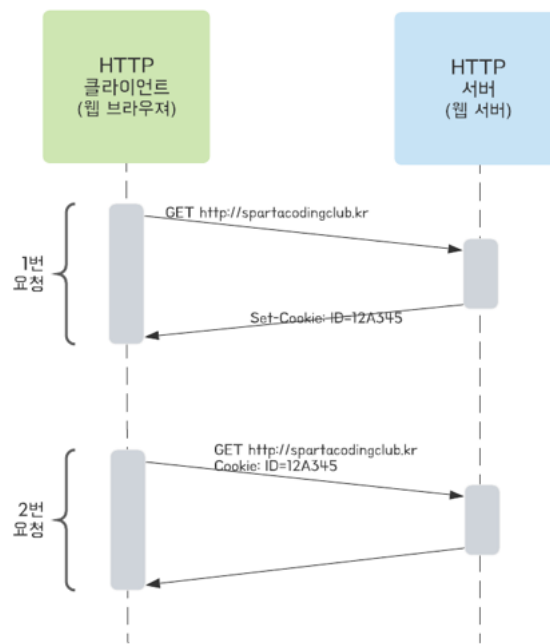


구성요소

- Name (이름): 쿠키를 구별하는 데 사용되는 키 (중복될 수 없음)
- Value (값): 쿠키의 값
- Domain (도메인): 쿠키가 저장된 도메인
- Path (경로): 쿠키가 사용되는 경로
- Expires (만료기한): 쿠키의 만료기한 (만료기한 지나면 삭제됩니다.)

2. 세션

- 서버에서 일정시간 동안 클라이언트 상태를 유지하기 위해 사용
- 서버에서 클라이언트 별로 유일무이한 '세션 ID' 를 부여한 후 클라이언트 별 필요한 정보를 서버에 저장
- 서버에서 생성한 '세션 ID' 는 클라이언트의 쿠키값('세션 쿠키' 라고 부름)으로 저장되어 클라이언트 식별에 사용됨
- 세션 동작 방식



위 그림에서와 같이 서버는 세션ID 를 사용하여 세션을 유지합니다.

1. 클라이언트가 서버에 1번 요청
2. 서버가 세션ID 를 생성하고, 응답 헤더에 전달
 1. 세션 ID 형태: "SESSIONID = 12A345"
3. 클라이언트가 쿠키를 저장 ('세션쿠키')
4. 클라이언트가 서버에 2번 요청
 - 쿠키값 (세션 ID) 포함하여 요청
5. 서버가 세션ID 를 확인하고, 1번 요청과 같은 클라이언트임을 인지

쿠키와 세션 비교

	≡ 쿠키 (Cookie)	≡ 세션 (Session)
설 명	클라이언트에 저장될 목적으로 생성한 작은 정보를 담은 파일	서버에서 일정시간 동안 클라이언트 상태를 유지하기 위해 사용
저 장 위 치	클라이언트 (웹 브라우저)	웹 서버
사 용 예	사이트 팝업의 "오늘 다시보지 않기" 정보 저장	로그인 정보 저장
만 료 시 점	쿠키 저장 시 만료일시 설정 가능 (브라우저 종료 시도 유지 가능)	다음 조건 중 하나가 만족될 경우 만료됨 1. 브라우저 종료 시까지 2. 클라이언트 로그아웃 시까지 3. 서버에 설정한 유지기간까지 해당 클라이언트의 재요청이 없는 경우
용 량 제 한	브라우저 별로 다름 (크롬 기준) - 하나의 도메인 당 180개 - 하나의 쿠키 당 4KB(=4096byte)	개수 제한 없음 (단, 세션 저장소 크기 이상 저장 불가능)

Aa -	≡ 쿠키 (Cookie)	≡ 세션 (Session)
보 안	취약 (클라이언트에서 쿠키 정보를 쉽게 변경, 삭제 및 가로채기 당할 수 있음)	비교적 안전 (서버에 저장되기 때문에 상대적으로 안전)

04. 회원 관리 기능 요구사항 및 설계



'나만의 셀렉샵'에 이용자가 많아지면서, 회원 가입, 로그인 기능을 통해 회원을 관리하고자하는 요구가 생겼습니다.

▼ 7) 요구사항

요구사항

Aa 기능	≡ 일반 사용자	≡ 관리자
<u>회원 가입</u>	- 입력 항목: 아이디, 패스워드, 이메일 - 아이디 중복 불가	- 입력 항목: 아이디, 패스워드, 이메일, 관리자 식별암호 - 아이디 중복 불가
<u>회원 로그인</u>	- 필수 항목: 아이디, 패스워드 - 인증 조건: 회원 가입 시 입력한 아이디와 패스워드가 일치	동일
<u>회원 로그아웃</u>	조건: 로그아웃 버튼을 클릭	동일
<u>회원 별 상품 등록 및 조회</u>	- 회원 별 등록/조회 상품 구분 - 다른 사용자가 등록한 상품들은 조회 불가	모든 사용자가 등록한 상품들 조회 가능

▼ 8) API 설계

회원관리 기능 API 설계

Aa Name	≡ tag	≡ Request	≡ Response
회원 가입	GET /user/signup	-	회원 가입 HTML
제목 없음	POST /user/signup	회원 가입에 필요한 정보	-
회원 로그인	GET /user/login	-	회원 로그인 HTML
제목 없음	POST /user/login	아이디, 패스워드	-
회원 로그아웃	GET /user/logout	-	-

상품 등록 및 조회 API 설계

Aa Name	≡ tag	≡ 설명
상품 등록 (회원별)	POST /api/products	로그인되어 있는 회원 별로 상품 등록
상품 조회 (회원별)	GET /api/products	로그인되어 있는 회원이 등록한 상품만 조회
상품 조회 (관리자용)	GET /api/admin/products	모든 회원의 상품 조회

05. '스프링 시큐리티' 프레임워크

▼ 9) 스프링 시큐리티



'스프링 시큐리티' 프레임워크는 스프링 서버에 필요한 인증 및 인가를 위해 많은 기능을 제공해 줌으로써 개발의 수고를 덜어 줍니다. 마치 '스프링' 프레임워크가 웹 서버 구현에 편의를 제공해 주는 것처럼요!

- '스프링 시큐리티' 프레임워크 추가

▼ [코드스니펫] build.gradle

```
// 스프링 시큐리티
implementation 'org.springframework.boot:spring-boot-starter-security'
// Thymeleaf (뷰 템플릿 엔진)
implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'
```

- '스프링 시큐리티' 활성화

▼ [코드스니펫] src > main > com.sparta.springcore.security > WebSecurityConfig

```
package com.sparta.springcore.security;

import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

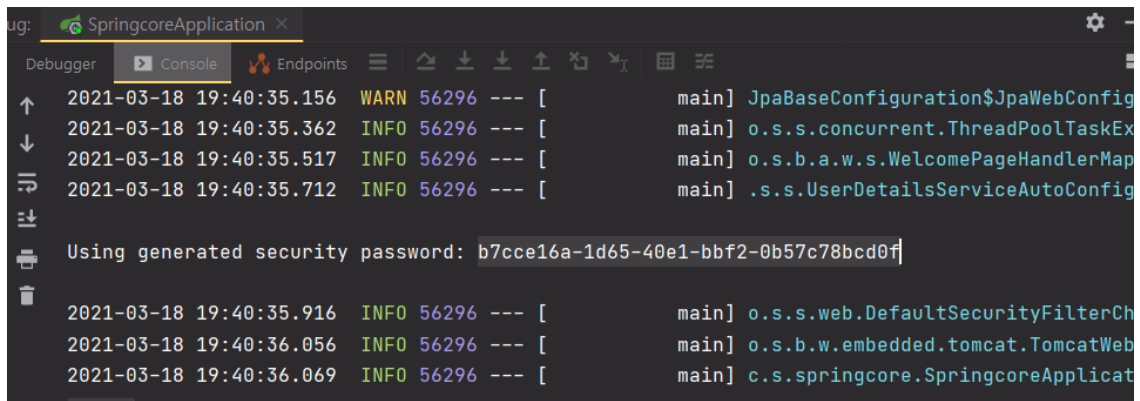
@Configuration
@EnableWebSecurity // 스프링 Security 지원을 가능하게 함
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.csrf().disable();
        http.headers().frameOptions().disable();

        http.authorizeRequests()
            .anyRequest().authenticated()
            .and()
            .formLogin()
            .defaultSuccessUrl("/")
            .permitAll()
            .and()
            .logout()
            .permitAll();
    }
}
```

▼ 10) 스프링 시큐리티의 default 로그인 기능

The screenshot shows a web browser window with the address bar displaying 'localhost:8080/login'. The page content includes the text 'Please sign in' in a large, bold font. Below this text are two white input fields with light blue borders. The first field is labeled 'Username' and the second is labeled 'Password'. Below these fields is a prominent blue button with the text 'Sign in' in white. The background of the page is a light gray.

- Username: **user**
- Password: spring 로그 확인 (서버 시작 시마다 변경됨)



06. 회원 가입 UI 반영

▼ 11) 회원 가입 UI 반영

- 프론트 개발자가 이미 구현한 UI 파일을 서버에 반영한다고 가정

▼ [코드스니펫] src > main > resources > static > login.html

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml" xmlns:th="http://www.thymeleaf.org">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport"
      content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link href="https://fonts.googleapis.com/css2?family=Nanum+Gothic&display=swap" rel="stylesheet">
    <link rel="stylesheet" type="text/css" href="/css/style.css">
    <meta charset="UTF-8">
    <title>로그인 페이지</title>
  </head>
  <body>
    <div id="login-form">
      <div id="login-title">Log into Select Shop</div>
      <button id="login-kakao-btn" onclick="location.href='https://kauth.kakao.com/oauth/authorize?client_id=&redirect_uri=http://localhost:8080/user/kakao/callback&response_type=code'">
        카카오로 로그인하기
      </button>
      <button id="login-id-btn" onclick="location.href='/user/signup'">
        회원 가입하기
      </button>
      <div th:if="${loginError}" class="alert alert-danger" role="alert">로그인에 실패하였습니다.</div>
      <form action="/user/login" method="post">
        <div class="login-id-label">아이디</div>
        <input type="text" name="username" class="login-input-box">

        <div class="login-id-label">비밀번호</div>
        <input type="password" name="password" class="login-input-box">

        <button id="login-id-submit">로그인</button>
      </form>
    </div>
  </body>
</html>
```

▼ [코드스니펫] src > main > resources > static > signup.html

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml" xmlns:th="http://www.thymeleaf.org">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport"
      content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link href="https://fonts.googleapis.com/css2?family=Nanum+Gothic&display=swap" rel="stylesheet">
    <link rel="stylesheet" type="text/css" href="/css/style.css">
    <meta charset="UTF-8">
    <title>회원가입 페이지</title>
```



```

<script>
    function onclickAdmin() {
        // Get the checkbox
        var checkBox = document.getElementById("admin-check");
        // Get the output text
        var box = document.getElementById("admin-token");

        // If the checkbox is checked, display the output text
        if (checkBox.checked == true){
            box.style.display = "block";
        } else {
            box.style.display = "none";
        }
    }
</script>
</head>
<body>
<div id="login-form">
    <div id="login-title">Sign up Select Shop</div>

    <form action="/user/signup" method="post">
        <div class="login-id-label">Username</div>
        <input type="text" name="username" placeholder="Username" class="login-input-box">

        <div class="login-id-label">Password</div>
        <input type="password" name="password" class="login-input-box">

        <div class="login-id-label">E-mail</div>
        <input type="text" name="email" placeholder="E-mail" class="login-input-box">

        <div>
            <input id="admin-check" type="checkbox" name="admin" onclick="onclickAdmin()" style="margin-top: 40px;">관리자
            <input id="admin-token" type="password" name="adminToken" placeholder="관리자 암호" class="login-input-box" style="display:none">
        </div>
        <button id="login-id-submit">회원 가입</button>
    </form>
</div>
</body>
</html>

```

▼ [코드스니펫] src > main > resources > static > css > style.css

```

* {
    font-family: 'Nanum Gothic', sans-serif;
}

body {
    margin: 0px;
}

#search-result-box {
    margin-top: 15px;
}

.search-itemDto {
    width: 530px;
    display: flex;
    flex-direction: row;
    align-content: center;
    justify-content: space-around;
}

.search-itemDto-left img {
    width: 159px;
    height: 159px;
}

.search-itemDto-center {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: space-evenly;
}

.search-itemDto-center div {
    width: 280px;
    height: 23px;
    font-size: 18px;
    font-weight: normal;
    font-stretch: normal;
    font-style: normal;
    line-height: 1.3;
}

```

```

        letter-spacing: -0.9px;
        text-align: left;
        color: #343a40;
        overflow: hidden;
        white-space: nowrap;
        text-overflow: ellipsis;
    }

    .search-itemDto-center div.price {
        height: 27px;
        font-size: 27px;
        font-weight: 600;
        font-stretch: normal;
        font-style: normal;
        line-height: 1;
        letter-spacing: -0.54px;
        text-align: left;
        color: #E8344E;
    }

    .search-itemDto-center span.unit {
        width: 17px;
        height: 18px;
        font-size: 18px;
        font-weight: 500;
        font-stretch: normal;
        font-style: normal;
        line-height: 1;
        letter-spacing: -0.9px;
        text-align: center;
        color: #000000;
    }

    .search-itemDto-right {
        display: inline-block;
        height: 100%;
        vertical-align: middle
    }

    .search-itemDto-right img {
        height: 25px;
        width: 25px;
        vertical-align: middle;
        margin-top: 60px;
        cursor: pointer;
    }

    input#query {
        padding: 15px;
        width: 526px;
        border-radius: 2px;
        background-color: #e9ecef;
        border: none;

        background-image: url('../images/icon-search.png');
        background-repeat: no-repeat;
        background-position: right 10px center;
        background-size: 20px 20px;
    }

    input#query::placeholder {
        padding: 15px;
    }

    button {
        color: white;
        border-radius: 4px;
        border-radius: none;
    }

    .popup-container {
        display: none;
        position: fixed;
        top: 0;
        left: 0;
        bottom: 0;
        right: 0;
        background-color: rgba(0, 0, 0, 0.5);
        align-items: center;
        justify-content: center;
    }

    .popup-container.active {
        display: flex;
    }

    .popup {

```

```

padding: 20px;
box-shadow: 2px 2px 10px rgba(0, 0, 0, 0.3);
position: relative;
width: 370px;
height: 209px;
border-radius: 11px;
background-color: #ffffff;
}

.popup h1 {
margin: 0px;
font-size: 22px;
font-weight: 500;
font-stretch: normal;
font-style: normal;
line-height: 1;
letter-spacing: -1.1px;
color: #000000;
}

.popup input {
width: 320px;
height: 22px;
border-radius: 2px;
border: solid 1.1px #dee2e6;
margin-right: 9px;
margin-bottom: 10px;
padding-left: 10px;
}

.popup button.close {
position: absolute;
top: 15px;
right: 15px;
color: #adb5bd;
background-color: #fff;
font-size: 19px;
border: none;
}

.popup button.cta {
width: 369.1px;
height: 43.9px;
border-radius: 2px;
background-color: #15aabf;
border: none;
}

#search-area, #see-area {
width: 530px;
margin: auto;
}

.nav {
width: 530px;
margin: 30px auto;
display: flex;
align-items: center;
justify-content: space-around;
}

.nav div {
cursor: pointer;
}

.nav div.active {
font-weight: 700;
}

.header {
height: 255px;
box-sizing: border-box;
background-color: #15aabf;
color: white;
text-align: center;
padding-top: 80px;
/*padding: 50px;*/
font-size: 45px;
font-weight: bold;
}

#header-title-login-user {
font-size: 36px;
letter-spacing: -1.08px;
}

#header-title-select-shop {

