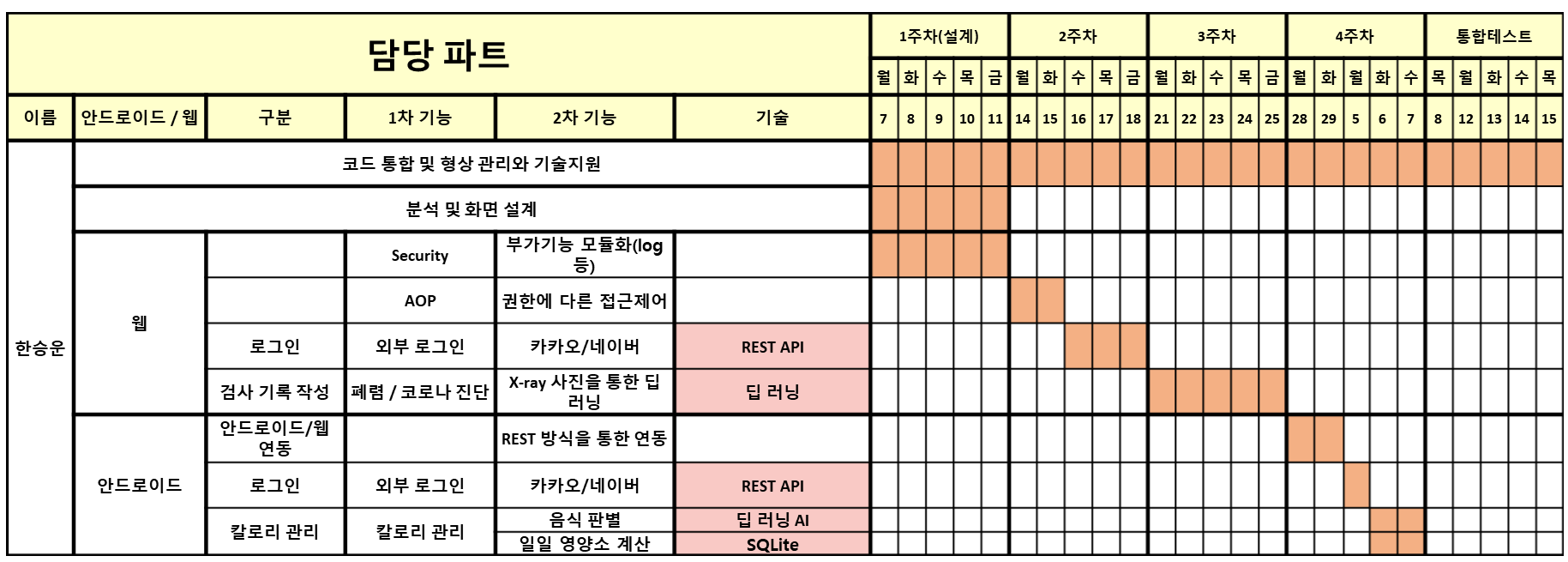
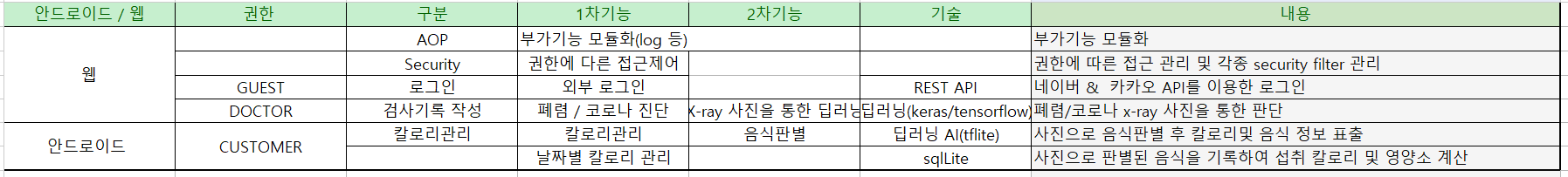
1. **개발일정**

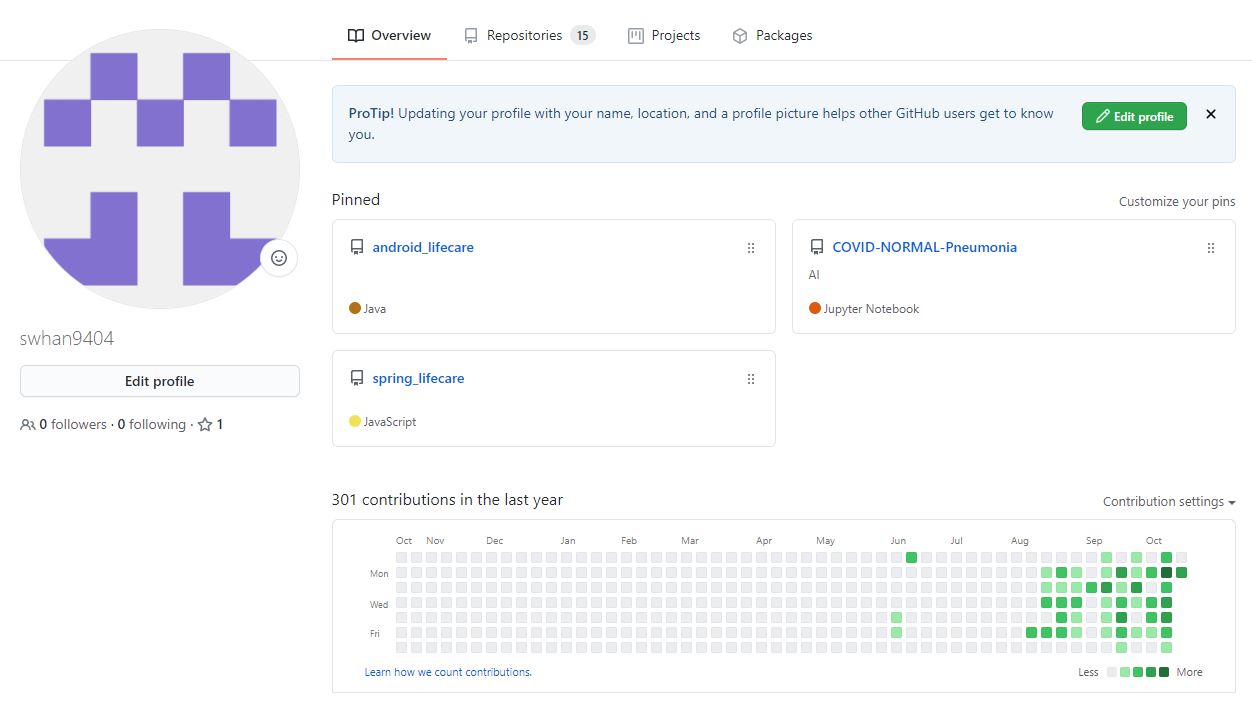
****

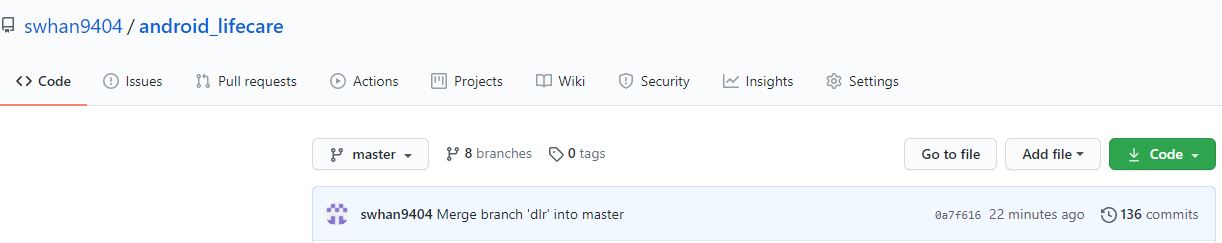
1. **기능정의서**

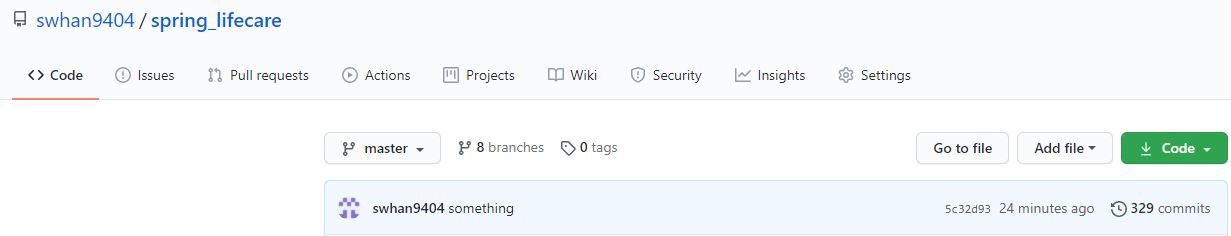


1. **Git 관리(**[**https://github.com/swhan9404/spring\_lifecare**](https://github.com/swhan9404/spring_lifecare)

**/ https://github.com/swhan9404/android\_lifecare)**

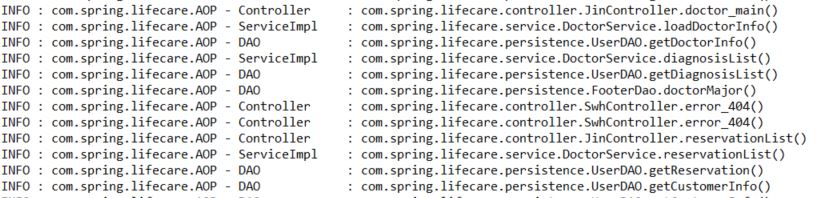
****

****

****

1. **스프링 세팅(AOP, security, transaction)**

AOP를 통한 log 표출



|  |  |  |  |
| --- | --- | --- | --- |
| 작성자 | 한승운 | 작성일 | 2020.09.10 |
| java | com.spring.lifecare.AOP.java | | |
| 소스캡처  @Aspect public class AOP {  @Pointcut("execution(\* com.spring.lifecare.service.User\*.\*(..))")  public void sec\_log() {};    private static final Logger *logger* = LoggerFactory.getLogger(AOP.class);  static String *name* = "";   static String *type* = "";   @Around("execution(\* com.spring.lifecare.controller.\*.\*(..)) or execution(\* com.spring.lifecare.service.\*.\*(..)) or execution(\* com.spring.lifecare.persistence.\*Impl.\*(..))")  public Object logPrint(ProceedingJoinPoint joinPoint) throws Throwable {   *type* = joinPoint.getSignature().getDeclaringTypeName();   if (*type*.indexOf("Controller") > -1) {   *name* = "Controller \t: ";   } else if (*type*.indexOf("Service") > -1) {   *name* = "ServiceImpl \t: ";   } else if (*type*.indexOf("DAO") > -1) {   *name* = "DAO \t\t: ";   }   *logger*.info(*name* + *type* + "." + joinPoint.getSignature().getName() + "()");   return joinPoint.proceed();   }    @Around("sec\_log()")  public Object logSecPrint(ProceedingJoinPoint joinPoint) throws Throwable {   *type* = joinPoint.getSignature().getDeclaringTypeName();     *logger*.info(*name* + *type* + "." + joinPoint.getSignature().getName() + "()");   return joinPoint.proceed();   } }  Controller, Service, DAO 접근에 따라 로그가 출력됨. | | | |

|  |  |
| --- | --- |
| servletContext | servletContext |
| <!-- AOP 관련 -->  <context:component-scan base-package="spring.project.bookshop4" use-default-filters="false">   <context:include-filter type="annotation" expression="org.springframework.stereotype.Controller"/>   </context:component-scan>    <aop:aspectj-autoproxy/>   <beans:bean id="loggerAspect" class="com.spring.lifecare.AOP" /> <!--// AOP 관련 -->  AOPBean등록 | |