```

```

        margin-top: 20px;
        font-size: 45px;
        letter-spacing: 1.1px;
    }

    #product-container {
        grid-template-columns: 100px 50px 100px;
        grid-template-rows: 80px auto 80px;
        column-gap: 10px;
        row-gap: 15px;
    }

    .product-card {
        width: 300px;
        margin: auto;
        cursor: pointer;
    }

    .product-card .card-header {
        width: 300px;
    }

    .product-card .card-header img {
        width: 300px;
    }

    .product-card .card-body {
        margin-top: 15px;
    }

    .product-card .card-body .title {
        font-size: 11px;
        font-weight: normal;
        font-stretch: normal;
        font-style: normal;
        line-height: 1;
        letter-spacing: -0.75px;
        text-align: left;
        color: #343a40;
        margin-bottom: 10px;
        overflow: hidden;
        white-space: nowrap;
        text-overflow: ellipsis;
    }

    .product-card .card-body .lprice {
        font-size: 15.8px;
        font-weight: normal;
        font-stretch: normal;
        font-style: normal;
        line-height: 1;
        letter-spacing: -0.79px;
        color: #000000;
        margin-bottom: 10px;
    }

    .product-card .card-body .lprice span {
        font-size: 13.4px;
        font-weight: 600;
        font-stretch: normal;
        font-style: normal;
        line-height: 1;
        letter-spacing: -0.43px;
        text-align: left;
        color: #E8344E;
    }

    .product-card .card-body .isgood {
        margin-top: 10px;
        padding: 10px 20px;
        color: white;
        border-radius: 2.6px;
        background-color: #ff8787;
        width: 48px;
    }

    .pagination {
        margin-top: 12px;
        margin-left: auto;
        margin-right: auto;
        width: 53%;
    }

    .folder-active {
        background-color: crimson !important;;
    }

```

```

.none {
  display: none;
}

.folder-bar {
  width: 100%;
  overflow: hidden;
}

.folder-black, .folder-black:hover {
  color: #fff!important;
  background-color: #000!important;
}

.folder-bar .folder-button {
  white-space: normal;
}

.folder-bar .folder-bar-item {
  padding: 8px 16px;
  float: left;
  width: auto;
  border: none;
  display: block;
  outline: 0;
}

.folder-btn, .folder-button {
  border: none;
  display: inline-block;
  padding: 8px 16px;
  vertical-align: middle;
  overflow: hidden;
  text-decoration: none;
  color: inherit;
  background-color: inherit;
  text-align: center;
  cursor: pointer;
  white-space: nowrap;
}

#login-form {
  width: 538px;
  height: 710px;
  margin: 70px auto 141px auto;
  display: flex;
  flex-direction: column;
  justify-content: flex-start;
  align-items: center;
  /*gap: 96px;*/
  padding: 56px 0 0;
  border-radius: 10px;
  box-shadow: 0 4px 25px 0 rgba(0, 0, 0, 0.15);
  background-color: #ffffff;
}

#login-title {
  width: 303px;
  height: 32px;
  /*margin: 56px auto auto auto;*/
  flex-grow: 0;
  font-family: SpoqaHanSansNeo;
  font-size: 32px;
  font-weight: bold;
  font-stretch: normal;
  font-style: normal;
  line-height: 1;
  letter-spacing: -0.96px;
  text-align: left;
  color: #212529;
}

#login-kakao-btn {
  border-width: 0;
  margin: 96px 0 8px;
  width: 393px;
  height: 62px;
  flex-grow: 0;
  display: flex;
  flex-direction: row;
  justify-content: center;
  align-items: center;
  gap: 10px;
  /*margin: 0 0 8px;*/
  padding: 11px 12px;
  border-radius: 5px;
  background-color: #ffd43b;
}

```

```

    font-family: SpoqaHanSansNeo;
    font-size: 20px;
    font-weight: bold;
    font-stretch: normal;
    font-style: normal;
    color: #414141;
}

#login-id-btn {
    width: 393px;
    height: 62px;
    flex-grow: 0;
    display: flex;
    flex-direction: row;
    justify-content: center;
    align-items: center;
    gap: 10px;
    /*margin: 8px 0 0 0;*/
    padding: 11px 12px;
    border-radius: 5px;
    border: solid 1px #212529;
    background-color: #ffffff;

    font-family: SpoqaHanSansNeo;
    font-size: 20px;
    font-weight: bold;
    font-stretch: normal;
    font-style: normal;
    color: #414141;
}

.login-input-box {
    border-width: 0;

    width: 370px !important;
    height: 52px;
    margin: 14px 0 0;
    border-radius: 5px;
    background-color: #e9ecef;
}

.login-id-label {
    /*width: 44.1px;*/
    /*height: 16px;*/
    width: 382px;
    padding-left: 11px;
    margin-top: 40px;
    /*margin: 0 337.9px 14px 11px;*/
    font-family: NotoSansCJKKR;
    font-size: 16px;
    font-weight: normal;
    font-stretch: normal;
    font-style: normal;
    line-height: 1;
    letter-spacing: -0.8px;
    text-align: left;
    color: #212529;
}

#login-id-submit {
    border-width: 0;
    width: 393px;
    height: 62px;
    flex-grow: 0;
    display: flex;
    flex-direction: row;
    justify-content: center;
    align-items: center;
    gap: 10px;
    margin: 40px 0 0;
    padding: 11px 12px;
    border-radius: 5px;
    background-color: #15aabf;

    font-family: SpoqaHanSansNeo;
    font-size: 20px;
    font-weight: bold;
    font-stretch: normal;
    font-style: normal;
    line-height: 1;
    letter-spacing: normal;
    text-align: center;
    color: #ffffff;
}

#logout-text {
    position: absolute;

```

```

        top:48px;
        right:48px;
        font-size: 18px;
        font-family: SpoqaHanSansNeo;
        font-size: 18px;
        font-weight: 500;
        font-stretch: normal;
        font-style: normal;
        line-height: 1;
        letter-spacing: 0.36px;
        text-align: center;
        color: #ffffff;
    }

    @media ( max-width: 768px ) {
        body {
            margin: 0px;
        }

        #search-result-box {
            margin-top: 15px;
        }

        .search-itemDto {
            width: 330px;
            display: flex;
            flex-direction: row;
            align-content: center;
            justify-content: space-around;
        }

        .search-itemDto-left img {
            width: 80px;
            height: 80px;
        }

        .search-itemDto-center {
            display: flex;
            flex-direction: column;
            align-items: center;
            justify-content: space-evenly;
        }

        .search-itemDto-center div {
            width: 190px;
            height: 23px;
            font-size: 13px;
            font-weight: normal;
            font-stretch: normal;
            font-style: normal;
            line-height: 1.3;
            letter-spacing: -0.9px;
            text-align: left;
            color: #343a40;
            overflow: hidden;
            white-space: nowrap;
            text-overflow: ellipsis;
        }

        .search-itemDto-center div.price {
            height: 27px;
            font-size: 17px;
            font-weight: 600;
            font-stretch: normal;
            font-style: normal;
            line-height: 1;
            letter-spacing: -0.54px;
            text-align: left;
            color: #E8344E;
        }

        .search-itemDto-center span.unit {
            width: 17px;
            height: 18px;
            font-size: 14px;
            font-weight: 500;
            font-stretch: normal;
            font-style: normal;
            line-height: 1;
            letter-spacing: -0.9px;
            text-align: center;
            color: #000000;
        }

        .search-itemDto-right {
            display: inline-block;
            height: 100%;

```

```

        vertical-align: middle;
    }

    .search-itemDto-right img {
        height: 14px;
        width: 14px;
        vertical-align: middle;
        margin-top: 26px;
        cursor: pointer;
        padding-right: 20px;
    }

    input#query {
        padding: 15px;
        width: 290px;
        border-radius: 2px;
        background-color: #e9ecef;
        border: none;

        background-image: url('../images/icon-search.png');
        background-repeat: no-repeat;
        background-position: right 10px center;
        background-size: 20px 20px;
    }

    input#query::placeholder {
        padding: 15px;
    }

    button {
        color: white;
        border-radius: 4px;
        border-radius: none;
    }

    .popup-container {
        display: none;
        position: fixed;
        top: 0;
        left: 0;
        bottom: 0;
        right: 0;
        background-color: rgba(0, 0, 0, 0.5);
        align-items: center;
        justify-content: center;
    }

    .popup-container.active {
        display: flex;
    }

    .popup {
        position: absolute;
        bottom: 0px;
        width: 333px;
        border-radius: 11px 11px 0px 0px;
    }

    .popup input {
        width: 278px !important;
        height: 39px;
        border-radius: 2px;
        border: solid 1.1px #dee2e6;
        margin-right: 9px;
        margin-bottom: 10px;
        padding-left: 10px;
    }

    .popup p {
        font-size: 14px;
    }

    .popup button.close {
        position: absolute;
        top: 15px;
        right: 15px;
        color: #adb5bd;
        background-color: #fff;
        font-size: 19px;
        border: none;
    }

    .popup button.cta {
        width: 319px;
        height: 43.9px;
        border-radius: 2px;
        background-color: #15aabf;
    }

```



```

        border: none;
    }

    #search-area, #see-area {
        width: 330px;
        margin: auto;
    }

    .nav {
        width: 330px;
        margin: 30px auto;
        display: flex;
        align-items: center;
        justify-content: space-around;
    }

    .nav div {
        cursor: pointer;
    }

    .nav div.active {
        font-weight: 700;
    }

    .header {
        height: 255px;
        box-sizing: border-box;
        background-color: #15aabf;
        color: white;
        text-align: center;
        padding-top: 80px;
        /*padding: 50px;*/
        font-size: 45px;
        font-weight: bold;
    }

    /*#product-container {*/
    /*    grid-template-columns: 100px 50px 100px;*/
    /*    grid-template-rows: 80px auto 80px;*/
    /*    column-gap: 10px;*/
    /*    row-gap: 15px;*/
    /*}*/

    /*.product-card {*/
    /*    width: 300px;*/
    /*    margin: auto;*/
    /*    cursor: pointer;*/
    /*}*/

    /*.product-card .card-header {*/
    /*    width: 300px;*/
    /*}*/

    /*.product-card .card-header img {*/
    /*    width: 300px;*/
    /*}*/

    /*.product-card .card-body {*/
    /*    margin-top: 15px;*/
    /*}*/

    /*.product-card .card-body .title {*/
    /*    font-size: 15px;*/
    /*    font-weight: normal;*/
    /*    font-stretch: normal;*/
    /*    font-style: normal;*/
    /*    line-height: 1;*/
    /*    letter-spacing: -0.75px;*/
    /*    text-align: left;*/
    /*    color: #343a40;*/
    /*    margin-bottom: 10px;*/
    /*    overflow: hidden;*/
    /*    white-space: nowrap;*/
    /*    text-overflow: ellipsis;*/
    /*}*/

    /*.product-card .card-body .lprice {*/
    /*    font-size: 15.8px;*/
    /*    font-weight: normal;*/
    /*    font-stretch: normal;*/
    /*    font-style: normal;*/
    /*    line-height: 1;*/
    /*    letter-spacing: -0.79px;*/
    /*    color: #000000;*/
    /*    margin-bottom: 10px;*/
    /*}*/

```

```

/* .product-card .card-body .lprice span {*/
/*   font-size: 21.4px;*/
/*   font-weight: 600;*/
/*   font-stretch: normal;*/
/*   font-style: normal;*/
/*   line-height: 1;*/
/*   letter-spacing: -0.43px;*/
/*   text-align: left;*/
/*   color: #E8344E;*/
/* }*/

/* .product-card .card-body .isgood {*/
/*   margin-top: 10px;*/
/*   padding: 10px 20px;*/
/*   color: white;*/
/*   border-radius: 2.6px;*/
/*   background-color: #ff8787;*/
/*   width: 48px;*/
/* }*/

.none {
  display: none;
}

input#kakao-login {
  padding: 15px;
  width: 526px;
  border-radius: 2px;
  background-color: #e9ecef;
  border: none;

  background-image: url('images/kakao_login_medium_narrow.png');
  background-repeat: no-repeat;
  background-position: right 10px center;
  background-size: 20px 20px;
}

.autocomplete {
  /*the container must be positioned relative:*/
  position: relative;
  display: inline-block;
}

input {
  border: 1px solid transparent;
  background-color: #f1f1f1;
  padding: 10px;
  font-size: 16px;
}

input[type=text] {
  background-color: #f1f1f1;
  /*width: 100%;*/
}

input[type=password] {
  background-color: #f1f1f1;
  /*width: 100%;*/
}

input[type=submit] {
  background-color: DodgerBlue;
  color: #fff;
}

.autocomplete-items {
  position: absolute;
  border: 1px solid #d4d4d4;
  border-bottom: none;
  border-top: none;
  z-index: 99;
  /*position the autocomplete items to be the same width as the container:*/
  top: 100%;
  left: 0;
  right: 0;
}

.autocomplete-items div {
  padding: 10px;
  cursor: pointer;
  background-color: #fff;
  border-bottom: 1px solid #d4d4d4;
}

.autocomplete-items div:hover {
  /*when hovering an item:*/
  background-color: #e9e9e9;
}

.autocomplete-active {
  /*when navigating through the items using the arrow keys:*/
  background-color: DodgerBlue !important;
  color: #ffffff;
}

```

```

.alert-danger {
    color: #721c24;
    background-color: #f8d7da;
    border-color: #f5c6cb;
}

.alert {
    width: 300px;
    margin-top: 22px;
    padding: 1.75rem 1.25rem;
    border: 1px solid transparent;
    border-radius: .25rem;
}

```

- 회원 가입 페이지 요청 처리

▼ [코드스니펫] src > main > resources > application.properties

```
spring.thymeleaf.prefix=classpath:/static/
```

▼ [코드스니펫] src > main > com.sparta.springcore.security > WebSecurityConfig

```

.formLogin()
.loginPage("/user/login")
.failureUrl("/user/login/error")

```

▼ [코드스니펫] src > main > com.sparta.springcore.controller > UserController

```

package com.sparta.springcore.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class UserController {

    // 회원 로그인 페이지
    @GetMapping("/user/login")
    public String login() {
        return "login";
    }

    @GetMapping("/user/login/error")
    public String loginError(Model model) {
        model.addAttribute("loginError", true);
        return "login";
    }

    // 회원 가입 페이지
    @GetMapping("/user/signup")
    public String signup() {
        return "signup";
    }
}

```

▼ 12) UI 이슈 해결

- 기대하는 UI

Log into Select Shop

카카오로 로그인하기

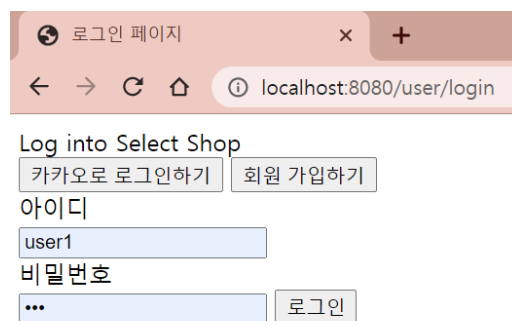
회원 가입하기

아이디

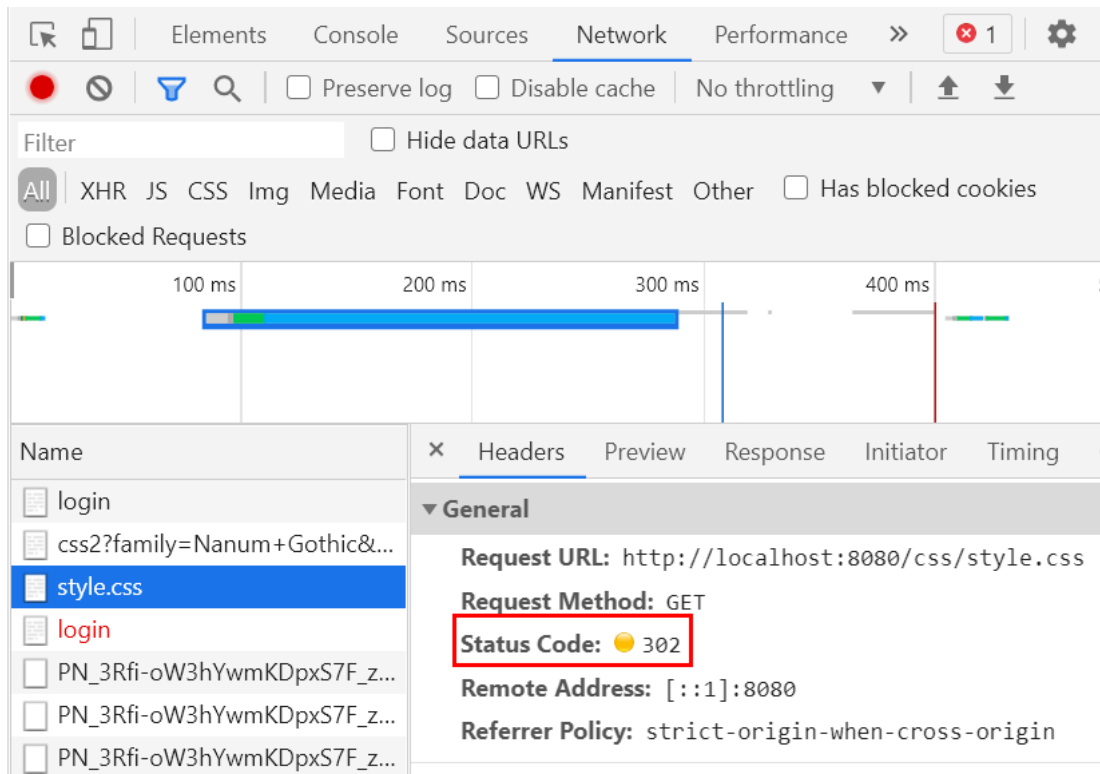
비밀번호

로그인

- 로그인 페이지에 CSS 가 적용되지 않음
- 웹 페이지 이슈가 생겼을 때 원인을 파악하고 해결할 수 있는 능력 필요



- '개발자 도구' 로 에러 내용 확인



- 스프링 시큐리티의 URL 허용 정책 변경 필요 확인

▼ [코드스니펫] src > main > com.sparta.springcore.security > WebSecurityConfig

```
http.authorizeRequests()
    // image 폴더를 login 없이 허용
    .antMatchers("/images/**").permitAll()
    // css 폴더를 login 없이 허용
    .antMatchers("/css/**").permitAll()
    // 그 외 모든 요청은 인증과정 필요
    .anyRequest().authenticated()
    .and()
    .formLogin()
    .loginPage("/user/login")
    .loginProcessingUrl("/user/login")
    .defaultSuccessUrl("/")
    .permitAll()
    .and()
    .logout()
    .permitAll();
```

07. 회원 가입 기능 구현



지금부터 '스프링 시큐리티' 프레임워크를 사용해서 회원 가입 기능을 구현해 보겠습니다.

▼ 13) 회원 가입 테이블 설계

- 요구사항과 디자인 UI 를 확인하며 어떤 정보가 저장되어야 할지 정리

Sign up Select Shop

Username

Password

E-mail

☒ 관리자
 관리자 암호

▼ DB 에 저장할 회원정보 Entity 설계

User 테이블

text	Field	Type
테이블 ID	<u>id</u>	Long
회원 ID	<u>username</u>	String
패스워드	<u>password</u>	String
이메일 주소	<u>email</u>	String
역할	<u>role</u>	Enum 1) 사용자: USER 2) 관리자: ADMIN

▼ [코드스니펫] src > main > com.sparta.springcore.model > User

```
package com.sparta.springcore.model;

import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

import javax.persistence.*;

@Setter
@Getter // get 함수를 일괄적으로 만들어줍니다.
@NoArgsConstructor // 기본 생성자를 만들어줍니다.
@Entity // DB 테이블 역할을 합니다.
public class User extends Timestamped {

    public User(String username, String password, String email, UserRole role) {
        this.username = username;
        this.password = password;
        this.email = email;
        this.role = role;
    }

    // ID가 자동으로 생성 및 증가합니다.
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Id
    private Long id;

    // 반드시 값을 가지도록 합니다.
    @Column(nullable = false)
    private String username;

    @Column(nullable = false)
    private String password;
}
```

```

@Column(nullable = false)
private String email;

@Column(nullable = false)
@Enumerated(value = EnumType.STRING)
private UserRole role;
}

```

▼ [코드스니펫] src > main > com.sparta.springcore.model > UserRole

```

package com.sparta.springcore.model;

public enum UserRole {
    USER, // 사용자 권한
    ADMIN // 관리자 권한
}

```

▼ 14) 관리자 회원 가입 방법

- '관리자 토큰' 을 알고 있어야 함

```

AAABnv/xRVkLrnYxKZ0aHgTBcXukeZygoC

```



잠깐!

실제로 '관리자' 권한을 이렇게 영성하게 인가하는 경우는 드뭅니다.

일부러 관리자 암호를 의미 없고 긴 문자로 구성했지만, 해커가 해당 암호를 갈취하게 되면 관리자 권한을 너무 쉽게 획득할 수 있게 되겠죠?

보통 현업에서는 '관리자' 권한을 부여할 수 있는 관리자 페이지를 구현하거나, 결재 과정을 통해 '관리자' 권한을 획득하는 방식으로 구현합니다.

▼ 15) 회원가입 API 구현

▼ [코드스니펫] src > main > com.sparta.springcore.dto> SignupRequestDto

```

package com.sparta.springcore.dto;

import lombok.Getter;
import lombok.Setter;

@Setter
@Getter
public class SignupRequestDto {
    private String username;
    private String password;
    private String email;
    private boolean admin = false;
    private String adminToken = "";
}

```

▼ [코드스니펫] src > main > com.sparta.springcore.controller > UserController

```

package com.sparta.springcore.controller;

import com.sparta.springcore.dto.SignupRequestDto;
import com.sparta.springcore.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;

@Controller
public class UserController {

```

```

private final UserService userService;

@Autowired
public UserController(UserService userService) {
    this.userService = userService;
}

// 회원 로그인 페이지
@GetMapping("/user/login")
public String login() {
    return "login";
}

@GetMapping("/user/login/error")
public String loginError(Model model) {
    model.addAttribute("loginError", true);
    return "login";
}

// 회원 가입 페이지
@GetMapping("/user/signup")
public String signup() {
    return "signup";
}

// 회원 가입 요청 처리
@PostMapping("/user/signup")
public String registerUser(SignupRequestDto requestDto) {
    userService.registerUser(requestDto);
    return "redirect:/";
}
}

```

- 회원 가입 요청 처리

1. 회원 중복 Id 확인
2. 관리자 가입 요청에 대해서는 '관리자 토큰' 인가

▼ [코드스니펫] src > main > com.sparta.springcore.service > UserService

```

package com.sparta.springcore.service;

import com.sparta.springcore.dto.SignupRequestDto;
import com.sparta.springcore.model.User;
import com.sparta.springcore.model.UserRole;
import com.sparta.springcore.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;

import java.util.Optional;

@Service
public class UserService {
    private final UserRepository userRepository;
    private static final String ADMIN_TOKEN = "AAABnv/xRVklrnYxKZ0aHgTBcXukeZygoC";

    @Autowired
    public UserService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    public void registerUser(SignupRequestDto requestDto) {
        String username = requestDto.getUsername();
        String password = requestDto.getPassword();
        // 회원 ID 중복 확인
        Optional<User> found = userRepository.findByUsername(username);
        if (found.isPresent()) {
            throw new IllegalArgumentException("중복된 사용자 ID 가 존재합니다.");
        }

        String email = requestDto.getEmail();
        // 사용자 ROLE 확인
        UserRole role = UserRole.USER;
        if (requestDto.isAdmin()) {
            if (!requestDto.getAdminToken().equals(ADMIN_TOKEN)) {
                throw new IllegalArgumentException("관리자 암호가 틀려 등록이 불가능합니다.");
            }
            role = UserRole.ADMIN;
        }
    }
}

```



```

    }

    User user = new User(username, password, email, role);
    userRepository.save(user);
}
}

```

▼ [코드스니펫] src > main > com.sparta.springcore.repository > UserRepository

```

package com.sparta.springcore.repository;

import com.sparta.springcore.model.User;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.Optional;

public interface UserRepository extends JpaRepository<User, Long> {
    Optional<User> findByUsername(String username);
}

```

- 접속 시도 시 또 막힘..

```

// 회원 관리 URL 전부를 login 없이 허용
.antMatchers("/user/**").permitAll()

```

▼ h2 console 통해 저장된 회원 가입 정보 확인

- 접속 시도 막힘..

```

// h2-console URL 을 login 없이 허용
.antMatchers("/h2-console/**").permitAll()

```

08. 패스워드 암호화 구현

▼ 16) 패스워드 암호화

- h2 console 을 통해 DB 에 저장된 패스워드 확인
- 문제점: 패스워드가 평문(=있는 그대로)으로 저장되어 있음



회원 등록 시 '비밀번호'는 사용자가 입력한 문자 그대로 DB 에 안 된다는 사실!!
'정보통신망법, 개인정보보호법' 에 의해 비밀번호는 암호화(Encryption)가 의무!!

암호화해야 하는 개인정보	정보통신망법	개인정보보호법	적용 암호기술
비밀번호	○	○	해시함수
바이오정보	○	○	블록암호
주민등록번호	○	○	
신용카드번호	○	○	
계좌번호	○	○	
여권번호	○	○	
운전면허번호	○	○	
외국인등록번호	○	○	

출처: KISA 개인정보 종류와 적용 가능 암호기술

예를 들어보겠습니다.

앨리스가 여러분의 사이트에 회원가입을 하며 아이디, 패스워드를 입력하였습니다.

- 아이디: alice
- 패스워드: nobodynobody

아무도 알 수 없기를 바라며 적은 패스워드를 아래와 같이 DB 에 평문 그대로 저장해 두었다고 해보죠.

```
SELECT USERNAME,PASSWORD FROM USER WHERE USER.USERNAME='alice';
```

USERNAME	PASSWORD
alice	nobodynobody

(1 row, 1 ms)

만약 해커에 의해 회원정보가 갈취당한다면 앨리스의 패스워드는 모두가 알게 됩니다. 꼭 해커 뿐만 아니겠죠. DB 조회가 가능한 내부 관계자들도 앨리스의 패스워드를 보자마자 영원히 기억해 버릴지도 모릅니다.

그래서 저희는 해시함수를 통해 아래와 같이 암호화된 형태로 패스워드를 저장하겠습니다.

```
SELECT USERNAME,PASSWORD FROM USER WHERE USER.USERNAME='alice';
```

USERNAME	PASSWORD
alice	\$2a\$10\$18eX3uOIEAWIU196xiGFPOov5a/d29n9gnIApTLKmGPzsbwvc.IW

(1 row, 2 ms)

이제 해커가 회원정보를 갈취하더라도 앨리스의 패스워드를 알 수 없습니다. 암호화는 되지만 복호화는 되지 않는 '일방향' 암호 알고리즘을 사용했기 때문입니다.

- 암호화 (O): "nobodynobody" → "\$2a\$10\$.."
- 복호화 (X): "\$2a\$10\$.." → "nobodynobody"

그럼 사용자가 로그인할 때는 암호화된 패스워드를 기억해야 하나요?

사용자는 본인이 기억하는 패스워드를 입력하면 우리가 구현한 서버에서 해당 암호를 암호화하고 DB 에 저장된 암호화된 패스워드와 비교합니다. 이렇게 서버에서는 사용자의 평문 암호를 모르지만 사용자가 등록한 패스워드와 일치하는지 여부는 알 수 있죠.

	EMAIL	PASSWORD	ROLE	USERNAME
49:05.445	user1@naver.com	\$2a\$10\$Tak3CqAbK2njdTBy9OMF0up0LGCWo0HAS7Bz8BH2ytVeHg6LdcdJK	ROLE_USER	user1
49:05.472	user2@naver.com	\$2a\$10\$MYdoVbz/r8VcNF0qkqalecDJtbkMMU06L/6ADkN6Zly0wWk3syUm	ROLE_USER	user2
49:05.472	admin1@naver.com	\$2a\$10\$/q6m5MXkpYBzCdP0W44KeY3h5GzV92cYoChRNXRgZleK.ww.RVQm	ROLE_ADMIN	admin1
49:25.736	hong@naver.com	\$2a\$10\$/dEP0tXEUAlcP6QzCWHuOIUSZWSo0ihtT.N5/3pmaPBj3NUyUP/a	ROLE_USER	alice

저희는 스프링 시큐리티에서 제공해 주고, 권고되고 있는 **'BCrypt'** 해시함수를 사용해 패스워드를 암호화하여 DB 에 저장하겠습니다.