|  |  |  |  |
| --- | --- | --- | --- |
| 작성자 | 한승운 | 작성일 | 2020.09.10 |
| context | | com.spring.lifecare.resources.security-context.xml | | |
| 소스캡처  <?xml version="1.0" encoding="UTF-8"?> <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:sec="http://www.springframework.org/schema/security"  xsi:schemaLocation="http://www.springframework.org/schema/security   https://www.springframework.org/schema/security/spring-security-5.2.xsd  http://www.springframework.org/schema/beans   http://www.springframework.org/schema/beans/spring-beans.xsd">   <!-- 주의 : 네임스페이스 security => sec로 수정 -->  <!-- 정적 리소스 파일들에는 보안 설정을 하지 않음 -->  <!-- 정적 리소스 파일들에는 보안 설정을 하지 않음 -->  <sec:http pattern="/include/\*\*" security="none" />  <sec:http pattern="/common/\*\*" security="none" />  <sec:http pattern="/guest/\*\*" security="none" />  <sec:http pattern="/android/\*\*" security="none" />  <sec:http auto-config="true" use-expressions="true" create-session="never">   <!-- 로그인한 사용자 영역(일반) -->   <sec:intercept-url pattern="/customer/\*\*" access="hasRole('ROLE\_CUSTOMER')" />    <!-- 로그인한 사용자 영역(의사) -->  <sec:intercept-url pattern="/doctor/\*\*" access="hasRole('ROLE\_DOCTOR')" />    <!-- 관리자 영역 설정 -->  <sec:intercept-url pattern="/admin/\*\*" access="hasRole('ROLE\_ADMIN')"/>    <!-- 권한에 관계없이 접속 가능한 영역(guest도 접속 가능) 맨뒤에 안넣으면 보안 안먹힘 주의-->  <sec:intercept-url pattern="/\*\*" access="permitAll" />    <!-- 로그인폼 -->  <sec:form-login login-page="/login"   login-processing-url="/login\_check"  authentication-success-handler-ref="userLoginSuccessHandler"  authentication-failure-handler-ref="userLoginFailureHandler"  username-parameter="userid" password-parameter="passwd" />  <!-- 로그아웃 -->  <sec:logout   logout-url="/logout"   invalidate-session="true"   success-handler-ref="userLogoutHandler"  delete-cookies="JSESSIONID"  />     <sec:session-management>  <!-- max-sessions="1" => 같은 브라우저의 여러 창에서 동시접속 막기   expired-url="/user/login.do" => 세션 만료시 이동할 주소  error-if-maximum-exceeded="true" => 최대 세션 초과시 에러 옵션   -->  <!--   <sec:concurrency-control max-sessions="1"  expired-url="/login"  error-if-maximum-exceeded="true"  />   -->   </sec:session-management>    <!-- 접근 권한이 없는 경우의 코드 -->  <sec:access-denied-handler ref="userDeniedHandler" />  </sec:http>    <bean id="userLoginSuccessHandler" class="com.spring.lifecare.service.UserLoginSuccessHandler" />  <bean id="userLoginFailureHandler" class="com.spring.lifecare.service.UserLoginFailureHandler" />  <bean id="userDeniedHandler" class="com.spring.lifecare.service.UserDeniedHandler" />  <bean id="userLogoutHandler" class="com.spring.lifecare.service.UserLogoutHandler" />      <!-- 로그인 인증을 처리하는 빈 -->  <!-- 주의 : 에러 => UserAuthenticationService에서 sqlSession을 매개변수 생성자에 주입하면 에러 사라짐 -->  <bean id="userService" class="com.spring.lifecare.service.UserAuthenticationService">  <constructor-arg name="sqlSession" ref="sqlSession" />  </bean>     <!-- 비밀번호 암호화 빈 -->  <bean id="bcryptPasswordEncoder" class="org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder" /> </beans>  security 암호화 및 로그인처리 | | | |

|  |  |
| --- | --- |
| context | security-datasource.xml |
| <?xml version="1.0" encoding="UTF-8"?> <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:tx="http://www.springframework.org/schema/tx"  xmlns:mybatis-spring="http://mybatis.org/schema/mybatis-spring"  xmlns:jdbc="http://www.springframework.org/schema/jdbc"  xsi:schemaLocation="http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc-4.3.xsd  http://mybatis.org/schema/mybatis-spring http://mybatis.org/schema/mybatis-spring-1.2.xsd  http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.3.xsd">   <!-- MyBatis 설정 시작 -->  <!-- 컨넥션풀 --> <!-- <bean id="dataSource" class="org.springframework.jndi.JndiObjectFactoryBean"> -->   <bean id="dataSourceSpied" class="org.springframework.jndi.JndiObjectFactoryBean">  <property name="jndiName" value="java:comp/env/jdbc/Oracle11g\_LifeCare" />  <!-- Oracle11g\_LifeCare -->    <property name="resourceRef" value="true" />  </bean>    <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">  <property name="dataSource" ref="dataSource" />  <property name="mapperLocations" value="classpath:mappers/\*\*/\*.xml" />  </bean>    <bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate">  <constructor-arg ref="sqlSessionFactory"/>  </bean>  <!-- MyBatis 설정 종료 -->     <!-- Mybatis Log -->  <bean id="dataSource" class="net.sf.log4jdbc.Log4jdbcProxyDataSource">  <constructor-arg ref="dataSourceSpied" />  <property name="logFormatter">  <bean class="net.sf.log4jdbc.tools.Log4JdbcCustomFormatter">  <property name="loggingType" value="MULTI\_LINE" />  <property name="sqlPrefix" value="[SQL]: "/>  </bean>  </property>  </bean>  <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">  <property name="basePackage" value="com.spring.lifecare" />  <property name="annotationClass" value="org.springframework.stereotype.Repository"/>  </bean>     <!-- 트랜잭션 관리자 : @Transactional 어노테이션 추가 -->  <!-- Namespaces 탭에서 tx 추가 -->  <tx:annotation-driven transaction-manager="transactionManager" />  <bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">  <property name="dataSource" ref="dataSource" />  </bean>   </beans>  커넥션풀 연결, mybatis 연동, 트랜잭션 설정, mapper 연결 | |

|  |  |
| --- | --- |
| service | com.spring.lifecare.service.UserAuthenticationService |
| public class UserAuthenticationService implements UserDetailsService {   @Autowired  SqlSessionTemplate sqlSession;    @Autowired  BCryptPasswordEncoder passwordEncoder;    public UserAuthenticationService(SqlSessionTemplate sqlSession) {  this.sqlSession = sqlSession;  }    /\* 핵심코드  \* 로그인 인증을 처리하는 코드  \* 매개변수가 userid뿐이지만 패스워드까지 전달된다.(매개변수를 username => userid로 수정)  \* 파라미터로 입력된 아이디값에 해당하는 테이블의 레코드를 읽어옴  \* 정보가 없으면 예외를 발생시키고, 있으면 해당 정보가 map(vo)로 리턴됨  \*/  @Override  public UserDetails loadUserByUsername(String userid) throws UsernameNotFoundException {  boolean emailcheck = true;  Map<String, Object> user = sqlSession.selectOne("com.spring.lifecare.persistence.UserDAO.selectUser", userid);  System.*out*.println("로그인 체크 ==> " + userid);  System.*out*.println(user);    // 인증실패시 인위적으로 예외 발생  if(user==null) throw new UsernameNotFoundException(userid);  else {  if(user.get("CUSTOMER\_ECHECK").toString().equals("0")) emailcheck=false;  }    // ArrayList 먼저 import, GrantedAuthority import  List<GrantedAuthority> authority = new ArrayList<GrantedAuthority>();    authority.add(new SimpleGrantedAuthority(user.get("AUTHORITY").toString()));  System.*out*.println("authority : "+ user.get("AUTHORITY").toString());    // 오라클에서는 필드명을 대문자로 취급  // 오라클에서는 BigInteger 관련 오류가 발생할수 있으므로 아래와 같이 처리  // 사용자가 입력한 값과 테이블의 USERNAME(=id), PASSWORD(아래)를 비교해서  // 비밀번호가 불일치시 UserLoginFailureHandler, 일치시 UserLoginSuccessHandler  // 스프링 시큐리티에서 체크하는 아래 필드로 select시에 별칭을 줌, 테이블의 암호화된 비밀번호와 사용자가 입력한 비밀번호를 내부적으로 비교처리    return new UserVO(user.get("USERNAME").toString(),  user.get("PASSWORD").toString(),  (Integer)Integer.*valueOf*(user.get("ENABLED").toString()) == 1,  true, true, emailcheck, authority, user.get("USERNAME").toString());  } }  로그인 체크, 이메일 인증, 아이디, 비번 불일치시 예외사항 발생처리 | |

|  |  |
| --- | --- |
| service | com.spring.lifecare.service.UserDeniedHandler |
| // 접근권한이 없는 경우 자동 실행  public class UserDeniedHandler implements AccessDeniedHandler {   // 관리자페이지에 일반사용자가 접근하거나, 허가받지 않은 사용자가 접근하는 경우 에러메시지 출력  @Override  public void handle(HttpServletRequest request, HttpServletResponse response,  AccessDeniedException accessDeniedException) throws IOException, ServletException {  PrintWriter out = response.getWriter();    out.println("<script>");  out.println("alert('접근권한이 없습니다.');");  out.println("</script>");     RequestDispatcher dispatcher = request.getRequestDispatcher("/main");  dispatcher.forward(request, response);  out.close();  } }  접근권한 없는 곳에 접근시 접근권한 없음 alert을 띄우는 handler | |

|  |  |
| --- | --- |
| service | com.spring.lifecare.service.UserLoginFailureHandler |
| //로그인이 실패한 경우 자동으로 실행 public class UserLoginFailureHandler implements AuthenticationFailureHandler {   @Override  public void onAuthenticationFailure(HttpServletRequest request, HttpServletResponse response,  AuthenticationException exception) throws IOException, ServletException {    String errormsg ="";    System.*out*.println(exception.getMessage());  if(exception instanceof BadCredentialsException) {  errormsg = "아이디나 비밀번호가 일치하지 않습니다";  } else if(exception instanceof LockedException) {  errormsg = "이메일인증을 하지 않았습니다.";  } else if(exception instanceof DisabledException) {  errormsg = "탈퇴한 아이디입니다";  } else if(exception instanceof InternalAuthenticationServiceException) {  errormsg = "회원가입하지 않은 아이디 입니다.";  }     request.setAttribute("fail", "1");  request.setAttribute("errMsg", errormsg);    RequestDispatcher dispatcher = request.getRequestDispatcher("/WEB-INF/views/guest/login.jsp");  dispatcher.forward(request, response);  } }  로그인 실패시 각 상황에 따라 error 메시지를 작성해서 보내주는 handler | |

|  |  |
| --- | --- |
| service | com.spring.lifecare.service |
| // 로그인이 성공한 경우 자동으로 실행 public class UserLoginSuccessHandler implements AuthenticationSuccessHandler {   @Override  public void onAuthenticationSuccess(HttpServletRequest request, HttpServletResponse response,  Authentication authentication) throws IOException, ServletException {      if(authentication.getPrincipal() instanceof UserVO) {  UserVO vo = (UserVO) authentication.getPrincipal();    request.getSession().setAttribute("userSession", vo.getUserid());          System.*out*.println("UserVO ==> " + vo);  }else if(authentication.getPrincipal() instanceof String) {  request.getSession().setAttribute("userSession", (String)authentication.getPrincipal());  }    RequestDispatcher dispatcher = request.getRequestDispatcher("/");    //권한에 따른 이동 위치  Collection<? extends GrantedAuthority> authorities = SecurityContextHolder.getContext().getAuthentication().getAuthorities();  String auth\_text = authorities.toString();    System.*out*.println("success Handler 출력 : " +auth\_text);  if(auth\_text.equals("[ROLE\_DOCTOR]")) {  dispatcher = request.getRequestDispatcher("/doctor/doctor\_main");  }else if(auth\_text.equals("[ROLE\_ADMIN]")) {  dispatcher = request.getRequestDispatcher("/admin/summary");  }    dispatcher.forward(request, response);    } }  로그인 성공시 각 권한에 따라 페이지 이동할 수 있도록 연동 | |

|  |  |
| --- | --- |
| VO | com.spring.lifecare.vo.UserVO |
| public class UserVO extends User {   private String userid;    // 두 생성자 중 첫번째 선택, 마지막 매개변수에 String userid 추가  // super는 부모 User 클래스  public UserVO(String username, String password, boolean enabled, boolean accountNonExpired,  boolean credentialsNonExpired, boolean accountNonLocked,  Collection<? extends GrantedAuthority> authorities, String userid) {  super(username, password, enabled, accountNonExpired, credentialsNonExpired, accountNonLocked, authorities);    this.userid = userid; // 추가  }   // getter, setter, toString 추가  public String getUserid() {  return userid;  }   public void setUserid(String userid) {  this.userid = userid;  }   @Override  public String toString() {  return "UserVO [userid=" + userid + "]";  }  }  UserVO 를 통해서 유저 정보를 저장 | |