▼ [코드스니펫] src > main > com.sparta.springcore.security > WebSecurityConfig

```
@Bean
public BCryptPasswordEncoder encodePassword() {
    return new BCryptPasswordEncoder();
}
```

▼ [코드스니펫] src > main > com.sparta.springcore.service > UserService

```
package com.sparta.springcore.service;

import com.sparta.springcore.dto.SignupRequestDto;
import com.sparta.springcore.model.User;
import com.sparta.springcore.model.UserRole;
import com.sparta.springcore.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;

import java.util.Optional;

@Service
public class UserService {
    private final PasswordEncoder passwordEncoder;
    private final UserRepository userRepository;
    private static final String ADMIN_TOKEN = "AAABnv/xRVklrnYxKZ0aHgTBcXukeZygoC";

    @Autowired
    public UserService(UserRepository userRepository, PasswordEncoder passwordEncoder) {
        this.userRepository = userRepository;
        this.passwordEncoder = passwordEncoder;
    }

    public void registerUser(SignupRequestDto requestDto) {
        String username = requestDto.getUsername();
        // 회원 ID 중복 확인
        Optional<User> found = userRepository.findByUsername(username);
        if (found.isPresent()) {
            throw new IllegalArgumentException("중복된 사용자 ID 가 존재합니다.");
        }

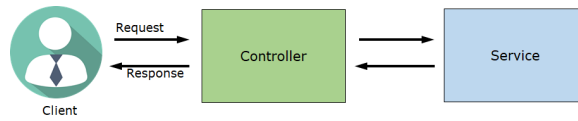
        // 패스워드 인코딩
        String password = passwordEncoder.encode(requestDto.getPassword());
        String email = requestDto.getEmail();
        // 사용자 ROLE 확인
        UserRole role = UserRole.USER;
        if (requestDto.isAdmin()) {
            if (!requestDto.getAdminToken().equals(ADMIN_TOKEN)) {
                throw new IllegalArgumentException("관리자 암호가 틀려 등록이 불가능합니다.");
            }
            role = UserRole.ADMIN;
        }

        User user = new User(username, password, email, role);
        userRepository.save(user);
    }
}
```

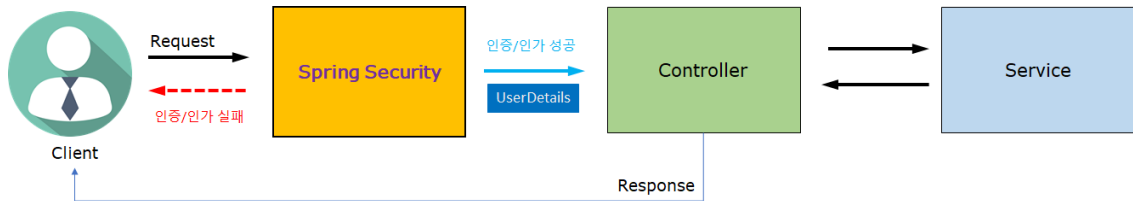
09. 로그인, 로그아웃 기능 구현

▼ 17) 스프링 시큐리티를 이용한 로그인 처리 구현

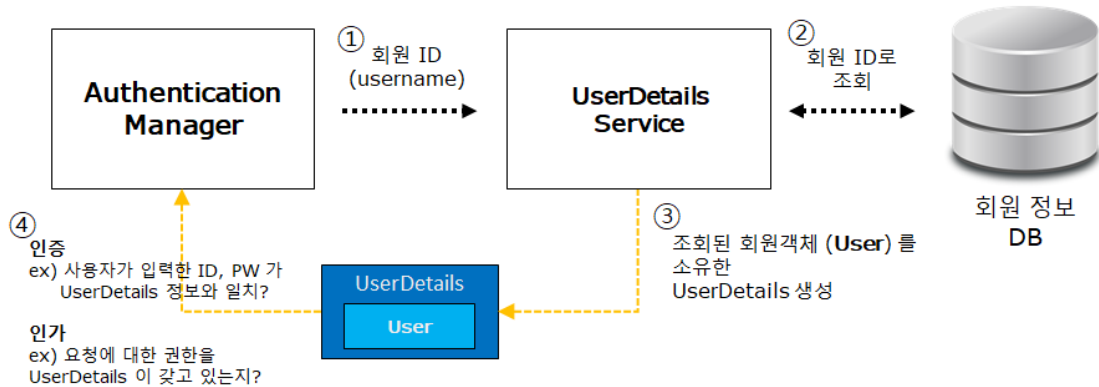
- 스프링 시큐리티 사용 전



- 스프링 시큐리티 사용 후



- 로그인 처리 과정



- 인증/인가 성공 시에만, Controller 에게 회원 정보 전달 (UserDetails)
- 우리가 구현해 줘야 할 클래스
 - UserDetailsService 인터페이스 → **UserDetailsServiceImpl** 클래스
 - UserDetails 인터페이스 → **UserDetailsImpl** 클래스

▼ [코드스니펫] src > main > com.sparta.springcore.security > UserDetailsServiceImpl

```

package com.sparta.springcore.security;

import com.sparta.springcore.model.User;
import com.sparta.springcore.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

import java.util.Optional;

@Service
public class UserDetailsServiceImpl implements UserDetailsService {

    @Autowired
    private UserRepository userRepository;

    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
  
```

```

        User user = userRepository.findByUsername(username)
            .orElseThrow(() -> new UsernameNotFoundException("Can't find " + username));

        return new UserDetailsImpl(user);
    }
}

```

▼ [코드스니펫] src > main > com.sparta.springcore.security > UserDetailsImpl

```

package com.sparta.springcore.security;

import com.sparta.springcore.model.User;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

import java.util.Collection;
import java.util.Collections;

public class UserDetailsImpl implements UserDetails {

    private final User user;

    public UserDetailsImpl(User user) {
        this.user = user;
    }

    public User getUser() {
        return user;
    }

    @Override
    public String getPassword() {
        return user.getPassword();
    }

    @Override
    public String getUsername() {
        return user.getUsername();
    }

    @Override
    public boolean isAccountNonExpired() {
        return true;
    }

    @Override
    public boolean isAccountNonLocked() {
        return true;
    }

    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }

    @Override
    public boolean isEnabled() {
        return true;
    }

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        return Collections.emptyList();
    }
}

```

▼ 18) 회원 로그인 페이지 구현

1. 로그인 페이지

Log into Select Shop

카카오로 로그인하기

회원 가입하기

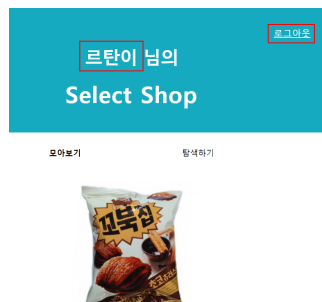
아이디

비밀번호

로그인

2. 로그인 성공 시 페이지

- 로그인된 사용자의 username 이 표시됨
- 로그아웃 버튼 생성



▼ [코드스니펫] src > main > resources > static > index.html

```

<!doctype html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport"
    content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <meta property="og:title" content="00만의 셀렉샵">
  <meta property="og:description" content="관심상품을 선택하고, 최저가 알림을 확인해보세요!">
  <meta property="og:image" content="images/og_selectshop.png">
  <link href="https://fonts.googleapis.com/css2?family=Nanum+Gothic&display=swap" rel="stylesheet">
  <link rel="stylesheet" href="css/style.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script src="basic.js"></script>
  <title>나만의 셀렉샵</title>
</head>
<body>
  <div class="header" style="position:relative;">
    <div id="header-title-login-user">
      <span th:text="${username}"></span> 님의
    </div>
    <div id="header-title-select-shop">
      Select Shop
    </div>
  </div>

```

```

<a id="logout-text" href="/user/logout">
로그아웃
</a>
</div>
<div class="nav">
<div class="nav-see active">
모아보기
</div>
<div class="nav-search">
탐색하기
</div>
</div>
<div id="see-area">
<div id="product-container">

</div>
</div>
<div id="search-area">
<div>
<input type="text" id="query">
<!-- 
</div>
<div id="search-result-box">

</div>
<div id="container" class="popup-container">
<div class="popup">
<button id="close" class="close">
X
</button>
<h1>❗ 최저가 설정하기</h1>
<p>최저가를 설정해두면 선택하신 상품의 최저가가 났을 때<br/> 표시해드려요!</p>
<div>
<input type="text" id="myprice" placeholder="200,000">원
</div>
<button class="cta" onclick="setMyprice()">설정하기</button>
</div>
</div>
</div>
</body>
</html>

```

▼ [코드스니펫] src > main > com.sparta.springcore > controller > HomeController

```

package com.sparta.springcore.controller;

import com.sparta.springcore.security.UserDetailsImpl;
import org.springframework.security.core.annotation.AuthenticationPrincipal;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class HomeController {
    @GetMapping("/")
    public String home(Model model, @AuthenticationPrincipal UserDetailsImpl userDetails) {
        model.addAttribute("username", userDetails.getUsername());
        return "index";
    }
}

```

▼ 19) 회원 로그아웃

👉 아래와 화면에서와 같이 웹 페이지에서 회원 '로그아웃' 버튼을 클릭 시 **'GET /user/logout'** 이 호출되고, 아래 설정만으로 스프링 시큐리티가 로그아웃 처리를 해 줍니다.

- src > main > com.sparta.springcore.security > WebSecurityConfig

```

.logout()
.logoutUrl("/user/logout")
.permitAll();

```

10. 회원별 상품 등록 및 조회

▼ 20) 상품 등록 (회원별)

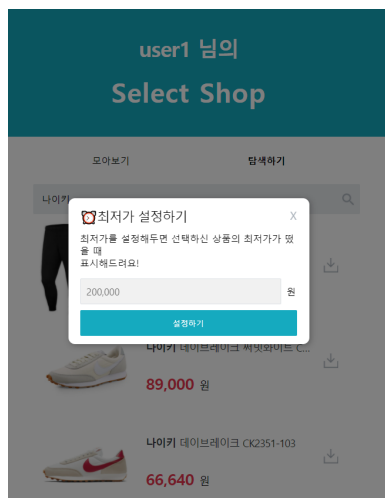
☞ 상품 등록의 경우 UI 변경은 없으나, 로그인한 회원 정보가 함께 저장 되어야 합니다. 왜냐하면 기존과 다르게 누가 등록한 상품인지 구분이 필요해진 거죠. 그래야 상품 조회 시 로그인한 회원마다 등록된 상품이 달라질 수 있으니까요.

사용자 뿐만 아니라 관리자도 상품 등록이 가능합니다.

그래서 Product 클래스에 상품을 등록한 회원 ID 를 저장합니다.

지금은 모든 회원에게 모든 관심상품이 보이고 있음 ⇒ 실습으로 확인!

앞으로 모든 관심상품은 회원 Id 를 갖도록 구현



▼ [코드스니펫] src > main > com.sparta.springcore.controller > ProductController

```
// 신규 상품 등록
@PostMapping("/api/products")
public Product createProduct(@RequestBody ProductRequestDto requestDto, @AuthenticationPrincipal UserDetailsImpl userDetails) {
    // 로그인 되어 있는 ID
    Long userId = userDetails.getUser().getId();

    Product product = productService.createProduct(requestDto, userId);
    // 응답 보내기
    return product;
}
```

▼ [코드스니펫] src > main > com.sparta.springcore.service> ProductService

```
@Transactional // 메소드 동작이 SQL 쿼리문임을 선언합니다.
public Product createProduct(ProductRequestDto requestDto, Long userId) {
    // 요청받은 DTO 로 DB에 저장할 객체 만들기
    Product product = new Product(requestDto, userId);
    productRepository.save(product);
    return product;
}
```

▼ [코드스니펫] src > main > com.sparta.springcore.model > Product

```
@Column(nullable = false)
private Long userId;

// 관심 상품 생성 시 이용합니다.
public Product(ProductRequestDto requestDto, Long userId) {
    // 관심상품을 등록한 회원 Id 저장
    this.userId = userId;
}
```



```

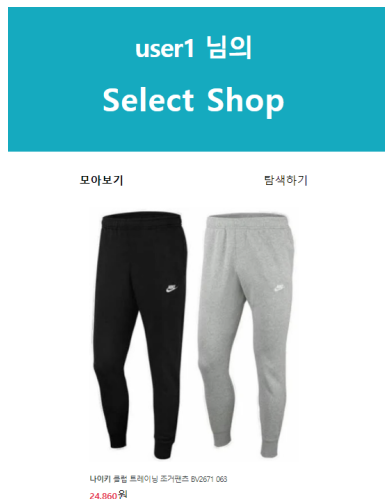
        this.title = requestDto.getTitle();
        this.image = requestDto.getImage();
        this.link = requestDto.getLink();
        this.lprice = requestDto.getLprice();
        this.myprice = 0;
    }

```

- H2 console 통해 회원 ID 로 잘 저장되는지 확인

▼ 21) 상품 조회 (회원별)

👉 회원 별 등록된 상품을 조회합니다.



▼ [코드스니펫] src > main > com.sparta.springcore.controller > ProductController

```

// 로그인한 회원이 등록한 상품들 조회
@GetMapping("/api/products")
public List<Product> getProducts(@AuthenticationPrincipal UserDetailsImpl userDetails) {
    Long userId = userDetails.getUser().getId();
    return productService.getProducts(userId);
}

```

▼ [코드스니펫] src > main > com.sparta.springcore.service > ProductService

```

// 회원 ID 로 등록된 모든 상품 조회
public List<Product> getProducts(Long userId) {
    return productRepository.findAllByUserId(userId);
}

```

▼ [코드스니펫] src > main > com.sparta.springcore.repository > ProductRepository

```

public interface ProductRepository extends JpaRepository<Product, Long> {
    List<Product> findAllByUserId(Long userId);
}

```

- 구현 완료 후 UI 테스트 검증

▼ 22) 상품 조회 (관리자용)



관리자에게는 회원 전체가 등록한 모든 상품을 조회할 수 있는 API 를 제공합니다. 사용자는 해당 API 에 접근하지 못하게 하는 방법은 뒤에서 다룹니다.

▼ [코드스니펫] src > main > com.sparta.springcore.controller > HomeController

```
@GetMapping("/admin")
public String admin(Model model, @AuthenticationPrincipal UserDetailsImpl userDetails) {
    model.addAttribute("username", userDetails.getUsername());
    model.addAttribute("admin", true);
    return "index";
}
```

▼ [코드스니펫] src > main > resources > static > index.html

```
<!doctype html>
<html lang="ko" xmlns="http://www.w3.org/1999/xhtml" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<meta name="viewport"
    content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<meta property="og:title" content="00만의 셀렉샵">
<meta property="og:description" content="관심상품을 선택하고, 최저가 알림을 확인해보세요!">
<meta property="og:image" content="images/og_selectshop.png">
<link href="https://fonts.googleapis.com/css2?family=Nanum+Gothic&display=swap" rel="stylesheet">
<link rel="stylesheet" href="css/style.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script src="basic.js"></script>
<title>나만의 셀렉샵</title>
</head>
<body>
<div class="header" style="position:relative;">
<div id="header-title-login-user">
<span th:text="${username}"></span> 님의
</div>
<div id="header-title-select-shop">
Select Shop
</div>
<a id="logout-text" href="/user/logout">
로그아웃
</a>
</div>
<div class="nav">
<div class="nav-see active">
모아보기
</div>
<div class="nav-search">
탐색하기
</div>
</div>
<div id="see-area">
<div id="product-container">
</div>
</div>
<div id="search-area">
<div>
<input type="text" id="query">
<!-- -->
</div>
<div id="search-result-box">
</div>
<div id="container" class="popup-container">
<div class="popup">
<button id="close" class="close">
X
</button>
<h1>🔴 최저가 설정하기</h1>
<p>최저가를 설정해두면 선택하신 상품의 최저가가 땔 때<br/> 표시해드려요!</p>
<div>
<input type="text" id="myprice" placeholder="200,000">원
</div>
<button class="cta" onclick="setMyprice()">설정하기</button>
</div>
</div>
<div th:if="${admin}" id="admin"></div>
</body>
</html>
```

▼ [코드스니펫] src > main > resources > static > basic.js

```

let targetId;

$(document).ready(function () {
    // id 가 query 인 녀석 위에서 엔터를 누르면 execSearch() 함수를 실행하라는 뜻입니다.
    $('#query').on('keypress', function (e) {
        if (e.key == 'Enter') {
            execSearch();
        }
    });
    $('#close').on('click', function () {
        $('#container').removeClass('active');
    })

    $('#nav div.nav-see').on('click', function () {
        $('#div.nav-see').addClass('active');
        $('#div.nav-search').removeClass('active');

        $('#see-area').show();
        $('#search-area').hide();
    })
    $('#nav div.nav-search').on('click', function () {
        $('#div.nav-see').removeClass('active');
        $('#div.nav-search').addClass('active');

        $('#see-area').hide();
        $('#search-area').show();
    })

    $('#see-area').show();
    $('#search-area').hide();

    if ($('#admin').length === 1) {
        showProduct(true);
    } else {
        showProduct();
    }
})

function showProduct(isAdmin = false) {
    // 1. GET /api/products 요청
    // 2. #product-container(관심상품 목록), #search-result-box(검색결과 목록) 비우기
    // 3. for 문 마다 addProductItem 함수 실행시키고 HTML 만들어서 #product-container 에 붙이기
    $.ajax({
        type: 'GET',
        url: isAdmin ? '/api/admin/products' : '/api/products',
        success: function (response) {
            $('#product-container').empty();
            $('#search-result-box').empty();
            for (let i = 0; i < response.length; i++) {
                let product = response[i];
                let tempHtml = addProductItem(product);
                $('#product-container').append(tempHtml);
            }
        }
    })
}

function addProductItem(product) {
    return `<div class="product-card" onclick="window.location.href='${product.link}'">
        <div class="card-header">
            
        </div>
        <div class="card-body">
            <div class="title">
                ${product.title}
            </div>
            <div class="lprice">
                <span>${numberWithCommas(product.lprice)}</span>원
            </div>
            <div class="isgood ${product.lprice > product.myprice ? 'none' : ''}">
                최저가
            </div>
        </div>
    </div>`;
}

function numberWithCommas(x) {
    return x.toString().replace(/\B(?=(\d{3})+(?!\d))/g, ",");
}

function execSearch() {
    let query = $('#query').val();

```

```

    if (query == '') {
        alert('검색어를 입력해주세요');
        $('#query').focus();
        return;
    }
    $.ajax({
        type: 'GET',
        url: `/api/search?query=${query}`,
        success: function (response) {
            $('#search-result-box').empty();
            for (let i = 0; i < response.length; i++) {
                let itemDto = response[i];
                let tempHtml = addHTML(itemDto);
                $('#search-result-box').append(tempHtml);
            }
        }
    })
}

function addHTML(itemDto) {
    return `<div class="search-itemDto">
        <div class="search-itemDto-left">
            
        </div>
        <div class="search-itemDto-center">
            <div>${itemDto.title}</div>
            <div class="price">
                ${numberWithCommas(itemDto.lprice)}
                <span class="unit">원</span>
            </div>
        </div>
        <div class="search-itemDto-right">
            
        </div>
    </div>`
}

function addProduct(itemDto) {
    $.ajax({
        type: "POST",
        url: '/api/products',
        contentType: "application/json",
        data: JSON.stringify(itemDto),
        success: function (response) {
            $('#container').addClass('active');
            targetId = response.id;
        }
    })
}

function setMyprice() {
    let myprice = $('#myprice').val();
    if (myprice == '') {
        alert('올바른 가격을 입력해주세요');
        return;
    }
    $.ajax({
        type: "PUT",
        url: `/api/products/${targetId}`,
        contentType: "application/json",
        data: JSON.stringify({myprice: myprice}),
        success: function (response) {
            $('#container').removeClass('active');
            alert('성공적으로 등록되었습니다.');
```

▼ [코드스니펫] src > main > com.sparta.springcore.controller > ProductController

```

// (관리자용) 등록된 모든 상품 목록 조회
@GetMapping("/api/admin/products")
public List<Product> getAllProducts() {
    return productService.getAllProducts();
}

```

▼ [코드스니펫] src > main > com.sparta.springcore.controller > ProductService

```

// 모든 상품 조회 (관리자용)
public List<Product> getAllProducts() {

```

```
return productRepository.findAll();
}
```

- 구현 완료 후 UI 테스트 검증

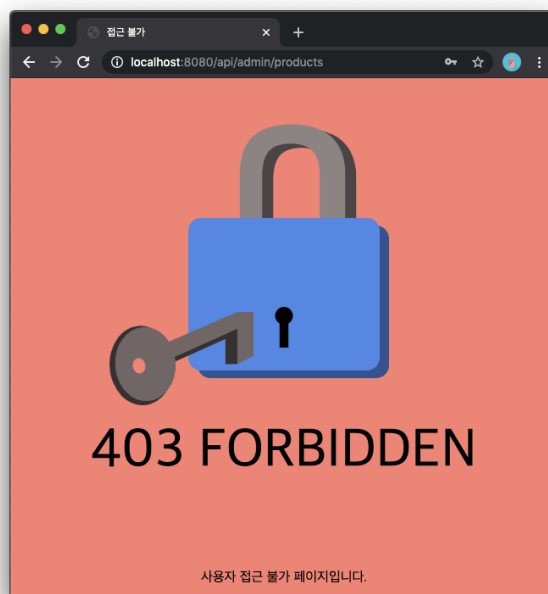
▼ [코드스니펫] 관리자용 모든 상품 조회

<http://localhost:8080/admin>

11. 접근 불가 페이지 만들기

▼ 23) 접근 불가 페이지

👉 사용자는 관리자 페이지에 인가되지 않아야 합니다!!



▼ [코드스니펫] src > main > resources > static > forbidden.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>접근 불가</title>
    <style>
      html,body{
        margin:0;
        padding:0;
        display:flex;
        justify-content:center;
        align-items:center;
        background-color:salmon;
        font-family:"Quicksand", sans-serif;
      }

      #container_anim{
        position:relative;
        width:100%;
        height:70%;
      }
    </style>
  </head>
  <body>
    <div id="container_anim">
      <div>
        <img alt="A large blue padlock icon with a keyhole." data-bbox="400 420 580 540"/>
        <h1>403 FORBIDDEN</h1>
        <p>사용자 접근 불가 페이지입니다.</p>
      </div>
    </div>
  </body>
</html>
```

```

    }

    #key{
        position:absolute;
        top:77%;
        left:-33%;
    }

    #text{
        font-size:4rem;
        position:absolute;
        top:55%;
        width:100%;
        text-align:center;
    }

    #credit{
        position:absolute;
        bottom:0;
        width:100%;
        text-align:center;
        bottom:
    }

    a{
        color: rgb(115,102,102);
    }
}
</style>
<script>
    var lock = document.querySelector('#lock');
    var key = document.querySelector('#key');

    function keyAnimate(){
        dynamics.animate(key, {
            translateX: 33
        }, {
            type:dynamics.easeInOut,
            duration:500,
            complete:lockAnimate
        })
    }

    function lockAnimate(){
        dynamics.animate(lock, {
            rotateZ:-5,
            scale:0.9
        }, {
            type:dynamics.bounce,
            duration:3000,
            complete:keyAnimate
        })
    }

    setInterval(keyAnimate, 3000);
</script>
<body>
    <div id="container_anim">
        <div id="lock" class="key-container">
            <?xml version="1.0" standalone="no"?><!-- Generator: Gravit.io --><svg xmlns="http://www.w3.org/2000/svg" x
            mlns:xlink="http://www.w3.org/1999/xlink" style="isolation:isolate" viewBox="317.286 -217 248 354" width="248" height="35
            4"><g><path d="M 354.586 -43 L 549.986 -43 C 558.43 -43 565.286 -36.144 565.286 -27.7 L 565.286 121.7 C 565.286 130.144 55
            8.43 137 549.986 137 L 354.586 137 C 346.141 137 339.286 130.144 339.286 121.7 L 339.286 -27.7 C 339.286 -36.144 346.141 -4
            3 354.586 -43 Z" style="stroke:none;fill:#2D5391;stroke-miterlimit:10;"/><g transform="matrix(-1,0,0,-1,543.786,70)"><text
            transform="matrix(1,0,0,1,0,234)" style="font-family:'Quicksand';font-weight:700;font-size:234px;font-style:normal;fill:#4
            a4444;stroke:none;">U</text></g><g transform="matrix(-1,0,0,-1,530.786,65)"><text transform="matrix(1,0,0,1,0,234)" style
            ="font-family:'Quicksand';font-weight:700;font-size:234px;font-style:normal;fill:#8e8383;stroke:none;">U</text></g><path d
            ="M 343.586 -52 L 538.986 -52 C 547.43 -52 554.286 -45.144 554.286 -36.7 L 554.286 112.7 C 554.286 121.144 547.43 128 538.9
            86 128 L 343.586 128 C 335.141 128 328.286 121.144 328.286 112.7 L 328.286 -36.7 C 328.286 -45.144 335.141 -52 343.586 -52
            Z" style="stroke:none;fill:#4A86E8;stroke-miterlimit:10;"/><g><circle vector-effect="non-scaling-stroke" cx="441.285714285
            71433" cy="63.46153846153848" r="10.461538461538453" fill="rgb(0,0,0)"/><rect x="436.055" y="66.538" width="10.462" height
            ="34.462" transform="matrix(1,0,0,1,0,0)" fill="rgb(0,0,0)"/></g></g></svg>
            </div>

            <div id="key">
                <?xml version="1.0" standalone="no"?><!-- Generator: Gravit.io --><svg xmlns="http://www.w3.org/2000/svg" x
                mlns:xlink="http://www.w3.org/1999/xlink" style="isolation:isolate" viewBox="232.612 288.821 169.348 109.179" width="169.34
                8" height="109.179"><g><path d=" M 382.96 349.821 L 368.96 349.821 L 368.96 314.821 L 382.96 307.821 L 382.96 349.821 Z " f
                ill="rgb(55,49,49)"/><path d=" M 292.134 354.827 L 379.96 315.39 L 379.96 305.547 L 292.134 343.094 L 292.134 354.827 Z " f
                ill="rgb(55,49,49)"/><path d=" M 280.96 340.109 L 401.96 288.821 L 401.96 340.109 L 382.96 349.972 L 382.96 308.547 L 265.9
                6 360.821 L 259.96 349.972 L 280.96 340.109 Z " fill="rgb(115,102,102)"/><path d=" M 401.96 288.821 L 382.96 288.821 L 280.
                96 332.821 L 292.134 340.109 L 401.96 288.821 Z " fill="rgb(115,102,102)"/><g><path d=" M 232.755 354.125 C 230.958 328.501
                246.297 306.519 266.988 305.068 C 287.679 303.617 305.937 323.243 307.734 348.867 C 309.531 374.492 294.191 396.473 273.5 3
                97.924 C 252.809 399.375 234.552 379.75 232.755 354.125 Z " fill="rgb(55,49,49)"/><path d=" M 239.241 352.316 C 237.564 32

```

```

8.406 252.144 307.876 271.779 306.499 C 291.414 305.122 308.716 323.416 310.393 347.326 C 312.07 371.236 297.49 391.766 27
7.855 393.143 C 258.22 394.52 240.917 376.226 239.241 352.316 Z " fill="rgb(115,102,102)"/><path d=" M 260.038 353.084 C 25
9.196 348.171 261.788 343.621 265.822 342.929 C 269.856 342.238 273.816 345.665 274.658 350.578 C 275.5 355.49 272.909 360.
041 268.874 360.732 C 264.84 361.424 260.88 357.997 260.038 353.084 Z " fill="salmon"/></g></svg>
</div>
</div>

<p id="text">403 FORBIDDEN</p>
<p id="credit">사용자 접근 불가 페이지입니다.</a></p>
</body>
</html>

```

👉 스프링 시큐리티 설정을 이용해 '관리자'만 접속 가능한 페이지를 설정해 봅시다.

1. 스프링 시큐리티가 로그인한 회원의 권한을 인식하도록 수정

▼ [코드스니펫] src > main > com.sparta.springcore.security > UserDetailsImpl

```

private static final String ROLE_PREFIX = "ROLE_";

@Override
public Collection<? extends GrantedAuthority> getAuthorities() {
    UserRole userRole = user.getRole();

    SimpleGrantedAuthority authority = new SimpleGrantedAuthority(ROLE_PREFIX + userRole.toString());
    Collection<GrantedAuthority> authorities = new ArrayList<>();
    authorities.add(authority);

    return authorities;
}

```

2. WebSecurityConfig 파일 수정

• "@EnableGlobalMethodSecurity(securedEnabled = true)" 추가

▼ [코드스니펫] src > main > com.sparta.springcore.security > WebSecurityConfig

```

package com.sparta.springcore.security;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodSecurity;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

@Configuration
@EnableWebSecurity // 스프링 Security 지원을 가능하게 함
@EnableGlobalMethodSecurity(securedEnabled = true)
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

    @Bean
    public BCryptPasswordEncoder encodePassword() {
        return new BCryptPasswordEncoder();
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.csrf().disable();
        http.headers().frameOptions().disable();
        http.authorizeRequests()
            // image 폴더를 login 없이 허용
            .antMatchers("/images/**").permitAll()
            // css 폴더를 login 없이 허용
            .antMatchers("/css/**").permitAll()
            .antMatchers("/user/**").permitAll()
            .antMatchers("/h2-console/**").permitAll()
            // 그 외 모든 요청은 인증과정 필요
            .anyRequest().authenticated()
            .and()
            .formLogin()
            .loginPage("/user/login")
            .loginProcessingUrl("/user/login")
            .defaultSuccessUrl("/")
            .permitAll()
            .and()
            .logout()

```

```

        .logoutUrl("/user/logout")
        .permitAll()
        .and()
        .exceptionHandling()
        .accessDeniedPage("/user/forbidden");
    }
}

```

3. 컨트롤러에서 인가가 필요한 API 에 **@Secured** 어노테이션과 "ROLE 이름" 추가

- **@Secured("{ROLE 이름}")**

▼ [코드스니펫] src > main > com.sparta.springcore.controller > HomeController

```

@Secured("ROLE_ADMIN")
@GetMapping("/admin")

```

▼ [코드스니펫] src > main > com.sparta.springcore.controller > ProductController