|  |  |
| --- | --- |
| controller | com.spring.lifecare.controller.JinController |
| // 예약하기 @Transactional(rollbackFor=Exception.class) @ResponseBody @RequestMapping(value="/android/addReservation") public Map<String, String> addReservation(HttpServletRequest req, HttpServletResponse res, Model model) throws Exception {  try {  int appoint\_num = Integer.*parseInt*(req.getParameter("appoint\_num"));  String customer\_id = req.getParameter("customer\_id");  String doctor\_id = req.getParameter("doctor\_id");  String appoint\_date = req.getParameter("appoint\_date");     int updateCnt = 0;  int insertCnt = 0;    updateCnt = dao.updateAppoint(appoint\_num);  Map<String, Object> map = new HashMap<String, Object>();  System.*out*.println(appoint\_date);  String date = appoint\_date.substring(0,4) + "-" + appoint\_date.substring(6,8) + "-" + appoint\_date.substring(10,12) +   " " + appoint\_date.substring(15,17) + ":" + "00:00.0";  System.*out*.println(date);  java.sql.Timestamp reservation\_date = java.sql.Timestamp.*valueOf*(date);    System.*out*.println(reservation\_date);    map.put("appoint\_num", appoint\_num);  map.put("customer\_id", customer\_id);  map.put("doctor\_id", doctor\_id);  map.put("reservation\_date", reservation\_date);   insertCnt = dao.addReservation(map);    Map<String, String> out = new HashMap<String, String>();    if(updateCnt == 1 && insertCnt == 1) {  out.put("updateCnt", Integer.*toString*(updateCnt));  out.put("insertCnt", Integer.*toString*(insertCnt));  }else {  throw new Exception("트랜잭션");  }  System.*out*.println(out);    // 파이어 베이스  String tokenId="ffFjqIswxZ8:APA91bHRGb-d9Ke4wgXtn\_Ymm\_Kzpcht8t9POxmqYi-oK0lvuIYTk8xgGPWlQujhJU0dUAALnGtoNF6RbhGvzesCGV0\_gnKm1rujSg5AffaksMgF6S7UteN0FkkMGGGFWl01\_sFHBhlL";  String title="Lifecare";  String content= customer\_id + "님" + appoint\_date + "예약이 확정되었습니다.";    FcmUtil FcmUtil = new FcmUtil();  FcmUtil.send\_FCM(tokenId, title, content);    return out;  }catch(Exception e) {  TransactionAspectSupport.currentTransactionStatus().setRollbackOnly();  System.*out*.println(e.getMessage());  }  return null; }  진료 예약 부분에 transaction 적용을 통해 잘못된 데이터 입력이나 에러 발생을 미연에 방지 | |

1. **404 page 등 에러페이지 처리**

|  |  |
| --- | --- |
| controller | com.spring.lifecare.controller.swhController. |
| //404 에러 페이지 @RequestMapping("/error\_404") public String error\_404(HttpServletRequest request, Model model) {  return "error/error\_404"; }  //500 에러 페이지 @RequestMapping("/error\_500") public String error\_500(HttpServletRequest request, Model model) {  return "error/error\_500"; }  에러 페이지 구성을 통해 좀 더 안정적인 웹 환경 구성 | |

|  |  |
| --- | --- |
| xml | web.xml |
| <!-- 5. 에러처리 페이지 -->  <error-page>  <error-code>404</error-code>  <location>/error\_404</location> </error-page>      <error-page>  <error-code>500</error-code>  <location>/error\_500</location>  </error-page>  web.xml에서 error-page를 등록해서 404 페이지와 500 페이지 연결 | |
| View(JSP) | WEB-INF/views/error/error\_404.jsp |
| <%@ page language="java" contentType="text/html; charset=UTF-8"  pageEncoding="UTF-8"%> <%@ page isErrorPage = "true" %> <%@ include file="/resources/setting/setting.jsp"%> <!DOCTYPE html> <html> <head> <meta charset="UTF-8"> <title>Insert title here</title> </head> <body>  <%@include file="../common/header\_error.jsp"%>  <div class="divclass1" align="center">  <a href="javascript:history.back();">  <img src="${path\_resources}img/404error.png">  </a>  <br><br>  <h2>현재 페이지는 존재하지 않습니다</h2>  </div>  <%@include file="../common/footer.jsp"%> </body> </html>  404 페이지 구성 | |

1. **Enum 클래스를 활용한 많이 사용하는 변수 관리**

|  |  |
| --- | --- |
| controller | com.spring.lifecare.controller.swhController |
| public enum FinalString {  *CALLBACKIP*("192.168.219.125");    private String value;    private FinalString(String value) {  this.value= value;  }    public String getValue() {  return value;  } }  jsp, Controller, Service 등 여러 곳에서 사용되는 홈페이지 ip 주소를 Enum화 시켜서 수정을 보다 간략화하였습니다. | |

1. **카카오 로그인 연동**



|  |  |
| --- | --- |
| controller | com.spring.lifecare.controller.swhController |
| @RequestMapping("/kakaoLogin")  public String kakaoLogin(@RequestParam("code") String code, HttpSession session, Model model, HttpServletRequest request, HttpServletResponse response)   throws IOException {  String access\_Token = kakao.getAccessToken(code);  System.*out*.println("controller access\_token : " + access\_Token);    HashMap<String, Object> userKakaoInfo = kakao.getUserInfo(access\_Token);  System.*out*.println("userKakaoInfo : " + userKakaoInfo);    Map<String, String> userInfo = kakao.checkKakaoId(userKakaoInfo);    if(userInfo == null ||userInfo.get("USERNAME").equals("") || userInfo.get("USERNAME")==null ) {//데이터베이스에 카카오 로그인 정보가 없을경우  String errormsg = "카카오와 연동된 아이디가 없습니다";  model.addAttribute("fail", "1");  model.addAttribute("errMsg", errormsg);    return "guest/login";//로그인 페이지로 이동  }else {//데이터베이스에 카카오 로그인 정보가 있을경우  //클라이언트의 이메일이 존재할 때 세션에 해당 이메일과 토큰 등록  if (userInfo.get("email") != null) {  //session.setAttribute("userId", userInfo.get("email"));  //session.setAttribute("access\_Token", access\_Token);  model.addAttribute("email", userInfo.get("email"));  }  //권한 부여  kakao.kakaoAutehntication(request, response, userInfo);    }  model.addAttribute("trash", "trash");  System.*out*.println("model : " +model.toString());  return "main";  }  kakaoLogin 접근 제어 및 accssToken 발급 및 토큰 정보 추출 | |

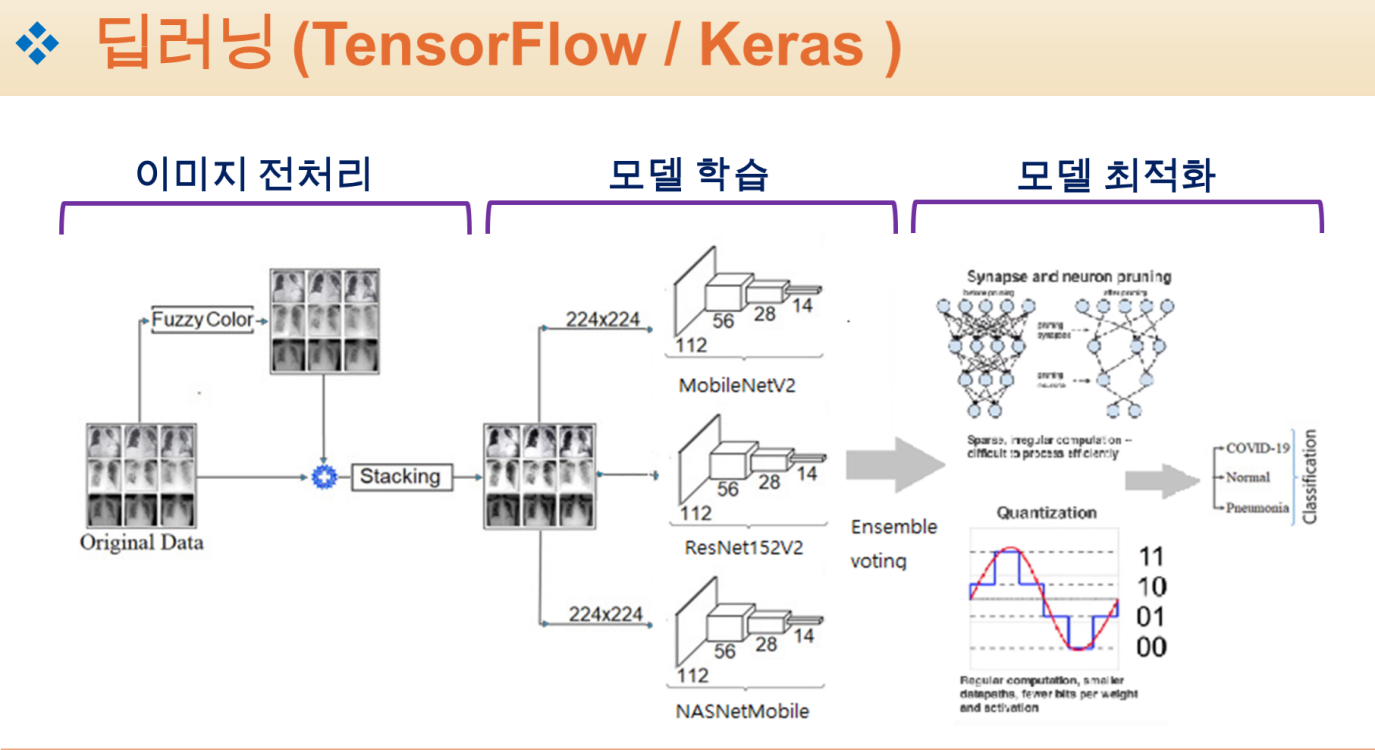
|  |  |
| --- | --- |
| service | com.spring.lifecare.service.KakaoServiceImpl |
| @Service public class KakaoLoginServiceImpl implements KakaoLoginService{    @Autowired  UserDAO userDAO;    String redirectIP = FinalString.*CALLBACKIP*.getValue();      public String getAccessToken (String authorize\_code) {  String access\_Token = "";  String refresh\_Token = "";  String reqURL = "https://kauth.kakao.com/oauth/token";    try {  URL url = new URL(reqURL);  HttpURLConnection conn = (HttpURLConnection) url.openConnection();    // POST 요청을 위해 기본값이 false인 setDoOutput을 true로  conn.setRequestMethod("POST");  conn.setDoOutput(true);    // POST 요청에 필요로 요구하는 파라미터 스트림을 통해 전송  BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(conn.getOutputStream()));  StringBuilder sb = new StringBuilder();  sb.append("grant\_type=authorization\_code");  sb.append("&client\_id=db39a7b6654b2f994af177b271416561");  sb.append("&redirect\_uri=http://"+redirectIP+"/lifecare/kakaoLogin");  sb.append("&code=" + authorize\_code);  bw.write(sb.toString());  bw.flush();    // 결과 코드가 200이라면 성공  int responseCode = conn.getResponseCode();  System.*out*.println("responseCode : " + responseCode);    // 요청을 통해 얻은 JSON타입의 Response 메세지 읽어오기  BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()));  String line = "";  String result = "";    while ((line = br.readLine()) != null) {  result += line;  }  System.*out*.println("response body : " + result);    // Gson 라이브러리에 포함된 클래스로 JSON파싱 객체 생성  JsonParser parser = new JsonParser();  JsonElement element = parser.parse(result);    access\_Token = element.getAsJsonObject().get("access\_token").getAsString();  refresh\_Token = element.getAsJsonObject().get("refresh\_token").getAsString();    System.*out*.println("access\_token : " + access\_Token);  System.*out*.println("refresh\_token : " + refresh\_Token);    br.close();  bw.close();  } catch (IOException e) {  e.printStackTrace();  }     return access\_Token;  }    public HashMap<String, Object> getUserInfo (String access\_Token) {  // 요청하는 클라이언트마다 가진 정보가 다를 수 있기에 HashMap타입으로 선언  HashMap<String, Object> userInfo = new HashMap<String, Object>();  String reqURL = "https://kapi.kakao.com/v2/user/me";  try {  URL url = new URL(reqURL);  HttpURLConnection conn = (HttpURLConnection) url.openConnection();  conn.setRequestMethod("POST");    // 요청에 필요한 Header에 포함될 내용  conn.setRequestProperty("Authorization", "Bearer " + access\_Token);    int responseCode = conn.getResponseCode();  System.*out*.println("responseCode : " + responseCode);    BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()));    String line = "";  String result = "";    while ((line = br.readLine()) != null) {  result += line;  }  System.*out*.println("response body : " + result);    JsonParser parser = new JsonParser();  JsonElement element = parser.parse(result);    JsonObject properties = element.getAsJsonObject().get("properties").getAsJsonObject();  JsonObject kakao\_account = element.getAsJsonObject().get("kakao\_account").getAsJsonObject();    try {  String email = kakao\_account.getAsJsonObject().get("email").getAsString();  userInfo.put("email", email);  }catch(Exception e) {  System.*out*.println("카카오에서 email 정보를 허용하지 않았습니다");  }    //카카오 토큰 id 가져오기  String id = element.getAsJsonObject().get("id").getAsString();  userInfo.put("id", id);  } catch (IOException e) {  e.printStackTrace();  }    return userInfo;  }    //카카오토큰 id가 존재하는지 확인  public Map<String, String> checkKakaoId(HashMap<String, Object> userInfo){  Map<String, String> user = userDAO.kakaoFindId((String)userInfo.get("id"));    if(user != null) {  System.*out*.println("카카오 로그인 된 실제 아이디 : "+ (String)user.get("USERNAME")+"/ 받은 토큰 id :"+ (String)userInfo.get("id"));  }  return user;  }    //카카오에서 권한주기  public void kakaoAutehntication(HttpServletRequest request, HttpServletResponse response, Map<String, String> user) {  //권한 manager 생성  OutLoginAuthenticationManager am = new OutLoginAuthenticationManager();    //실제 아이디   String id = (String)user.get("USERNAME");    try {  Authentication token = new UsernamePasswordAuthenticationToken(id, null);  Authentication result = am.authenticate(token);  SecurityContextHolder.getContext().setAuthentication(result);    System.*out*.println("result getPrincipal() : "+result.getPrincipal());    SecurityContext securityContext = SecurityContextHolder.getContext();  securityContext.setAuthentication(result);   // Create a new session and add the security context.  HttpSession session = request.getSession(true);  session.setAttribute("SPRING\_SECURITY\_CONTEXT", securityContext);  } catch(AuthenticationException e) {  System.*out*.println("Authentication failed: " + e.getMessage());  }    //손수 successHandler 태워주기  UserLoginSuccessHandler successHandler = new UserLoginSuccessHandler();  try {  successHandler.onAuthenticationSuccess(request, response, SecurityContextHolder.getContext().getAuthentication());  }catch(Exception e) {  System.*out*.print("내가 만든 successHanlder 에서 에러");  }    System.*out*.println("Successfully authenticated. Security context contains: \n" +  SecurityContextHolder.getContext().getAuthentication());    }   }  Token 정보 교환 설정 및 토큰정보로 dao에 접근해서 로그인 아이디 검색 및 로그인이 되면 kakaoAutehntication로 권한 부여 | |