```

// (관리자용) 등록된 모든 상품 목록 조회
@Secured("ROLE_ADMIN")
@GetMapping("/api/admin/products")

```

4. forbidden 페이지를 위한 UserController 설정

▼ [코드스니펫] src > main > com.sparta.springcore.controller > UserController

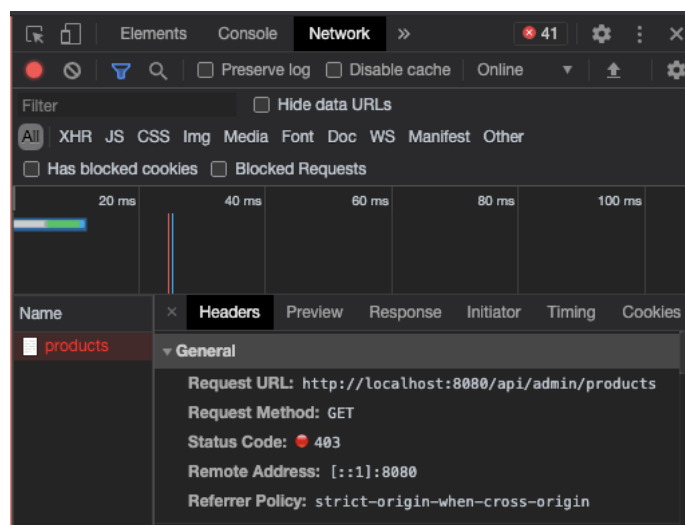
```

@GetMapping("/user/forbidden")
public String forbidden() {
    return "forbidden";
}

```

👉 이제 해당 API ("GET /admin, GET /api/admin/products") 접근 시, 스프링 시큐리티가 로그인 시도하는 회원이 관리자 역할을 가지고 있는지 인가합니다. 만약 관리자 권한을 가지고 있지 않은 회원의 요청이라면, 스프링 시큐리티 설정에 따라 접근 금지 페이지를 내려줍니다.

👉 서버에서는 접근 권한에 대한 응답을 "HTTP Status Code 403 (Forbidden)" 으로 정의해두고 사용합니다. 다음은 크롬 브라우저에서 HTTP 응답으로 403 을 받는 경우의 캡처화면입니다.



12. 소셜 로그인

▼ 21) 소셜 로그인 탄생 배경

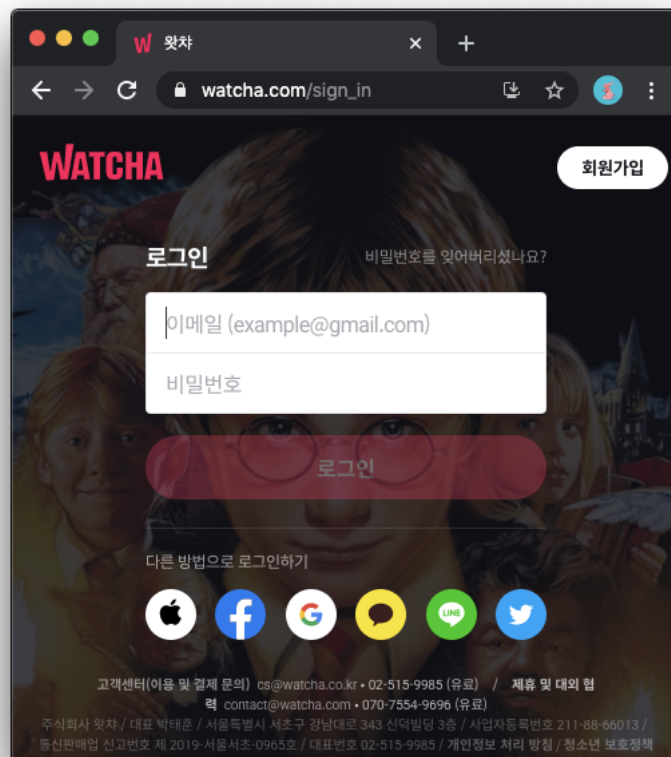


모든 웹 사이트에서 회원가입 과정을 거치는 것은 사용자에게 부담이 됩니다.

매번 번거로운 회원가입 과정을 수행해야 할 뿐 아니라, 웹 사이트마다 다른 아이디와 비밀번호를 기억해야 합니다.

또한 웹 사이트를 운영하는 측에서도 회원들의 개인정보를 지켜야하는 역할이 부담이 됩니다. 바이러스와 백신의 관계 처럼, 발전하는 해킹 기술을 막기 위해 보안을 강화하는 노력이 지속적으로 필요하기 때문이죠.

이런 문제를 해결하기 위해 OAuth 를 사용한 소셜 로그인이 등장합니다.



출처: Watcha 로그인

▼ 22) OAuth 란?



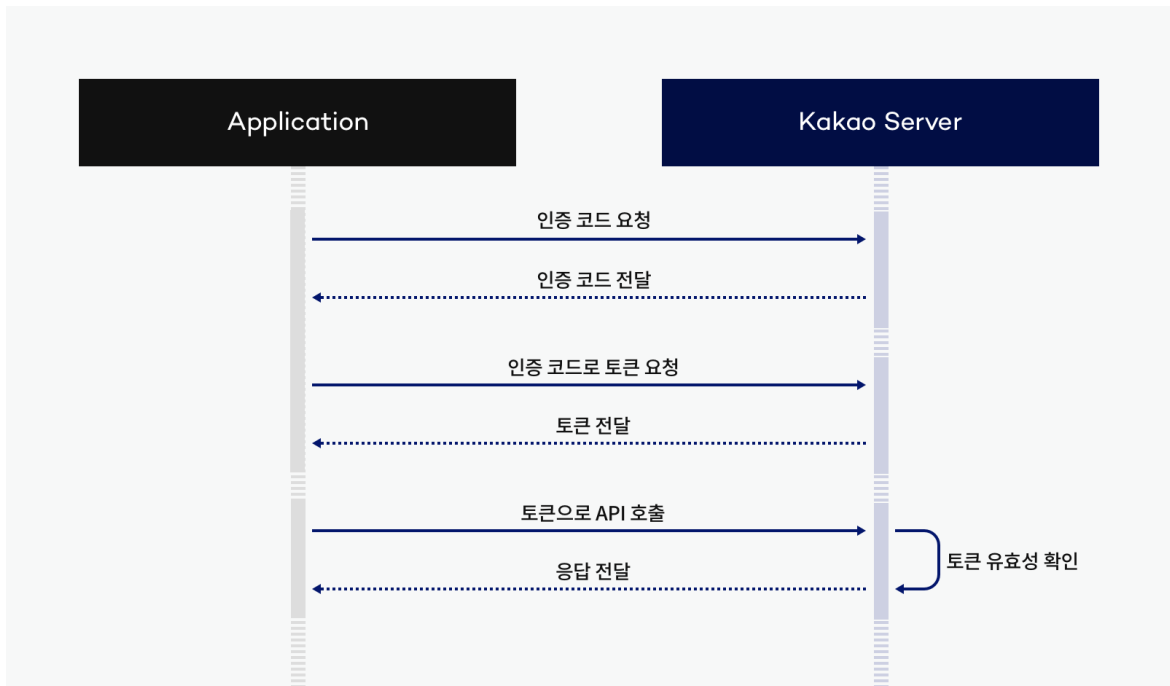
OAuth는 인터넷 사용자들이 비밀번호를 제공하지 않고 다른 웹사이트 상의 자신들의 정보에 대해 웹사이트나 애플리케이션의 접근 권한을 부여할 수 있는 공통적인 수단으로서 사용되는, 접근 위임을 위한 개방형 표준입니다. 사용자가 애플리케이션에게 모든 권한을 넘기지 않고 사용자 대신 서비스를 이용할 수 있게 해주는 HTTP 기반의 보안 프로토콜입니다. OAuth를 사용하는 서비스 제공자는 대표적으로 구글, 페이스북 등이 있습니다. 국내에는 대표적으로 네이버와 카카오가 있죠.

- 우리는 이 중 카카오 로그인을 실습 해 볼 예정입니다. 다음과 같은 순서로 진행합니다.

13. 카카오 로그인 설정하기

▼ 23) 카카오 로그인 설정하기

☞ 아래는 카카오 로그인의 큰 흐름을 설명해 주는 그림입니다.



☞ 일단 카카오 로그인을 사용하기 위해서는 카카오 개발 사이트에서 '애플리케이션 등록'이 필요합니다. 그럼 '카카오 개발자' 사이트로 이동하여, 회원가입을 진행하고, 본인만의 애플리케이션을 등록 해 봅니다.

▼ [코드스니펫] 카카오디벨로퍼스 사이트

<https://developers.kakao.com/console/app>

1. 회원가입

The screenshot shows the '회원가입' (Sign Up) page on the Kakao Developers console. The page has a dark blue header with the 'kakao developers' logo and navigation links: '내 애플리케이션', '제품', '문서', '도구', '포럼', '로그인', and 'KOR ENG'. The main content area has a dark blue background with the title '회원가입' and a subtitle '카카오디벨로퍼스의 모든 기능을 사용하기 위해 서비스 이용 약관에 동의해주세요.' (Please agree to the service terms of use to use all features of Kakao Developers). Below this, there are two sections for terms of service: '서비스 이용 동의' (Service Use Agreement) and '개인정보 수집 및 이용에 대한 동의' (Agreement on Collection and Use of Personal Information). The '서비스 이용 동의' section is expanded, showing a list of terms including the requirement to agree to the terms of use, the prohibition of unauthorized use, and the requirement to provide accurate information. The '개인정보 수집 및 이용에 대한 동의' section is also expanded, showing the requirement to agree to the terms of use for personal information collection and use.

2. 내 애플리케이션 메뉴 선택 > 애플리케이션 추가하기

kakao developers

내 애플리케이션

제품

문서

도구

포럼

lhe1997@naver.com

KOR ENG

내 애플리케이션

전체 애플리케이션 (0)

애플리케이션 이름

+

애플리케이션 추가하기

3. 앱 아이콘, 앱 이름, 사업자명 저장

애플리케이션 추가하기

앱 아이콘



파일 선택

JPG, GIF, PNG
권장 사이즈 128px, 최대 250KB

앱 이름

나만의 셀렉샵

사업자명

스파르타코딩클럽

• 입력된 정보는 사용자가 카카오 로그인을 할 때 표시됩니다.

• 정보가 정확하지 않은 경우 서비스 이용이 제한될 수 있습니다.

취소

저장

▼ [코드스니펫] 스프링 르탄이 아이콘


https://s3.ap-northeast-2.amazonaws.com/materials.spartacodingclub.kr/spring_plus/%E1%84%B9%E1%85%B3%E1%84%91%E1%85%B3%E1

<https://s3-us-west-2.amazonaws.com/secure.notion-static.com/3f9b9cd1-7a97-4394-94e0-73c24e8bdc92.png>

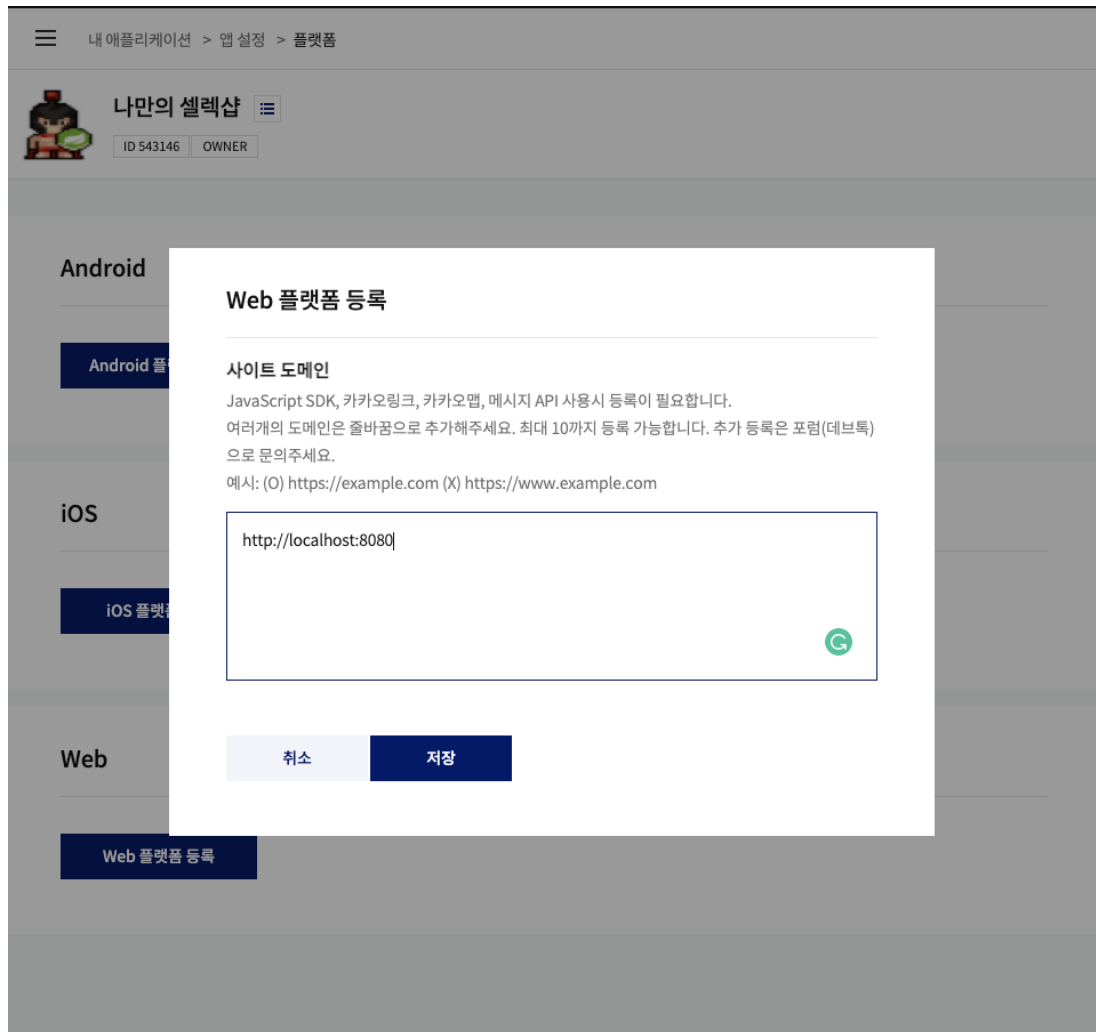
4. 사이트 도메인 등록하기

1. 애플리케이션 선택
2. 플랫폼 메뉴 선택

3. Web 플랫폼 등록
4. 사이트 도메인 입력

 사이트 도메인에는 개발중인 로컬환경의 서버 주소를 입력해 줍니다.

`http://localhost:8080`

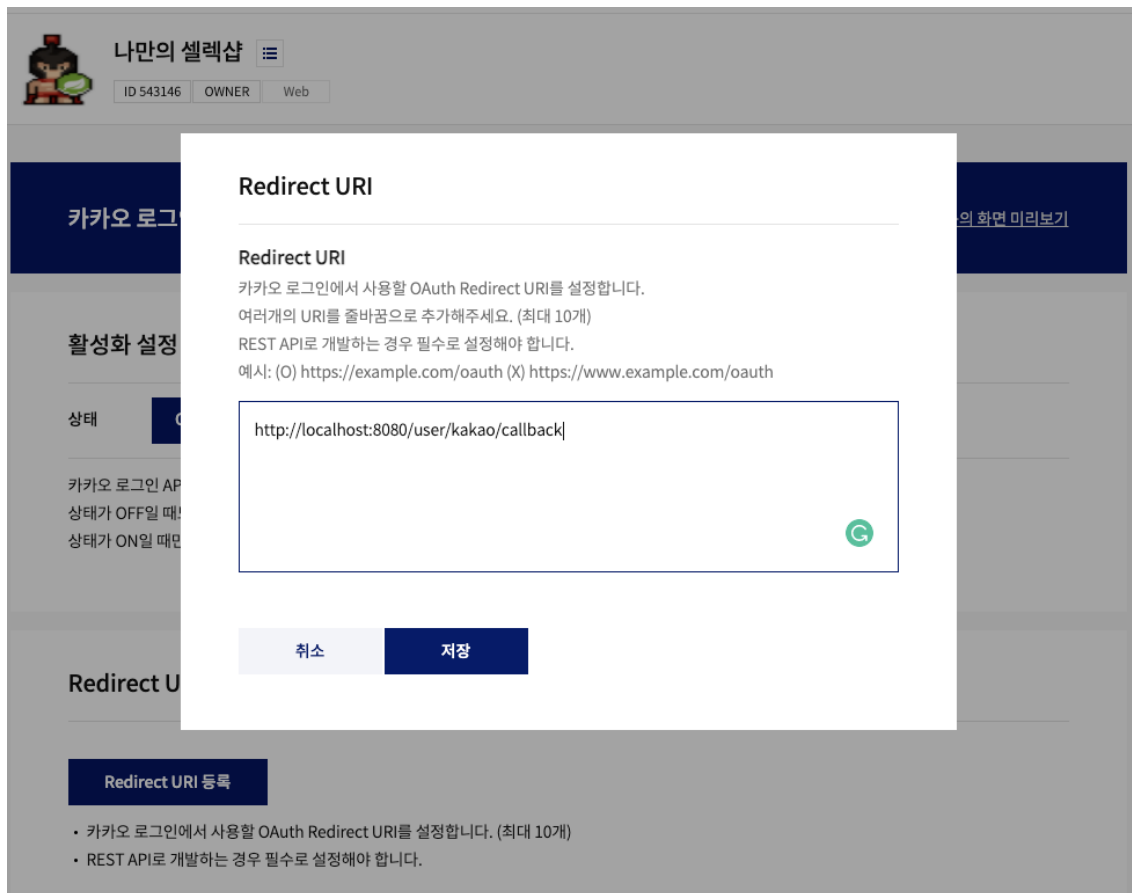


5. 카카오로 로그인 했을 때 인가토큰을 받게 될 Redirect URI (callback) 를 설정하기

Web		삭제	수정
사이트 도메인	http://localhost:8080		

- 카카오 로그인 사용 시 Redirect URI를 등록해야 합니다. [등록하러 가기](#)

`http://localhost:8080/user/kakao/callback`



6. 동의항목 설정하기

- 카카오 서버로부터 사용자의 어떤 정보를 받을지 정할 수 있습니다.

kakao developers

내 애플리케이션

내 애플리케이션 > 제품 설정 > 카카오 로그인 > 동의항목

일반

비즈니스

앱 키

플랫폼

팀 관리

제품 설정

카카오 로그인

동의항목

간편가입

카카오톡 채널

개인정보 국외이전

연결 끊기

사용자 프로퍼티

보안

고급

카카오링크

카카오톡 채널

음성

푸시 알림

카카오 로그인 ON

동의 화면 미리보기

동의항목

카카오 로그인으로 서비스를 시작할 때 동의 받는 항목을 설정합니다. 미리 보기를 통해 사용자에게 보여질 화면을 확인할 수 있습니다. 사업자 정보를 등록하여 비즈 앱으로 전환하고 비즈니스 채널을 연결하면 권한이 없는 동의 항목에 대한 검수 신청을 할 수 있습니다.

비즈니스 설정 바로가기

개인정보 보호

항목 이름	ID	상태
프로필 정보(닉네임/프로필 사진)	profile	필수 동의 <div>설정</div>
카카오톡 채널 추가 상태 및 내역	plusfriends	권한 없음
카카오계정(이메일)	account_email	선택 동의 (수집) <div>설정</div>
성별	gender	사용 안함 <div>설정</div>
연령대	age_range	사용 안함 <div>설정</div>
카카오 서비스내 친구목록(즐거찾기 친구포함)	friends	사용 안함 <div>설정</div>

1. 프로필 정보를 '필수 동의'로 받습니다.

[스파르타코딩클럽] Spring 심화반 - 2주차

45

동의 항목 설정

항목

프로필 정보(닉네임/프로필 사진) / profile

동의 단계

☒ 필수 동의
카카오 로그인 시 사용자가 필수로 동의해야 합니다.

☐ 선택 동의
사용자가 동의하지 않아도 카카오 로그인을 완료할 수 있습니다.

☐ 이용 중 동의
카카오 로그인 시 동의를 받지 않고, 항목이 필요한 시점에 동의를 받습니다.

☐ 사용 안함
사용자에게 동의를 요청하지 않습니다.

카카오 계정으로 정보 수집 후 제공

☐ 사용자에게 값이 없는 경우 카카오 계정 정보 입력을 요청하여 수집

동의 목적 [\[필수\]](#)

회원가입

개발자 앱 동의 항목 관리 화면에 입력하는 사실이 실제 서비스 내용과 다를 경우 API 서비스의 거부 사유가 될 수 있습니다.

취소

저장

2. 카카오회정(이메일) 정보를 '선택 동의'로 받습니다.

동의 항목 설정

항목

카카오회정(이메일) / account_email

동의 단계

☐ 필수 동의 [\[필수 필요\]](#)
카카오 로그인 시 사용자가 필수로 동의해야 합니다.

☒ 선택 동의
사용자가 동의하지 않아도 카카오 로그인을 완료할 수 있습니다.

☐ 이용 중 동의
카카오 로그인 시 동의를 받지 않고, 항목이 필요한 시점에 동의를 받습니다.

☐ 사용 안함
사용자에게 동의를 요청하지 않습니다.

카카오 계정으로 정보 수집 후 제공

☐ 사용자에게 값이 없는 경우 카카오 계정 정보 입력을 요청하여 수집

동의 목적 [\[필수\]](#)

회원가입

개발자 앱 동의 항목 관리 화면에 입력하는 사실이 실제 서비스 내용과 다를 경우 API 서비스의 거부 사유가 될 수 있습니다.

취소

저장

자! 이제 카카오 서버를 사용할 준비가 끝났으니 우리 사이트와의 연동을 해 보겠습니다.

14. 카카오 로그인 구현하기

▼ 24) 인가코드 받기

- 카카오 디벨로퍼스 사이트에서 REST API 키 확인

▼ [코드스니펫] 카카오 디벨로퍼스 사이트

<https://developers.kakao.com/console/app>

앱 키

네이티브 앱 키	7fc61630bb5bacf467fc9d585efdd61f
REST API 키	201392c25344311dcaa4f0ded2b34928
JavaScript 키	35e539218ffdb3fc3dbb386098f662ff
Admin 키	f87bb202c0651d1e1bbf73c142957239

• 카카오 로그인 요청 URI

- login.html 파일에서 변경