1. **네이버로그인**

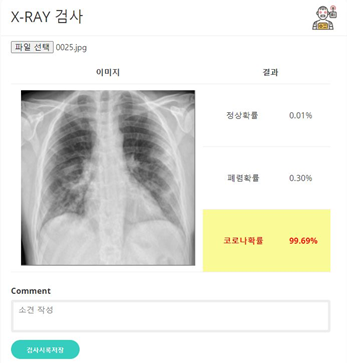
|  |  |
| --- | --- |
| controller | com.spring.lifecare.controller.swhController |
| //네이버 로그인 @RequestMapping("/naverLogin") public String naverLogin(HttpSession session, Model model, HttpServletRequest request, HttpServletResponse response) throws IOException {    HashMap<String, Object> userInfo= null;  Map<String, String> user = null;    try {  userInfo = naver.callback(request);  }catch(UnsupportedEncodingException e) {  System.*out*.println("encodingException");  }    System.*out*.println("userInfo : "+ userInfo);    if(userInfo != null) {  user = naver.checkNaverId(userInfo);  }    System.*out*.println("user" + user);    if(user == null ||user.get("USERNAME").equals("") || user.get("USERNAME")==null ) {//데이터베이스에 카카오 로그인 정보가 없을경우    String errormsg = "네이버와 연동된 아이디가 없습니다";  model.addAttribute("fail", "1");  model.addAttribute("errMsg", errormsg);    return "guest/login";//로그인 페이지로 이동  }else {//데이터베이스에 카카오 로그인 정보가 있을경우  //클라이언트의 이메일이 존재할 때 세션에 해당 이메일과 토큰 등록  if (userInfo.get("email") != null) {  //session.setAttribute("userId", userInfo.get("email"));  //session.setAttribute("access\_Token", access\_Token);  model.addAttribute("email", userInfo.get("email"));  }  //권한 부여  naver.naverAutehntication(request, response, user);  }  return "main"; }  NaverLogin 접근 제어 및 토큰 제어 | |

|  |  |
| --- | --- |
| service | com.spring.lifecare.service.NaverLoginServiceImpl |
| @Service public class NaverLoginServiceImpl implements NaverLoginService{  @Autowired  UserDAO userDAO;    String clientId = "LFKH6Ooda771daTdxSSO";  String clientSecret = "GWXuQveGk8";//애플리케이션 클라이언트 시크릿값";  String redirectIP = FinalString.*CALLBACKIP*.getValue();    /\*  public void naverLogin(HttpSession session) throws UnsupportedEncodingException {  String redirectURI = URLEncoder.encode("http://127.0.0.1/lifecare/naverLogin", "UTF-8");  SecureRandom random = new SecureRandom();  String state = new BigInteger(130, random).toString();  String apiURL = "https://nid.naver.com/oauth2.0/authorize?response\_type=code";  apiURL += "&client\_id=" + clientId;  apiURL += "&redirect\_uri=" + redirectURI;  apiURL += "&state=" + state;  session.setAttribute("state", state);  }  \*/    public HashMap<String, Object> callback(HttpServletRequest request) throws UnsupportedEncodingException {  HashMap<String, Object> userInfo = new HashMap<String, Object>();    String code = request.getParameter("code");  String state = request.getParameter("state");  String redirectURI = URLEncoder.*encode*("http://"+redirectIP+"/lifecare/naverLogin", "UTF-8");  String apiURL;  apiURL = "https://nid.naver.com/oauth2.0/token?grant\_type=authorization\_code&";  apiURL += "client\_id=" + clientId;  apiURL += "&client\_secret=" + clientSecret;  apiURL += "&redirect\_uri=" + redirectURI;  apiURL += "&code=" + code;  apiURL += "&state=" + state;  String access\_token = "";  String refresh\_token = "";  System.*out*.println("apiURL="+apiURL);    try {  URL url = new URL(apiURL);  HttpURLConnection con = (HttpURLConnection)url.openConnection();  con.setRequestMethod("GET");  int responseCode = con.getResponseCode();  BufferedReader br;  System.*out*.print("responseCode="+responseCode);  if(responseCode==200) { // 정상 호출  br = new BufferedReader(new InputStreamReader(con.getInputStream()));  } else { // 에러 발생  br = new BufferedReader(new InputStreamReader(con.getErrorStream()));  }  String inputLine;    StringBuffer res = new StringBuffer();  while ((inputLine = br.readLine()) != null) {  res.append(inputLine);  }  br.close();  if(responseCode==200) {  System.*out*.println(res.toString());  JSONParser parsing = new JSONParser();  Object obj = parsing.parse(res.toString());  JSONObject jsonObj = (JSONObject)obj;    access\_token = (String)jsonObj.get("access\_token");  refresh\_token = (String)jsonObj.get("refresh\_token");    System.*out*.println("access Token :" +access\_token);  System.*out*.println("refresh\_token :" +refresh\_token);  }  } catch (Exception e) {  System.*out*.println(e);  }        if(access\_token != null) { // access\_token을 잘 받아왔다면  try {  String header = "Bearer "+ access\_token;  String apiurl = "https://openapi.naver.com/v1/nid/me";  URL url = new URL(apiurl);  HttpURLConnection con = (HttpURLConnection)url.openConnection();  con.setRequestMethod("GET");  con.setRequestProperty("Authorization", header);  int responseCode = con.getResponseCode();  BufferedReader br;  if(responseCode==200) { // 정상 호출  br = new BufferedReader(new InputStreamReader(con.getInputStream()));  } else { // 에러 발생  br = new BufferedReader(new InputStreamReader(con.getErrorStream()));  }  String inputLine;  StringBuffer res = new StringBuffer();  while ((inputLine = br.readLine()) != null) {  res.append(inputLine);  }  br.close();    JSONParser parsing = new JSONParser();  Object obj = parsing.parse(res.toString());  JSONObject jsonObj = (JSONObject)obj;  JSONObject resObj = (JSONObject)jsonObj.get("response");    //왼쪽 변수 이름은 원하는 대로 정하면 된다.   //단, 우측의 get()안에 들어가는 값은 와인색 상자 안의 값을 그대로 적어주어야 한다.      String naverCode = (String)resObj.get("id");  System.*out*.println("naverCode : "+naverCode);  userInfo.put("id", naverCode);      if(resObj.containsKey("email")) {  String email = (String)resObj.get("email");  System.*out*.println("email : "+email);  userInfo.put("email", email);  }  if(resObj.containsKey("name")) {  String name = (String)resObj.get("name");  System.*out*.println("name : "+name);  userInfo.put("name", name);  }  } catch (Exception e) {  e.printStackTrace();  }  }  return userInfo;  }      //네이버 id가 존재하는지 확인  public Map<String, String> checkNaverId(HashMap<String, Object> userInfo){  Map<String, String> user = userDAO.naverFindId((String)userInfo.get("id"));    if(user != null) {  System.*out*.println("카카오 로그인 된 실제 아이디 : "+ (String)user.get("USERNAME")+"/ 받은 토큰 id :"+ (String)userInfo.get("id"));  }    return user;  }    //카카오에서 권한주기  public void naverAutehntication(HttpServletRequest request, HttpServletResponse response, Map<String, String> user) {  //권한 manager 생성  OutLoginAuthenticationManager am = new OutLoginAuthenticationManager();    //실제 아이디   String id = (String)user.get("USERNAME");    try {  Authentication token = new UsernamePasswordAuthenticationToken(id, null);  Authentication result = am.authenticate(token);  SecurityContextHolder.getContext().setAuthentication(result);    System.*out*.println("result getPrincipal() : "+result.getPrincipal());    SecurityContext securityContext = SecurityContextHolder.getContext();  securityContext.setAuthentication(result);   // Create a new session and add the security context.  HttpSession session = request.getSession(true);  session.setAttribute("SPRING\_SECURITY\_CONTEXT", securityContext);  } catch(AuthenticationException e) {  System.*out*.println("Authentication failed: " + e.getMessage());  }    //손수 successHandler 태워주기  UserLoginSuccessHandler successHandler = new UserLoginSuccessHandler();  try {  successHandler.onAuthenticationSuccess(request, response, SecurityContextHolder.getContext().getAuthentication());  }catch(Exception e) {  System.*out*.print("내가 만든 successHanlder 에서 에러");  }    System.*out*.println("Successfully authenticated. Security context contains: \n" +  SecurityContextHolder.getContext().getAuthentication());    }   }  Token 정보 교환 설정 및 토큰정보로 dao에 접근해서 로그인 아이디 검색 및 로그인이 되면 naverAutehntication로 권한 부여 | |

1. **X-ray를 통한 코로나 / 폐렴 / 정상 구분**

****

****

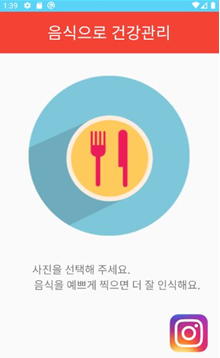
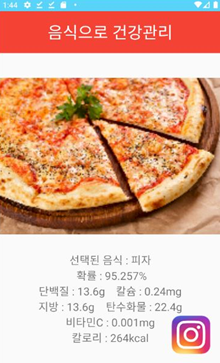
****

**파이썬 tensorflow와 keras로 이미지 전처리(FuzzyColor, Stacking)와 학습을 시킨 모델(MobileNetV2, ResNet152V2, NASNetMobile 을 emsemble voting한 모델)을 이용해, 사진을 통해 코로나 / 폐렴 / 정상을 구분해내고, 그 결과를 표시합니다. 의사는 인공지능의 결론을 보고 판단을 해서 더 효율적이고 정확하게 업무를 수행할 수 있도록 돕습니다.**

|  |  |
| --- | --- |
| view(jsp) | doctor\_medicalNote.jsp |
| sp  //딥러닝 - 코로나 $(function() {  //이미지 클릭시 업로드창 실행  $('#CoronaStart').click(function() {  console.log('코로나 딥러닝 시작');    var CoronaForm = new FormData(document.getElementById('CoronaForm'));    $.ajax({  type: "post",  enctype: 'multipart/form-data',  url: "${path}/doctor/DeepLearningCorona?${\_csrf.parameterName}=${\_csrf.token}",  data: CoronaForm,  // processData: true=> get방식, false => post방식  // contentType: true => application/x-www-form-urlencoded,   // false => multipart/form-data  processData: false,  contentType: false,  beforeSend: function() {  //통신을 시작할때 처리되는 함수   $('html').css("cursor","wait"); // 현재 html 문서위에 있는 마우스 커서를 로딩 중 커서로 변경  $('.wrap-loading').removeClass('display-none');    },  complete: function() {  //통신이 완료된 후 처리되는 함수  $('html').css("cursor","auto"); // 통신이 완료 된 후 다시 일반적인 커서 모양으로 변경  $('.wrap-loading').addClass('display-none');  },  success: function(data){  console.log("data : " + JSON.stringify(data));    var data\_JSON\_String = JSON.stringify(data);  var data\_parse = JSON.parse(data\_JSON\_String);    $("#normal").removeClass();  $("#normal").prev().removeClass();  $("#corona").removeClass();  $("#corona").siblings().removeClass();  $("#pneumonia").removeClass();  $("#pneumonia").siblings().removeClass();    if(data\_parse.normal >80){  $("#normal").addClass("CoronaText");  $("#normal").prev().addClass("CoronaText");  }  if(data\_parse.corona >80){  $("#corona").addClass("CoronaText");  $("#corona").siblings().addClass("CoronaText");  }  if(data\_parse.pneumonia >80){  $("#pneumonia").addClass("CoronaText");  $("#pneumonia").siblings().addClass("CoronaText");  }    $("#normal").text(data\_parse.normal+"%");  $("#corona").text(data\_parse.corona+"%");  $("#pneumonia").text(data\_parse.pneumonia+"%");    document.asdf.normal\_percentage.value = data\_parse.normal+"%";  document.asdf.corona\_percentage.value = data\_parse.corona+"%";  document.asdf.pneumonia\_percentage.value = data\_parse.pneumonia+"%";  alert("딥러닝 완료");  }    });  });  //업로드 파일체인지가 됬을경우 실행되는 이벤트 form태그에 fileProfile은 hidden으로 넣어줌 });  <div class="card">  <div class="card-title">  <span style="font-size: 1.8em; font-weight:500;">X-RAY 검사</span>  <div style="width:50px; height:50px; float:right;">  <a href="javascript:void(0)" id="CoronaStart"><img src="${path\_resources}images/ai.png"></a>  </div>   </div>   <div class="card-block">  <div class="wrap-loading display-none">  <div><img src="${path\_resources }img/deepLearning2\_edit.gif" /></div>  </div>   <div class="canvas-wrapper">  <form action="${path}/doctor/xrayExPro?${\_csrf.parameterName}=${\_csrf.token}" method="post" name="asdf" id="CoronaForm" enctype="multipart/form-data">  <input type="file" name="xray\_img" id="file">  <input type="hidden" name="${\_csrf.parameterName}" value="${\_csrf.token}">  <input type="hidden" name="customer\_id" value="${vo.getCustomer\_id()}">  <input type="hidden" name="normal\_percentage" value="">  <input type="hidden" name="corona\_percentage" value="">  <input type="hidden" name="pneumonia\_percentage" value="">  <table>  <thead>  <tr>  <th style="width:60%;">이미지</th>  <th style="width:40%; text-align:center" colspan="2">결과</th>  </tr>  </thead>  <tbody style="height:600px;">   <tr>   <td rowspan="3"><img style="height:350px;" id="img"></td>  <td>정상확률</td>  <td id ="normal"></td>  </tr>  <tr class='even'>  <td>폐렴확률</td>  <td id ="pneumonia"></td>  </tr>  <tr>  <td>코로나확률</td>  <td id="corona"></td>  </tr>  </tbody>   </table>  <label for="commentField">Comment</label>  <textarea style="resize: none;" placeholder="소견 작성" id="commentField" name="xray\_result"></textarea>  <input class="button-primary" type="button" value="검사시록저장" id="xrayExPro">  </form>  </div>  </div>  </div> </div>  Ajax를 통한 사진 전송 및 처리된 데이터에 따라 페이지 구성 변화, 로딩페이지 구현 | |

|  |  |
| --- | --- |
| service | com.spring.lifecare.service.DeepLearningServiceImpl |
| @Override  public Map<String, Object> DeepLearningCorona(MultipartHttpServletRequest req, Model model) {  MultipartFile file = req.getFile("xray\_img");    String TestImgSrc ="";    // 업로드할 파일의 최대 사이즈(10 \* 1024 \* 1024 = 10MB)  String saveDir = req.getSession().getServletContext().getRealPath("/resources/img/");   System.*out*.println("saveDir : "+saveDir);    ClassPathResource resource = new ClassPathResource("deeplearning/testImg/"); //resource 파일 루트    String formatType = file.getContentType().split("/")[1];//png    String realDir="";  try {  realDir = resource.getFile().getPath();  } catch (IOException e1) {  // *TODO Auto-generated catch block* e1.printStackTrace();  }  System.*out*.println("realDir : "+realDir);  System.*out*.println("파일이름 " +file.getOriginalFilename());    try {  file.transferTo(new File(saveDir+file.getOriginalFilename()));  System.*out*.println("file :" + file); // BufferedReader in = new BufferedReader( // new FileReader(saveDir + file.getOriginalFilename()) // ); // BufferedWriter out = new BufferedWriter( // new FileWriter(realDir + "/test2."+formatType) // );    FileInputStream fis = new FileInputStream(saveDir + file.getOriginalFilename());  FileOutputStream fos = new FileOutputStream(realDir + "/test2."+formatType);  TestImgSrc = realDir + "/test2."+formatType;  int data = 0;    while((data = fis.read()) != -1) {  fos.write(data);  }  //fos.flush();    //in.close();  //out.close();  }catch(Exception e) {  e.printStackTrace();  }      /////////////////////////// 여기까지 사진 저장  String modelSrc = "";  try {  modelSrc = new ClassPathResource("deeplearning/model/corona.py").getFile().getPath();  System.*out*.println("진짜 modleSrc : "+modelSrc);  } catch (IOException e) {  // *TODO Auto-generated catch block* e.printStackTrace();  }     /////////////////////////////////cmd 실행  List<String> list = new ArrayList<String>();    try {  //Linux의 경우는 /bin/bash  //Process process = Runtime.getRuntime().exec("/bin/bash");  Process process = Runtime.*getRuntime*().exec("cmd");  //Process의 각 stream을 받는다.  //process의 입력 stream  OutputStream stdin = process.getOutputStream();  //process의 출력 stream  InputStream stdout = process.getInputStream();   //입력 stream을 BufferedWriter로 받아서 콘솔로부터 받은 입력을 Process 클래스로 실행시킨다.  System.*out*.println("modelSrc : "+modelSrc.substring(0,modelSrc.length()-9));  List<String> commendList = new ArrayList<String>();  commendList.add("activate tensorflow3.6.5");  commendList.add("cd "+modelSrc.substring(0,modelSrc.length()-10));  System.*out*.println("실행어 : "+ "cd "+modelSrc.substring(0,modelSrc.length()-10));  modelSrc=modelSrc.substring(0,modelSrc.length()-9) +"mobileNetV2model7.h5";  commendList.add("python corona.py "+ modelSrc+" "+TestImgSrc);  System.*out*.println("실행어 : "+ "corona.py "+ modelSrc+" "+TestImgSrc);    try (BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(stdin))) {  for(int i=0; i<commendList.size(); i++) {  // 콘솔로 부터 엔터가 포함되면 input String 변수로 값이 입력됩니다.  String input =commendList.get(i);  // 콘솔에서 \n가 포함되어야 실행된다.(엔터의 의미인듯 싶습니다.)  input += "\n";  writer.write(input);  // Process로 명령어 입력  writer.flush();  // exit 명령어가 들어올 경우에는 프로그램을 종료합니다.  if ("exit\n".equals(input)) {  process.destroy();  }  }  writer.close();  } catch (IOException e) {  // *TODO Auto-generated catch block* e.printStackTrace();  }         try (BufferedReader reader = new BufferedReader(new InputStreamReader(stdout))) {  String line;  while ((line = reader.readLine()) != null) {  list.add(line);  System.*out*.println(line);    if(line.contains("[[")) {  reader.close();  break;  }  }  } catch (IOException e) {  e.printStackTrace();  }    try (BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(stdin))) {  for(int i=0; i<=commendList.size(); i++) {  // 콘솔로 부터 엔터가 포함되면 input String 변수로 값이 입력됩니다.  String input ="exit";  // 콘솔에서 \n가 포함되어야 실행된다.(엔터의 의미인듯 싶습니다.)  input += "\n";  writer.write(input);  // Process로 명령어 입력  // exit 명령어가 들어올 경우에는 프로그램을 종료합니다.  if ("exit\n".equals(input)) {  process.destroy();  }  }  } catch (IOException e) {  // *TODO Auto-generated catch block* //e.printStackTrace();  }  }catch(Exception e) {  e.printStackTrace();  }    ArrayList<String> realresult = new ArrayList<String>();    for(String test : list) {  if(test.contains("[[")) {  System.*out*.println(test.substring(2,test.length()-2));  test = test.substring(2,test.length()-2);    String[] testList = test.split(" ");  for(String a : testList) {  if(!a.equals("")) {  realresult.add(a);  }  }  }  }    Map<String, Object> result = new HashMap<String, Object>();        result.put("corona", String.*format*("%.2f", Double.*parseDouble*(realresult.get(0))\*100));  result.put("normal", String.*format*("%.2f", Double.*parseDouble*(realresult.get(1))\*100));  result.put("pneumonia", String.*format*("%.2f", Double.*parseDouble*(realresult.get(2))\*100));    System.*out*.println("최종값 코로나 : "+result.get("corona"));    return result;  }  사진을 전송받아 톰캣 서버에 저장하고, process 실행으로 딥러닝 실행파일을 실행하여 결과를 BufferReader를 통해 수집하고, Controller에 그 값을 전달 | |

|  |  |
| --- | --- |
| x-ray딥러닝 | python  소스는 파일로 첨부하였습니다. |
| 딥러닝 관련 pip install 해야할 리스트  pip install tensorflow  pip install keras  pip install matplotlib  pip install opencv-python  딥러닝 자료 : 3024장  - 전전처리 작업 : 크기조정(224x224) , 형식변환(JPG로 통일), for문을 사용하기 편하도록 이름변환  -전처리 작업 :학습의 능률 향상(Xray>ImageConverting 폴더)  FuzzyColor : 색상은 RGB로 이루어져있는데, RGB값이 그대로 들어가게 되면 의미 없는 수치에 가중치가 붙어 학습의 효율이 떨어지기 때문에, RGB 중 가장 높은 값만 살리고 나머지 값을 0으로 처리하여 이미지 선명도를 증대(fuzzy color.ipynb)  Stacking : 원본 사진과 FuzzyColor처리된 사진, 즉 2개의 다른 초점을 가진 사진을 가지고 다중 초점으로 좀더 선명한 사진을 만들어냄.(STACKING.ipynb)  oneHotEncoding : 컴퓨터는 라벨의 단어를 이해할 수 없기 때문에 0과 1로 구분할 수 있도록 재라벨링을 해주는 작업.(Image To Batch.ipynb)  -모델학습(손실함수는 adam으로 통일) : CNN- tensorflow, keras, sklearn(Xray>makingModel)  MobileNetV2 : 휴대기기 전용으로 만들어진 모델로써, 연산량과 파라미터 수를 줄인 네트워크.(epoch 50, batch 16) (exercise\_learning\_MoblileNetV2.ipynb)  ResNet152V2 : 학습의 깊이를 깊게 만드는데 특화된 모델로, Inception이라는 기술을 사용해서 조금 더 개선된 모델이다. (epoch 100, batch 16) (exercise\_learning\_ResNet.ipynb)  NASNetMobile : 학습의 깊이도 깊고, 구성된 학습망을 다시 RNN컨트롤러가 학습하여 더 좋은 변수를 찾을 수 있도록한 모델이다. (epoch 100, batch 16) (exercise\_learning\_NasNet.ipynb)  Emsembel Voting : 여러 모델의 집단 지성을 통해 더 나은 결과를 도출하는 방법. sklearn을 통해 구현하였으며, 모델들의 결과를 가지고 투표를 통해 최종 결과를 결정하는 방식이다.( exericse ensemble voting.ipynb)  -최적화 : tensorflow <이 작업은 뒤에 나올 음식분류에만 사용하였습니다>(foodClassifiy>optimizateModel)  pruning(가지치기) : 중요도가 0에 수렴하는 시냅스를 제거하여 연산의 수를 줄이는 작업(pruning For Model.ipynb)  quantization(양자화) : 가중치를 16비트 부동 소수점에 대한 float16으로 양자화하여 부동 소수점 모델의 크기를 줄이는 과정. 크기를 최대 75%~50% 까지 줄일 수 있습니다. 정확도의 하락을 최대한 방지하기 위해 재학습도 epoch 2로 진행합니다.(양자화 및 tflite화.ipynb) | |