```
https://auth.kakao.com/oauth/authorize?client_id={본인의 REST API키}&redirect_uri=http://localhost:8080/user/kakao/callback&response_type=code
```

• 카카오 로그인

- "카카오계정(이메일)" 을 꼭 동의!

kakao

나만의 셀렉샵
개발자

✓ **전체 동의하기**

전체동의를 선택목적에 대한 동의를 포함하고 있으며, 선택목적에 대한 동의를 거부해도 서비스 이용이 가능합니다.

lthysk@hotmail.com
 [계정 변경](#)

나만의 셀렉샵 서비스 제공을 위해 회원번호와 함께 개인정보를 제공합니다. 보다 자세한 개인정보 제공항목은 동의 내용에서 확인하실 수 있습니다. 정보는 서비스 탈퇴 시 지체없이 파기됩니다.

✓ **[필수] 필수 제공 항목** [보기](#)

프로필 정보(닉네임/프로필 사진)

[선택] 선택 제공 항목 [보기](#)

✓ 카카오계정(이메일)

동의하고 계속하기



사용자가 카카오 로그인 페이지를 통해 '동의하고 계속하기'를 클릭하면, 미리 설정해둔 Redirect URI (callback) 로 인가토큰이 전달 됩니다.

▼ [코드스니펫] src > main > com.sparta.springcore.controller > UserController

```
@GetMapping("/user/kakao/callback")
public String kakaoLogin(String code) {
    // authorizedCode: 카카오 서버로부터 받은 인가 코드
    userService.kakaoLogin(code);

    return "redirect:/";
}
```

▼ 25) 카카오로그인 처리부터 구현

▼ [코드스니펫] src > main > com.sparta.springcore > model > User

```
package com.sparta.springcore.model;

import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

import javax.persistence.*;

@Setter
@Getter // get 함수를 일괄적으로 만들어줍니다.
@NoArgsConstructor // 기본 생성자를 만들어줍니다.
@Entity // DB 테이블 역할을 합니다.
public class User extends Timestamped {

    public User(String username, String password, String email, UserRole role) {
        this.username = username;
        this.password = password;
        this.email = email;
        this.role = role;
        this.kakaoId = null;
    }

    public User(String username, String password, String email, UserRole role, Long kakaoId) {
        this.username = username;
        this.password = password;
        this.email = email;
        this.role = role;
        this.kakaoId = kakaoId;
    }

    // ID가 자동으로 생성 및 증가합니다.
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Id
    private Long id;

    // 반드시 값을 가지도록 합니다.
    @Column(nullable = false)
    private String username;

    @Column(nullable = false)
    private String password;

    @Column(nullable = false)
    private String email;

    @Column(nullable = false)
    @Enumerated(value = EnumType.STRING)
    private UserRole role;

    @Column(nullable = true)
    private Long kakaoId;
}
```

▼ [코드스니펫] src > main > com.sparta.springcore > service > UserService

```
package com.sparta.springcore.service;

import com.sparta.springcore.dto.SignupRequestDto;
import com.sparta.springcore.model.User;
import com.sparta.springcore.model.UserRole;
import com.sparta.springcore.repository.UserRepository;
import com.sparta.springcore.security.kakao.KakaoOAuth2;
import com.sparta.springcore.security.kakao.KakaoUserInfo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;

import java.util.Optional;

@Service
public class UserService {
    private final PasswordEncoder passwordEncoder;
```



```

private final UserRepository userRepository;
private final KakaoOAuth2 kakaoOAuth2;
private final AuthenticationManager authenticationManager;
private static final String ADMIN_TOKEN = "AAABnv/xRVklrnYxKZ0aHgTBcXukeZygoC";

@Autowired
public UserService(UserRepository userRepository, PasswordEncoder passwordEncoder, KakaoOAuth2 kakaoOAuth2, AuthenticationManager authenticationManager) {
    this.userRepository = userRepository;
    this.passwordEncoder = passwordEncoder;
    this.kakaoOAuth2 = kakaoOAuth2;
    this.authenticationManager = authenticationManager;
}

public void registerUser(SignupRequestDto requestDto) {
    String username = requestDto.getUsername();
    // 회원 ID 중복 확인
    Optional<User> found = userRepository.findByUsername(username);
    if (found.isPresent()) {
        throw new IllegalArgumentException("중복된 사용자 ID 가 존재합니다.");
    }

    // 비밀번호 인코딩
    String password = passwordEncoder.encode(requestDto.getPassword());
    String email = requestDto.getEmail();
    // 사용자 ROLE 확인
    UserRole role = UserRole.USER;
    if (requestDto.isAdmin()) {
        if (!requestDto.getAdminToken().equals(ADMIN_TOKEN)) {
            throw new IllegalArgumentException("관리자 암호가 틀려 등록이 불가능합니다.");
        }
        role = UserRole.ADMIN;
    }

    User user = new User(username, password, email, role);
    userRepository.save(user);
}

public void kakaoLogin(String authorizedCode) {
    // 카카오 OAuth2 를 통해 카카오 사용자 정보 조회
    KakaoUserInfo userInfo = kakaoOAuth2.getUserInfo(authorizedCode);
    Long kakaoId = userInfo.getId();
    String nickname = userInfo.getNickname();
    String email = userInfo.getEmail();

    // 우리 DB 에서 회원 Id 와 비밀번호
    // 회원 Id = 카카오 nickname
    String username = nickname;
    // 비밀번호 = 카카오 Id + ADMIN TOKEN
    String password = kakaoId + ADMIN_TOKEN;

    // DB 에 중복된 Kakao Id 가 있는지 확인
    User kakaoUser = userRepository.findByKakaoId(kakaoId)
        .orElse(null);

    // 카카오 정보로 회원가입
    if (kakaoUser == null) {
        // 비밀번호 인코딩
        String encodedPassword = passwordEncoder.encode(password);
        // ROLE = 사용자
        UserRole role = UserRole.USER;

        kakaoUser = new User(nickname, encodedPassword, email, role, kakaoId);
        userRepository.save(kakaoUser);
    }

    // 로그인 처리
    Authentication kakaoUsernamePassword = new UsernamePasswordAuthenticationToken(username, password);
    Authentication authentication = authenticationManager.authenticate(kakaoUsernamePassword);
    SecurityContextHolder.getContext().setAuthentication(authentication);
}
}

```

▼ [코드스니펫] src > main > com.sparta.springcore > repository > UserRepository

```

package com.sparta.springcore.repository;

import com.sparta.springcore.model.User;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.Optional;

```

```
public interface UserRepository extends JpaRepository<User, Long> {
    Optional<User> findByUsername(String username);
    Optional<User> findByKakaoId(Long kakaoId);
}
```

▼ [코드스니펫] src > main > com.sparta.springcore > security > WebSecurityConfig

```
@Bean
@Override
public AuthenticationManager authenticationManagerBean() throws Exception {
    return super.authenticationManagerBean();
}
```

▼ [코드스니펫] src > main > com.sparta.springcore > security > kakao > KakaoUserInfo

```
package com.sparta.springcore.security.kakao;

import lombok.AllArgsConstructor;
import lombok.Getter;

@AllArgsConstructor
@Getter
public class KakaoUserInfo {
    Long id;
    String email;
    String nickname;
}
```

▼ [코드스니펫] src > main > com.sparta.springcore > security > kakao > KakaoOAuth2

```
package com.sparta.springcore.security.kakao;

import org.json.JSONObject;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpMethod;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Component;
import org.springframework.util.LinkedMultiValueMap;
import org.springframework.util.MultiValueMap;
import org.springframework.web.client.RestTemplate;

@Component
public class KakaoOAuth2 {
    public KakaoUserInfo getUserInfo(String authorizedCode) {
        // 1. 인가코드 -> 액세스 토큰
        String accessToken = getAccessToken(authorizedCode);
        // 2. 액세스 토큰 -> 카카오 사용자 정보
        KakaoUserInfo userInfo = getUserInfoByToken(accessToken);

        return userInfo;
    }

    private String getAccessToken(String authorizedCode) {
        // HttpHeaders 오브젝트 생성
        HttpHeaders headers = new HttpHeaders();
        headers.add("Content-type", "application/x-www-form-urlencoded;charset=utf-8");