****

1. **음식 사진을 통한 딥러닝과 건강관리 (안드로이드)**

**10만장의 이미지를 InceptionV3 모델을 가지고 학습시켜 101가지 음식을 구분해내는 모델을 만들었습니다. pruning과 양자화를 이용해 모델을 최적화 시키고, tflite 형태로 바꿔 안드로이드에 이식하였습니다. 인식된 음식을 날짜별로 저장해서 사용자가 칼로리 및 영양소 관리를 할 수 있도록 구성하였습니다.**

|  |  |
| --- | --- |
| activity | com.example.lifecare.food.FoodPhoto |
| public class FoodPhoto extends AppCompatActivity {   public static final String *FILE\_NAME* = "temp.jpg";  private static final String *ANDROID\_CERT\_HEADER* = "X-Android-Cert";  private static final String *ANDROID\_PACKAGE\_HEADER* = "X-Android-Package";  private static final int *MAX\_LABEL\_RESULTS* = 10;  private static final int *MAX\_DIMENSION* = 200;   private static final int *GALLERY\_PERMISSIONS\_REQUEST* = 0;  private static final int *GALLERY\_IMAGE\_REQUEST* = 1;  public static final int *CAMERA\_PERMISSIONS\_REQUEST* = 2;  public static final int *CAMERA\_IMAGE\_REQUEST* = 3;   private TextView mImageDetails;  private ImageView mMainImage;   Interpreter interpreter;  float[][] modelOutput;  //ByteBuffer modelOutput;   private FoodVO foodVO;  SimpleDateFormat format1 = new SimpleDateFormat( "yyyy-MM-dd");  Date time;   @Override  public void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_food\_photo*);   /\*상단바 숨기기\*/  getSupportActionBar().hide();    Button fab = findViewById(R.id.*fab*);  fab.setOnClickListener(view -> {  AlertDialog.Builder builder = new AlertDialog.Builder(FoodPhoto.this);  builder  .setMessage(R.string.*dialog\_select\_prompt*)  .setPositiveButton(R.string.*dialog\_select\_gallery*, (dialog, which) -> startGalleryChooser())  .setNegativeButton(R.string.*dialog\_select\_camera*, (dialog, which) -> startCamera());   builder.create().show();  });   mImageDetails = findViewById(R.id.*foodDetail*);  mMainImage = findViewById(R.id.*main\_image*);  foodVO= new FoodVO();   time = new Date();  interpreter = getTfliteInterpreter("your\_model.tflite"); // FirebaseCustomRemoteModel remoteModel = // new FirebaseCustomRemoteModel.Builder("your\_model").build(); // FirebaseModelDownloadConditions conditions = new FirebaseModelDownloadConditions.Builder() // .requireWifi() // .build(); // FirebaseModelManager.getInstance().download(remoteModel, conditions) // .addOnSuccessListener(new OnSuccessListener<Void>() { // @Override // public void onSuccess(Void v) { // // Download complete. Depending on your app, you could enable // // the ML feature, or switch from the local model to the remote // // model, etc. // } // }); // FirebaseModelManager.getInstance().getLatestModelFile(remoteModel) // .addOnCompleteListener(new OnCompleteListener<File>() { // @Override // public void onComplete(@NonNull Task<File> task) { // File modelFile = task.getResult(); // if (modelFile != null) { // interpreter = new Interpreter(modelFile); // } // } // });   }       private Interpreter getTfliteInterpreter(String modelPath){  try{  return new Interpreter(loadModelFile(FoodPhoto.this, modelPath));  }catch(Exception e){  e.printStackTrace();  }  System.*out*.println("인터프리터 null");  return null;   }   private MappedByteBuffer loadModelFile(Activity activity, String modelPath) throws IOException {  AssetFileDescriptor fileDescriptor = activity.getAssets().openFd(modelPath);  FileInputStream inputStream = new FileInputStream(fileDescriptor.getFileDescriptor());  FileChannel fileChannel = inputStream.getChannel();  long startOffset = fileDescriptor.getStartOffset();  long declaredLength = fileDescriptor.getDeclaredLength();  return fileChannel.map(FileChannel.MapMode.*READ\_ONLY*, startOffset, declaredLength);  }   private void doInterpreter(Bitmap yourInputImage){  Bitmap bitmap = Bitmap.*createScaledBitmap*(yourInputImage, 200, 200, true);  int batchNum =0;  float[][][][] input = new float[1][200][200][3];  for (int x = 0; x < 200; x++) {  for (int y = 0; y < 200; y++) {  int pixel = bitmap.getPixel(x, y);  input[batchNum][x][y][0] = Color.*red*(pixel) / 255.0f;  System.*out*.println(Color.*red*(pixel) / 255.0f);  input[batchNum][x][y][1] = Color.*green*(pixel) / 255.0f;  input[batchNum][x][y][2] = Color.*blue*(pixel) / 255.0f;  }  }  modelOutput=new float[1][101];  System.*out*.println("======================model Interpreter : "+interpreter.toString());  System.*out*.println("======================model modelOutput : "+modelOutput.toString());  interpreter.run(input, modelOutput);  }   public void getLabel(){   int maxNum = 0;  String maxLabel="";  float maxPossibility =0;   String maxProtein ="";  String maxCalcium = "";  String maxFat = "";  String maxCarbohydrate = "";  String maxVitaminC = "";  String maxKcal="";   try {  BufferedReader reader = new BufferedReader(  new InputStreamReader(getAssets().open("custom\_labels.txt")));  for (int i = 0; i < 101; i++) {  String line = reader.readLine();  String[] foodInfo = line.split(" ");   System.*out*.println(i);   String label = foodInfo[1];  String protein = foodInfo[2];  String calcium = foodInfo[3];  String fat = foodInfo[4];  String carbohydrate = foodInfo[5];  String vitaminC = foodInfo[6];  String kcal = foodInfo[7];   float probability = modelOutput[0][i];  if(maxPossibility <probability){  maxNum = i;  maxPossibility = probability;  maxLabel=label;  maxProtein= protein;  maxCalcium=calcium;  maxFat = fat;  maxCarbohydrate=carbohydrate;  maxVitaminC=vitaminC;  maxKcal = kcal;  }  Log.*i*("TAG", String.*format*("%s: %1.3f", label, probability));  }  } catch (IOException e) {  }   StringBuilder sb = new StringBuilder();  sb.append("선택된 음식 : " + maxLabel +"\n 확률 : " + String.*format*("%1.3f", maxPossibility\*100)+"%\n");  sb.append("단백질 : " + maxProtein +"g\t\t 칼슘 : " + maxCalcium+"mg\n");  sb.append("지방 : " + maxFat +"g\t\t 탄수화물 : " + maxCarbohydrate+"g\n");  sb.append("비타민C : " +maxVitaminC+"mg\n");  sb.append("칼로리 : "+maxKcal+"kcal");   foodVO.setDate(format1.format(time));  foodVO.setFoodName(maxLabel);  foodVO.setKcal(maxKcal);  foodVO.setCarbo(maxCarbohydrate);  foodVO.setProtein(maxProtein);  foodVO.setFat(maxFat);  foodVO.setFoodNum(maxNum);  int lid = getResources().getIdentifier("food\_"+ String.*format*("%03d",maxNum), "drawable", this.getPackageName());  System.*out*.println("======================================="+"food\_"+ String.*format*("%03d",maxNum));  System.*out*.println("======================================"+lid);  foodVO.setFoodImg(lid);   mImageDetails.setText(sb.toString());  }    public void startGalleryChooser() {  Toast.*makeText*(FoodPhoto.this, "갤러리가 선택되었습니다.", Toast.*LENGTH\_LONG*).show();  if (PermissionForFood.*requestPermission*(this, *GALLERY\_PERMISSIONS\_REQUEST*, Manifest.permission.*READ\_EXTERNAL\_STORAGE*)) {  Intent intent = new Intent();  intent.setType("image/\*");  intent.setAction(Intent.*ACTION\_GET\_CONTENT*);  startActivityForResult(Intent.*createChooser*(intent, "사진 선택"),  *GALLERY\_IMAGE\_REQUEST*);  }  }   public void startCamera() {  Toast.*makeText*(FoodPhoto.this, "카메라 선택되었습니다.", Toast.*LENGTH\_LONG*).show();  if (PermissionForFood.*requestPermission*(  this,  *CAMERA\_PERMISSIONS\_REQUEST*,  Manifest.permission.*READ\_EXTERNAL\_STORAGE*,  Manifest.permission.*CAMERA*)) {  Intent intent = new Intent(MediaStore.*ACTION\_IMAGE\_CAPTURE*);  Uri photoUri = FileProvider.*getUriForFile*(this, getApplicationContext().getPackageName() + ".provider", getCameraFile());  intent.putExtra(MediaStore.*EXTRA\_OUTPUT*, photoUri);  intent.addFlags(Intent.*FLAG\_GRANT\_READ\_URI\_PERMISSION*);  startActivityForResult(intent, *CAMERA\_IMAGE\_REQUEST*);  }  }   public File getCameraFile() {  File dir = getExternalFilesDir(Environment.*DIRECTORY\_PICTURES*);  return new File(dir, *FILE\_NAME*);  }   @Override  protected void onActivityResult(int requestCode, int resultCode, Intent data) {  super.onActivityResult(requestCode, resultCode, data);   if (requestCode == *GALLERY\_IMAGE\_REQUEST* && resultCode == *RESULT\_OK* && data != null) {  uploadImage(data.getData());  } else if (requestCode == *CAMERA\_IMAGE\_REQUEST* && resultCode == *RESULT\_OK*) {  Uri photoUri = FileProvider.*getUriForFile*(this, getApplicationContext().getPackageName() + ".provider", getCameraFile());  uploadImage(photoUri);  }  }    @Override  public void onRequestPermissionsResult( int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  super.onRequestPermissionsResult(requestCode, permissions, grantResults);  switch (requestCode) {  case *CAMERA\_PERMISSIONS\_REQUEST*:  if (PermissionUtils.*permissionGranted*(requestCode, *CAMERA\_PERMISSIONS\_REQUEST*, grantResults)) {  startCamera();  }  break;  case *GALLERY\_PERMISSIONS\_REQUEST*:  if (PermissionUtils.*permissionGranted*(requestCode, *GALLERY\_PERMISSIONS\_REQUEST*, grantResults)) {  startGalleryChooser();  }  break;  }  }   public void uploadImage(Uri uri) {  if (uri != null) {  try {  // scale the image to save on bandwidth  Bitmap bitmap =  scaleBitmapDown(  MediaStore.Images.Media.*getBitmap*(getContentResolver(), uri),  *MAX\_DIMENSION*);   doInterpreter(bitmap);  getLabel();  mMainImage.setImageBitmap(bitmap);   } catch (IOException e) {  Log.*d*("사진 선택에 실패 하였습니다. " , e.getMessage());  Toast.*makeText*(this, R.string.*image\_picker\_error*, Toast.*LENGTH\_LONG*).show();  }  } else {  Log.*d*("사진을 가져올 수 없습니다.", "사진을 가져올 수 없습니다.");  Toast.*makeText*(this, R.string.*image\_picker\_error*, Toast.*LENGTH\_LONG*).show();  }  }    private Bitmap scaleBitmapDown(Bitmap bitmap, int maxDimension) {   int originalWidth = bitmap.getWidth();  int originalHeight = bitmap.getHeight();  int resizedWidth = maxDimension;  int resizedHeight = maxDimension;   if (originalHeight > originalWidth) {  resizedHeight = maxDimension;  resizedWidth = (int) (resizedHeight \* (float) originalWidth / (float) originalHeight);  } else if (originalWidth > originalHeight) {  resizedWidth = maxDimension;  resizedHeight = (int) (resizedWidth \* (float) originalHeight / (float) originalWidth);  } else if (originalHeight == originalWidth) {  resizedHeight = maxDimension;  resizedWidth = maxDimension;  }  return Bitmap.*createScaledBitmap*(bitmap, resizedWidth, resizedHeight, false);  }   public void addFood(View w){  if(foodVO.getFoodName()==null){  StyleableToast.*makeText*(FoodPhoto.this, "음식사진을 아직 등록하지 않으셨습니다.", Toast.*LENGTH\_LONG*, R.style.*mytoast*).show();  return;  }   DbOpenHelper mDbOpenHelper = new DbOpenHelper(this);  mDbOpenHelper.open();  mDbOpenHelper.create();  mDbOpenHelper.insertColumn(foodVO);   StyleableToast.*makeText*(FoodPhoto.this, foodVO.getFoodName()+"음식을 등록하셨습니다.", Toast.*LENGTH\_LONG*, R.style.*mytoast*).show();  }   public void todayFoodButton(View w){  Intent intent = new Intent(FoodPhoto.this, FoodToday.class);  startActivity(intent);  } }  사진을 비트맵형식 224x224로 변환 후 deeplearning 모델을 interpreter를 통해 불러와 결과를 내고 그 결과를 화면에 뿌려주는 Activity  + 버튼 클릭 > 오늘 먹은 음식에 저장  메뉴 옆 화살표 버튼 > 오늘 먹은 음식 페이지로 이동하는 버튼으로 구성되어있다. | |
| layout | activity\_food\_photo.xml |
| <?xml version="1.0" encoding="utf-8"?> <androidx.coordinatorlayout.widget.CoordinatorLayout  xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  xmlns:tools="http://schemas.android.com/tools"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:id="@+id/foodphoto">   <LinearLayout  android:layout\_width="match\_parent"  android:layout\_height="78dp"  android:orientation="horizontal"  android:background="@color/food"  >  <TextView  android:layout\_width="350dp"  android:layout\_height="75dp"  android:gravity="center"  android:text=" 음식으로 건강관리 "  android:textColor="#ffffff"  android:textSize="30dp" />  <ImageView  android:id="@+id/todayFoodButton"  android:layout\_width="50dp"  android:layout\_height="wrap\_content"  android:src="@drawable/post\_button"  android:onClick="todayFoodButton"  />  </LinearLayout>    <androidx.appcompat.widget.LinearLayoutCompat  android:layout\_width="match\_parent"  android:layout\_height="600dp"  android:layout\_marginTop="120dp">   <LinearLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:orientation="vertical">   <ImageView  android:id="@+id/main\_image"  android:layout\_width="match\_parent"  android:src="@drawable/foodpage"  android:layout\_height="300dp"  android:scaleType="fitCenter"/>   <TextView  android:id="@+id/foodDetail"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_marginTop="30dp"  android:text="사진을 선택해 주세요.  \n 음식을 예쁘게 찍으면 더 잘 인식해요."  android:textAlignment="center"  android:textSize="20dp" />  </LinearLayout>   </androidx.appcompat.widget.LinearLayoutCompat>  <Button  android:id="@+id/fab"  android:layout\_width="80dp"  android:layout\_height="80dp"  android:layout\_gravity="bottom|center|right"  android:layout\_margin="16dp"  android:background="@drawable/cam"  />   <LinearLayout  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_gravity="bottom|left"  android:gravity="center\_horizontal"  android:orientation="vertical">   <com.google.android.material.floatingactionbutton.FloatingActionButton  android:id="@+id/fab\_add"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_marginBottom="@dimen/spacing\_smlarge"  android:layout\_marginLeft="@dimen/spacing\_smlarge"  android:layout\_marginRight="@dimen/spacing\_smlarge"  android:layout\_marginTop="@dimen/spacing\_middle"  android:onClick="addFood"  android:tint="@android:color/white"  app:fabSize="normal"  app:rippleColor="@android:color/white"  app:srcCompat="@drawable/ic\_add" />   </LinearLayout>  </androidx.coordinatorlayout.widget.CoordinatorLayout> | |
| activity | com.example.lifecare.food.FoodPhoto.java |
| public class FoodToday extends AppCompatActivity {  private WeekCalendar weekCalendar;   private RecyclerView recyclerView;  private AdapterListFood mAdapter;   private TextView totalKcal;  private ProgressBar kcal\_progress;  private TextView totalCarbo;  private ProgressBar carbo\_progress;  private TextView totalProtein;  private ProgressBar protein\_progress;  private TextView totalFat;  private ProgressBar fat\_progress;   private ArrayList<FoodVO> arrayList;    SimpleDateFormat format1 = new SimpleDateFormat( "yyyy-MM-dd");  Date time;   float totalKcalInfo=0;  float totalCarboInfo=0;  float totalProteinInfo=0;  float totalFatInfo=0;   @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_food\_today*);   time = new Date();  arrayList = new ArrayList<FoodVO>();   recyclerView = (RecyclerView) findViewById(R.id.*todayFoodRecyclerView*);  LinearLayoutManager linearLayoutManager = new LinearLayoutManager(this);  recyclerView.setLayoutManager(linearLayoutManager);  recyclerView.setHasFixedSize(true);  mAdapter = new AdapterListFood(FoodToday.this ,arrayList, R.layout.*item\_food*);  //recyclerView.setAdapter(mAdapter); 어뎁터에 정보넣기    totalKcal = (TextView) findViewById(R.id.*totalKcal*);  kcal\_progress = (ProgressBar) findViewById(R.id.*kcal\_progress*);   carbo\_progress = (ProgressBar) findViewById(R.id.*carbo\_progress*);  totalCarbo= (TextView)findViewById(R.id.*totalCarbo*);   protein\_progress = (ProgressBar) findViewById(R.id.*protein\_progress*);  totalProtein= (TextView)findViewById(R.id.*totalProtein*);   fat\_progress = (ProgressBar) findViewById(R.id.*fat\_progress*);  totalFat= (TextView)findViewById(R.id.*totalFat*);    /\*상단바 숨기기\*/  getSupportActionBar().hide();   weekCalendar = findViewById(R.id.*weekCalendar*);   DbOpenHelper mDbOpenHelper = new DbOpenHelper(this);  mDbOpenHelper.open();  mDbOpenHelper.create();   Cursor iCursor = mDbOpenHelper.selectColumns();  while(iCursor.moveToNext()){  String date = iCursor.getString(iCursor.getColumnIndex("date"));  String foodName = iCursor.getString(iCursor.getColumnIndex("foodName"));  String kcal = iCursor.getString(iCursor.getColumnIndex("kcal"));  String carbo = iCursor.getString(iCursor.getColumnIndex("carbo"));  String protein = iCursor.getString(iCursor.getColumnIndex("protein"));  String fat = iCursor.getString(iCursor.getColumnIndex("fat"));  int foodNum = iCursor.getInt(iCursor.getColumnIndex("foodNum"));  int foodImg = iCursor.getInt(iCursor.getColumnIndex("foodImg"));   if(date.equals(format1.format(time))){  FoodVO foodVO = new FoodVO();  foodVO.setDate(date);  foodVO.setFoodName(foodName);  foodVO.setKcal(kcal);  foodVO.setCarbo(carbo);  foodVO.setProtein(protein);  foodVO.setFat(fat);  foodVO.setFoodNum(foodNum);  foodVO.setFoodImg(foodImg);   totalKcalInfo = totalKcalInfo+Float.*valueOf*(foodVO.getKcal());  totalCarboInfo = totalCarboInfo+Float.*valueOf*(foodVO.getCarbo());  totalProteinInfo = totalProteinInfo + Float.*valueOf*(foodVO.getProtein());  totalFatInfo=totalFatInfo+Float.*valueOf*(foodVO.getFat());   arrayList.add(foodVO);  }    //progress bar 설정  kcal\_progress.setProgress((int)totalKcalInfo);  carbo\_progress.setProgress((int)totalCarboInfo);  protein\_progress.setProgress((int)totalProteinInfo);  fat\_progress.setProgress((int)totalFatInfo);   //정보 설정  totalKcal.setText(Integer.*toString*((int)totalKcalInfo));  totalCarbo.setText(Integer.*toString*((int)totalCarboInfo) + "/315g");  totalProtein.setText(Integer.*toString*((int)totalProteinInfo) + "/97g");  totalFat.setText(Integer.*toString*((int)totalFatInfo) + "/58g");  }   recyclerView.setAdapter(mAdapter);   weekCalendar.setOnDateClickListener(new OnDateClickListener() {  @Override  public void onDateClick(DateTime dateTime) {  Toast.*makeText*(FoodToday.this,  "당신의 선택" + dateTime.toString(), Toast.*LENGTH\_SHORT*).show();  }  });  } }  오늘 먹은 음식을 SQLite로 구성된 database에서 가져와서 kcal 및 영양소 섭취량을 계산하고 RecycleView에 그 내역을 뿌려주는 역할, 상단에 달력을 주 단위로 구성하여 이전의 기록에 접근하여 확인하는 것도 가능하게 함. | |
| adapter | com.example.lifecare.food.AdapterListFood |
| public class AdapterListFood extends RecyclerView.Adapter<RecyclerView.ViewHolder> {   private List<FoodVO> items = new ArrayList<>();   private Context ctx;   @LayoutRes  private int layout\_id;   private OnItemClickListener mOnItemClickListener;   public interface OnItemClickListener {  void onItemClick(View view, FoodVO obj, int position);  }   public void setOnItemClickListener(final OnItemClickListener mItemClickListener) {  this.mOnItemClickListener = mItemClickListener;  }   public AdapterListFood(Context context, List<FoodVO> items, @LayoutRes int layout\_id) {  this.items = items;  ctx = context;  this.layout\_id = layout\_id;  }   public class OriginalViewHolder extends RecyclerView.ViewHolder {  public ImageView foodListImg;  public TextView foodKcal;  public TextView tdz;  public TextView foodListName;  public View lyt\_parent2;   public OriginalViewHolder(View v) {  super(v);  foodListImg = v.findViewById(R.id.*foodListImg*);  foodKcal = v.findViewById(R.id.*foodKcal*);  tdz = v.findViewById(R.id.*tdz*);  foodListName = v.findViewById(R.id.*foodListName*);  lyt\_parent2 = v.findViewById(R.id.*lyt\_parent2*);  }  }   @Override  public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {  RecyclerView.ViewHolder vh;  View v = LayoutInflater.*from*(parent.getContext()).inflate(R.layout.*item\_food*, parent, false);  vh = new OriginalViewHolder(v);  return vh;  }   // Replace the contents of a view (invoked by the layout manager)  @Override  public void onBindViewHolder(RecyclerView.ViewHolder holder, final int position) {  if (holder instanceof OriginalViewHolder) {  OriginalViewHolder view = (OriginalViewHolder) holder;   FoodVO n = items.get(position);  view.foodKcal.setText(n.getKcal()+" Kcal");   String tdz = "탄 : "+n.getCarbo() +"g 단 : "+n.getProtein() + "g 지 : "+n.getFat()+"g";  SpannableStringBuilder ssb = new SpannableStringBuilder(tdz); /\*  ssb.setSpan(new BackgroundColorSpan(Color.parseColor("#f6830f")), tdz.indexOf("탄"), tdz.indexOf("탄")+1, Spanned.SPAN\_EXCLUSIVE\_EXCLUSIVE);  ssb.setSpan(new BackgroundColorSpan(Color.parseColor("#0e918c")), tdz.indexOf("단"), tdz.indexOf("단")+1, Spanned.SPAN\_EXCLUSIVE\_EXCLUSIVE);  ssb.setSpan(new BackgroundColorSpan(Color.parseColor("#4c00b8")), tdz.indexOf("지"), tdz.indexOf("지")+1, Spanned.SPAN\_EXCLUSIVE\_EXCLUSIVE);   ssb.setSpan(new ForegroundColorSpan(Color.parseColor("#ffffff")), tdz.indexOf("탄"), tdz.indexOf("탄")+1, Spannable.SPAN\_EXCLUSIVE\_EXCLUSIVE);  ssb.setSpan(new ForegroundColorSpan(Color.parseColor("#ffffff")), tdz.indexOf("단"), tdz.indexOf("단")+1, Spannable.SPAN\_EXCLUSIVE\_EXCLUSIVE);  ssb.setSpan(new ForegroundColorSpan(Color.parseColor("#ffffff")), tdz.indexOf("지"), tdz.indexOf("지")+1, Spannable.SPAN\_EXCLUSIVE\_EXCLUSIVE); \*/  ssb.setSpan(new ForegroundColorSpan(Color.*parseColor*("#f6830f")), tdz.indexOf("탄")+1, tdz.indexOf("단")-1, Spannable.*SPAN\_EXCLUSIVE\_EXCLUSIVE*);  ssb.setSpan(new ForegroundColorSpan(Color.*parseColor*("#0e918c")), tdz.indexOf("단")+1, tdz.indexOf("지")-1, Spannable.*SPAN\_EXCLUSIVE\_EXCLUSIVE*);  ssb.setSpan(new ForegroundColorSpan(Color.*parseColor*("#4c00b8")), tdz.indexOf("지")+1, tdz.length(), Spannable.*SPAN\_EXCLUSIVE\_EXCLUSIVE*);   view.tdz.setText(ssb);  view.foodListImg.setImageResource(n.getFoodImg());  view.foodListName.setText(n.getFoodName());  }  }   @Override  public int getItemCount() {  return items.size();  } }  List<FoodVO>를 전달받아 recycelView를 구성해주는 adapter  SpannableStringBuilder를 통해서 색깔과 크기를 부분적으로 변경하여 가독성을 증가시켰다. | |
| database | com.example.lifecare.food.DataBases |
| public final class DataBases {  public static final class CreateDB implements BaseColumns {  public static final String *key* = "key";  public static final String *date* = "date";  public static final String *foodName* = "foodName";  public static final String *kcal* = "kcal";  public static final String *carbo* = "carbo";  public static final String *protein* = "protein";  public static final String *fat* = "fat";  public static final String *foodNum* = "foodNum";  public static final String *foodImg* = "foodImg";   public static final String *\_TABLENAME0* = "foodTable";  public static final String *\_CREATE0* = "create table if not exists "+*\_TABLENAME0*+"("  +*key*+" integer primary key autoincrement, "  +*date*+" text not null , "  +*foodName*+" text not null , "  +*kcal*+" text not null , "  +*carbo*+" text not null , "  +*protein*+" text not null , "  +*fat*+" text not null , "  +*foodNum*+" integer not null ,"  +*foodImg*+" integer not null );";  } }  데이터 베이스 생성(sqlLite) final 형식으로 구성하였고, foodTable이 존재하지 않을 시 생성하게 sql문을 구성하였습니다. 사용자가 데이터를 삭제해도 에러 없이 어플이 돌아가도록 구성하였습니다. | |
| java | com.example.lifecare.food.SQLiteOpenHelper |
| public class DbOpenHelper {  public static int *KeyNum* = 0;  private static final String *DATABASE\_NAME* = "InnerDatabase(SQLite).db";  private static final int *DATABASE\_VERSION* = 1;  public static SQLiteDatabase *mDB*;  private DatabaseHelper mDBHelper;  private Context mCtx;   private class DatabaseHelper extends SQLiteOpenHelper {   public DatabaseHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version) {  super(context, name, factory, version);  }   @Override  public void onCreate(SQLiteDatabase db){  db.execSQL(DataBases.CreateDB.*\_CREATE0*);  }   @Override  public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion){  db.execSQL("DROP TABLE IF EXISTS "+DataBases.CreateDB.*\_TABLENAME0*);  onCreate(db);  }  }   public DbOpenHelper(Context context){  this.mCtx = context;  }   public DbOpenHelper open() throws SQLException {  mDBHelper = new DatabaseHelper(mCtx, *DATABASE\_NAME*, null, *DATABASE\_VERSION*);  *mDB* = mDBHelper.getWritableDatabase();  return this;  }   public void create(){  mDBHelper.onCreate(*mDB*);  }   public void close(){  *mDB*.close();  }   //입력  public long insertColumn(FoodVO foodVO){  ContentValues values = new ContentValues();  values.put(DataBases.CreateDB.*key*, *KeyNum*);  *KeyNum*++;  values.put(DataBases.CreateDB.*date*, foodVO.getDate());  values.put(DataBases.CreateDB.*foodName*, foodVO.getFoodName());  values.put(DataBases.CreateDB.*kcal*, foodVO.getKcal());  values.put(DataBases.CreateDB.*carbo*, foodVO.getCarbo());  values.put(DataBases.CreateDB.*protein*, foodVO.getProtein());  values.put(DataBases.CreateDB.*fat*, foodVO.getFat());  values.put(DataBases.CreateDB.*foodNum*, foodVO.getFoodNum());  values.put(DataBases.CreateDB.*foodImg*, foodVO.getFoodImg());   return *mDB*.insert(DataBases.CreateDB.*\_TABLENAME0*, null, values);  }   public Cursor selectColumns(){  return *mDB*.query(DataBases.CreateDB.*\_TABLENAME0*, null, null, null, null, null, null);  } }  SQLite 데이터 베이스 insert 와 select를 수행하는 class, 서로 다른 Activity에서 db 접근을 하기 쉽도록 따로 구성하였습니다. | |