        // HttpBody 오브젝트 생성
        MultiValueMap<String, String> params = new LinkedMultiValueMap<>();
        params.add("grant_type", "authorization_code");
        params.add("client_id", "{본인의 REST API키}");
        params.add("redirect_uri", "http://localhost:8080/user/kakao/callback");
        params.add("code", authorizedCode);

        // HttpHeaders와 HttpBody를 하나의 오브젝트에 담기
        RestTemplate rt = new RestTemplate();
        HttpEntity<MultiValueMap<String, String>> kakaoTokenRequest =
            new HttpEntity<>(params, headers);

        // Http 요청하기 - Post방식으로 - 그리고 response 변수의 응답 받음.
        ResponseEntity<String> response = rt.exchange(
            "https://kauth.kakao.com/oauth/token",
            HttpMethod.POST,
            kakaoTokenRequest,
            String.class
        );
    }
}
```

```

// JSON -> 액세스 토큰 파싱
String tokenJson = response.getBody();
JSONObject rjson = new JSONObject(tokenJson);
String accessToken = rjson.getString("access_token");

return accessToken;
}

private KakaoUserInfo getUserInfoByToken(String accessToken) {
    // HttpHeaders 오브젝트 생성
    HttpHeaders headers = new HttpHeaders();
    headers.add("Authorization", "Bearer " + accessToken);
    headers.add("Content-type", "application/x-www-form-urlencoded;charset=utf-8");

    // HttpHeaders와 HttpBody를 하나의 오브젝트에 담기
    RestTemplate rt = new RestTemplate();
    HttpEntity<MultiValueMap<String, String>> kakaoProfileRequest = new HttpEntity<>(headers);

    // Http 요청하기 - Post방식으로 - 그리고 response 변수의 응답 받음.
    ResponseEntity<String> response = rt.exchange(
        "https://kapi.kakao.com/v2/user/me",
        HttpMethod.POST,
        kakaoProfileRequest,
        String.class
    );

    JSONObject body = new JSONObject(response.getBody());
    Long id = body.getLong("id");
    String email = body.getJSONObject("kakao_account").getString("email");
    String nickname = body.getJSONObject("properties").getString("nickname");

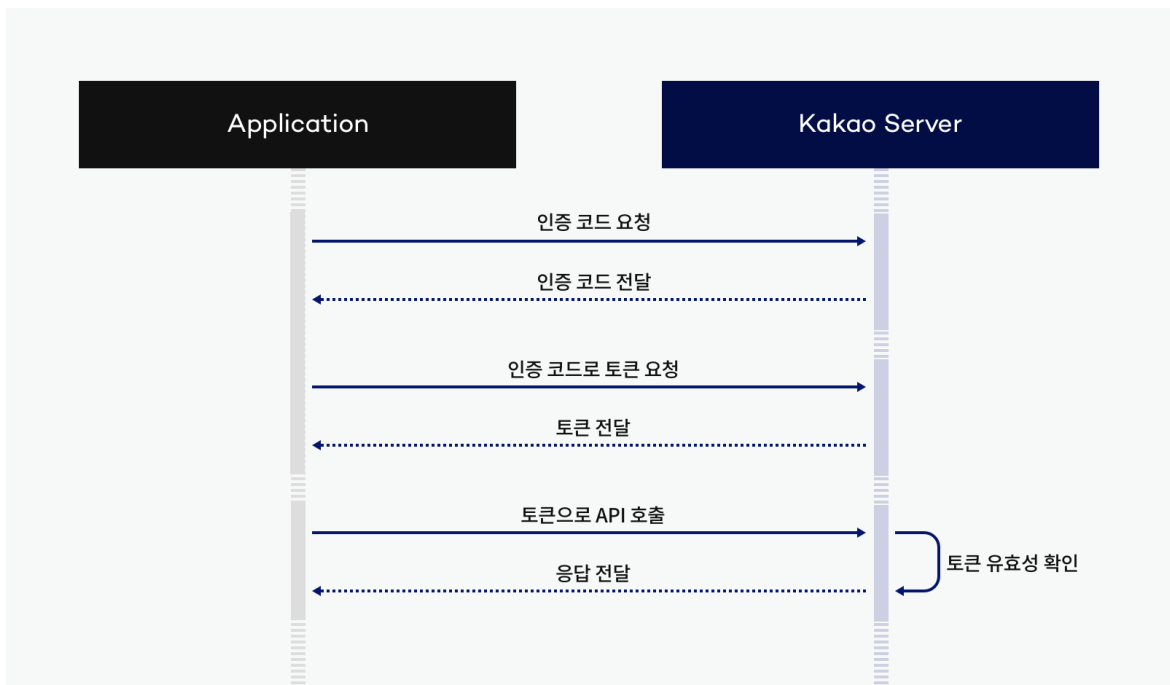
    return new KakaoUserInfo(id, email, nickname);
}
}

```

- **KakaoOAuth2** 에 발급 받은 본인의 REST API 키 입력

```
params.add("client_id", "{본인의 REST API키}");
```

▼ 26) 액세스 토큰을 받아, 카카오 사용자 정보 가져오기



- 카카오 사용자 정보 JSON 의 예

```

{
  "id": 1632335751,
  "properties": {


```

```

    "nickname": "르탄이",
    "profile_image": "http://k.kakaocdn.net/...jpg",
    "thumbnail_image": "http://k.kakaocdn.net/...jpg"
  },
  "kakao_account": {
    "profile_needs_agreement": false,
    "profile": {
      "nickname": "르탄이",
      "thumbnail_image_url": "http://k.kakaocdn.net/...jpg",
      "profile_image_url": "http://k.kakaocdn.net/...jpg"
    },
    "has_email": true,
    "email_needs_agreement": false,
    "is_email_valid": true,
    "is_email_verified": true,
    "email": "letan@spartacondingclub.com"
  }
}

```

▼ 27) 카카오 사용자 정보를 이용해 회원가입

 카카오서버에서 받은 사용자 정보를 이용해 회원 가입을 합니다.

• src > main > com.sparta.springcore.service > UserService

```

// 우리 DB 에서 회원 Id 와 패스워드
// 회원 Id = 카카오 nickname
String username = nickname;
// 패스워드 = 카카오 Id + ADMIN TOKEN
String password = kakaoId + ADMIN_TOKEN;


// DB 에 중복된 Kakao Id 가 있는지 확인
User kakaoUser = userRepository.findByKakaoId(kakaoId)
    .orElse(null);

if (kakaoUser == null) {
    // 패스워드 인코딩
    String encodedPassword = passwordEncoder.encode(password);
    // ROLE = 사용자
    UserRole role = UserRole.USER;

    kakaoUser = new User(nickname, encodedPassword, email, role, kakaoId);
    userRepository.save(kakaoUser);
}

```

▼ 28) 로그인 처리

 그리고 스프링 시큐리티를 이용해 로그인 처리를 해 줍니다.

• src > main > com.sparta.springcore.service > UserService

```

// 로그인 처리
Authentication kakaoUsernamePassword = new UsernamePasswordAuthenticationToken(username, password);
Authentication authentication = authenticationManager.authenticate(kakaoUsernamePassword);
SecurityContextHolder.getContext().setAuthentication(authentication);

```

15. 2주차 끝 & 숙제 설명

- ☺ 수고하셨습니다! 2주차 회원가입&로그인 기능을 모두 구현하셨습니다!
- 만약에 이미 회원가입한 사용자가 동일한 이메일로 카카오 로그인을 시도하면 어떻게 되어야 할까요?

▼ 숙제 설명

카카오 로그인 시 받게 되는 "카카오 사용자 정보"

1. 카카오 Id

2. 카카오 Nickname
3. 카카오 Email

현재) 카카오 로그인 처리 방법

1. '나만의 셀렉샵' 회원 중 "카카오 Id" 를 가진 회원 조회
 1. O → 회원 로그인
 2. X → 회원 가입 → 회원 로그인

숙제) 카카오 로그인 처리방법 수정 및 검증

1. '나만의 셀렉샵' 회원 중 "카카오 Id" 를 가진 회원 조회
 1. O → 회원 로그인
 2. X → **동일 Email 을 가진 기존 회원 존재?**
 1. O → 동일 Email 가진 회원에 "카카오 Id" 추가 → 회원 로그인
 2. X → 회원 가입 → 회원 로그인
2. 검증방법
 1. 카카오 email 과 동일한 email 을 가진 회원 가입
 2. Id, PW 로 회원 로그인
 3. 관심상품 1~2개 등록
 4. 로그아웃
 5. 카카오톡 로그인 시 2번에서 등록한 관심상품 확인
 6. h2 console 에서 USER 테이블에서 kakaoId 가 추가되었는지도 확인!!

Tip) 참고 코드

1. "카카오 Id" 를 가진 회원 조회

▼ [코드스니펫] 카카오 Id 사용자 조회

```
User kakaoUser = userRepository.findByKakaoId(kakaoId)
    .orElse(null);
```

```
User kakaoUser = userRepository.findByKakaoId(kakaoId)
    .orElse(null);
```

2. 로그인 처리 방법

▼ [코드스니펫] 로그인 처리

```
// 스프링 시큐리티 통해 인증된 사용자로 등록
UserDetailsImpl userDetails = new UserDetailsImpl(kakaoUser);
Authentication authentication = new UsernamePasswordAuthenticationToken(userDetails, null, userDetails.getAuthorities());
SecurityContextHolder.getContext().setAuthentication(authentication);
```

```
// 스프링 시큐리티 통해 로그인 처리
UserDetailsImpl userDetails = new UserDetailsImpl(kakaoUser);
Authentication authentication = new UsernamePasswordAuthenticationToken(userDetails, null, userDetails.getAuthorities());
SecurityContextHolder.getContext().setAuthentication(authentication);
```

3. 이메일 주소로 회원 정보 조회방법

▼ [코드스니펫] 이메일 정보 동일 여부 조회

```
public interface UserRepository extends JpaRepository<User, Long> {
    Optional<User> findByUsername(String username);
    Optional<User> findByKakaoId(Long kakaoId);
    // 추가! Email 주소가 같은 사용자 조회
    Optional<User> findByEmail(String email);
}
```

```
public interface UserRepository extends JpaRepository<User, Long> {
    Optional<User> findByUsername(String username);
    Optional<User> findByKakaoId(Long kakaoId);
    // 추가! Email 주소가 같은 사용자 조회
    Optional<User> findByEmail(String email);
}
```

4. 기존 사용자에게 "kakao id" 추가 방법

▼ [코드스니펫] 기존 사용자에게 Kakao Id 추가

```
existUser.setKakaoId(kakaoId);
userRepository.save(existUser);
```

```
existUser.setKakaoId(kakaoId);
userRepository.save(existUser);
```

16. 2주차 숙제 답안 코드

▼ [코드스니펫] - 2주차 숙제 답안 코드

전체 코드

▼ src > main > java > com.sparta.springcore > service > UserService 클래스

```
package com.sparta.springcore.service;

import com.sparta.springcore.dto.SignupRequestDto;
import com.sparta.springcore.model.User;
import com.sparta.springcore.model.UserRole;
import com.sparta.springcore.repository.UserRepository;
import com.sparta.springcore.security.UserDetailsImpl;
import com.sparta.springcore.security.kakao.KakaoOAuth2;
import com.sparta.springcore.security.kakao.KakaoUserInfo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;

import java.util.Optional;

@Service
public class UserService {
    private final PasswordEncoder passwordEncoder;
    private final UserRepository userRepository;
    private final KakaoOAuth2 kakaoOAuth2;
    private static final String ADMIN_TOKEN = "AAABnv/xRVklrnYxKZ0aHgTBcXukeZygoC";

    @Autowired
    public UserService(UserRepository userRepository, PasswordEncoder passwordEncoder, KakaoOAuth2 kakaoOAuth2, AuthenticationManager authenticationManager) {
        this.userRepository = userRepository;
        this.passwordEncoder = passwordEncoder;
        this.kakaoOAuth2 = kakaoOAuth2;
    }

    public void registerUser(SignupRequestDto requestDto) {
        String username = requestDto.getUsername();
        // 회원 ID 중복 확인
    }
}
```

```

Optional<User> found = userRepository.findByUsername(username);
if (found.isPresent()) {
    throw new IllegalArgumentException("중복된 사용자 ID 가 존재합니다.");
}

// 패스워드 인코딩
String password = passwordEncoder.encode(requestDto.getPassword());
String email = requestDto.getEmail();
// 사용자 ROLE 확인
UserRole role = UserRole.USER;
if (requestDto.isAdmin()) {
    if (!requestDto.getAdminToken().equals(ADMIN_TOKEN)) {
        throw new IllegalArgumentException("관리자 암호가 틀려 등록이 불가능합니다.");
    }
    role = UserRole.ADMIN;
}

User user = new User(username, password, email, role);
userRepository.save(user);
}

public void kakaoLogin(String authorizedCode) {
    // 카카오 OAuth2 를 통해 카카오 사용자 정보 조회
    KakaoUserInfo userInfo = kakaoOAuth2.getUserInfo(authorizedCode);
    Long kakaoId = userInfo.getId();
    String nickname = userInfo.getNickname();
    String email = userInfo.getEmail();

    // DB 에 중복된 Kakao Id 가 있는지 확인
    User kakaoUser = userRepository.findByKakaoId(kakaoId)
        .orElse(null);

    if (kakaoUser == null) {
        // 카카오 이메일과 동일한 이메일을 가진 회원이 있는지 확인
        User sameEmailUser = userRepository.findByEmail(email).orElse(null);
        if (sameEmailUser != null) {
            kakaoUser = sameEmailUser;
            // 카카오 이메일과 동일한 이메일 회원이 있는 경우
            // 카카오 Id 를 회원정보에 저장
            kakaoUser.setKakaoId(kakaoId);
            userRepository.save(kakaoUser);
        } else {
            // 카카오 정보로 회원가입
            // username = 카카오 nickname
            String username = nickname;
            // password = 카카오 Id + ADMIN TOKEN
            String password = kakaoId + ADMIN_TOKEN;
            // 패스워드 인코딩
            String encodedPassword = passwordEncoder.encode(password);
            // ROLE = 사용자
            UserRole role = UserRole.USER;

            kakaoUser = new User(username, encodedPassword, email, role, kakaoId);
            userRepository.save(kakaoUser);
        }
    }

    // 강제 로그인 처리
    UserDetailsImpl userDetails = new UserDetailsImpl(kakaoUser);
    Authentication authentication = new UsernamePasswordAuthenticationToken(userDetails, null, userDetails.getAuthorities());
    SecurityContextHolder.getContext().setAuthentication(authentication);
}
}

```

▼ src > main > java > com.sparta.springcore > repository > UserRepository 클래스

```

package com.sparta.springcore.repository;

import com.sparta.springcore.model.User;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.Optional;

public interface UserRepository extends JpaRepository<User, Long> {
    Optional<User> findByUsername(String username);
    Optional<User> findByKakaoId(Long kakaoId);

    Optional<User> findByEmail(String email);
}

```