**2) 웹과 안드로이드 연동**

|  |  |
| --- | --- |
| activity | com.example.lifecare.EclipseConnect.SignInActivity |
| @RequiresApi(api = Build.VERSION\_CODES.*M*) public class SignInActivity extends AppCompatActivity {   UserVO userVO = UserVO.*getInstance*();  EditText edtId, edtPwd;  Button btnSignIn;  ImageView btnKakaoLogin;  CheckBox idRemember;   SharedPreferenceHandler sh;   /\*지문인식 부분\*/  private static final String *KEY\_NAME* = "example\_key";  private FingerprintManager fingerprintManager;  private KeyguardManager keyguardManager;  private KeyStore keyStore;  private KeyGenerator keyGenerator;  private Cipher cipher;  private FingerprintManager.CryptoObject cryptoObject;  private ImageView jimunLogin;  /\*지문인식 부분\*/   //네이버 로그인 관련  private OAuthLoginButton mOAuthLoginButton;   //client 정보  public static String *OAUTH\_CLIENT\_ID* = "LFKH6Ooda771daTdxSSO";  public static String *OAUTH\_CLIENT\_SECRET* = "GWXuQveGk8";  public static String *OAUTH\_CLIENT\_NAME* = "홈페이지 연습";  public static OAuthLogin *mOAuthLoginInstance*;  private static Context *mContext*;   @Override  protected void onCreate(@Nullable Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_login*);   *mContext* =this;   edtId = (EditText) findViewById(R.id.*id*);  edtPwd = (EditText) findViewById(R.id.*pwd*);  btnSignIn = (Button) findViewById(R.id.*btn\_login*);  btnKakaoLogin = (ImageView) findViewById(R.id.*kakao*);  jimunLogin = (ImageView) findViewById(R.id.*jimun*);  idRemember = (CheckBox) findViewById(R.id.*idRemember*);   sh = new SharedPreferenceHandler(getApplicationContext());   if(sh.getId() != null ){  idRemember.setChecked(true);  edtId.setText(sh.getId());  }    btnSignIn.setOnClickListener(new View.OnClickListener() {  public void onClick(View v) {  InnerTask task = new InnerTask();  Map<String, String> map = new HashMap<>();  map.put("id", edtId.getText().toString());  map.put("pwd", edtPwd.getText().toString());   task.execute(map);  }  });   btnKakaoLogin.setOnClickListener(new View.OnClickListener() {  public void onClick(View v) {  Intent intent = new Intent(SignInActivity.this, KakaoLoginCheck.class);  startActivity(intent); /\*  if(userVO.getId() != ""){  Intent intent = new Intent(SignInActivity.this, MainActivity.class);  StyleableToast.makeText(getApplicationContext(), userVO.getId()+"님 로그인 되었습니다.", Toast.LENGTH\_LONG, R.style.mytoast).show();  startActivity(intent);  }else{  StyleableToast.makeText(getApplicationContext(), "로그인에 실패하였습니다", Toast.LENGTH\_LONG, R.style.mytoast).show();  }  \*/  }  });   /\*지문로그인\*/  jimunLogin.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  String Customer\_fingerprint = "";  if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*M*) {   /\*생체인식기능 manifests 에 권한 추가후 사용한다\*/  fingerprintManager = (FingerprintManager) getSystemService(*FINGERPRINT\_SERVICE*);   /\*잠금 해제와 해제된 잠금화면을 복구 시키는 역할을한다\*/  keyguardManager = (KeyguardManager) getSystemService(*KEYGUARD\_SERVICE*);   /\*지문 상태별 알림\*/  if (!fingerprintManager.isHardwareDetected()) {//Manifest에 Fingerprint 퍼미션을 추가해 워야 사용가능  Toast.*makeText*(SignInActivity.this, "지문을 사용할 수 없는 디바이스 입니다.", Toast.*LENGTH\_SHORT*).show();  } else if (ContextCompat.*checkSelfPermission*(SignInActivity.this, Manifest.permission.*USE\_FINGERPRINT*) != PackageManager.*PERMISSION\_GRANTED*) {  Toast.*makeText*(SignInActivity.this, "지문사용을 허용해 주세요.", Toast.*LENGTH\_SHORT*).show();  /\*잠금화면 상태를 체크한다.\*/  } else if (!keyguardManager.isKeyguardSecure()) {  Toast.*makeText*(SignInActivity.this, "잠금화면을 설정해 주세요.", Toast.*LENGTH\_SHORT*).show();  } else if (!fingerprintManager.hasEnrolledFingerprints()) {  Toast.*makeText*(SignInActivity.this, "등록된 지문이 없습니다.", Toast.*LENGTH\_SHORT*).show();  } else {//모든 관문을 성공적으로 통과(지문인식을 지원하고 지문 사용이 허용되어 있고 잠금화면이 설정되었고 지문이 등록되어 있을때)  Toast.*makeText*(SignInActivity.this, "손가락을 홈버튼에 대 주세요.", Toast.*LENGTH\_SHORT*).show();   generateKey();  if (cipherInit()) {  //지문 핸들러실행  jimunTask task = new jimunTask();  FingerprintHandler fingerprintHandler = new FingerprintHandler(SignInActivity.this);  fingerprintHandler.startAutho(fingerprintManager, cryptoObject ,task);  }  }  }  }  });   //초기화  initData();  }   public void onButtonClick(View v) throws Throwable {  switch (v.getId()) {  case R.id.*buttonOAuthLoginImg*: {  *mOAuthLoginInstance*.startOauthLoginActivity(SignInActivity.this, mOAuthLoginHandler);  break;  }  default:  break;  }  }    private void initData() {  //초기화  *mOAuthLoginInstance* = OAuthLogin.getInstance();  *mOAuthLoginInstance*.init(*mContext*, *OAUTH\_CLIENT\_ID*, *OAUTH\_CLIENT\_SECRET*, *OAUTH\_CLIENT\_NAME*);   mOAuthLoginButton = (OAuthLoginButton) findViewById(R.id.*buttonOAuthLoginImg*);  mOAuthLoginButton.setOAuthLoginHandler(mOAuthLoginHandler);   //custom img로 변경시 사용  //mOAuthLoginButton.setBgResourceId(R.drawable.btn\_naver\_white\_kor);  }   */\*\*  \* OAuthLoginHandler를 startOAuthLoginActivity() 메서드 호출 시 파라미터로 전달하거나 OAuthLoginButton  객체에 등록하면 인증이 종료되는 것을 확인할 수 있습니다.  \*/* private OAuthLoginHandler mOAuthLoginHandler = new OAuthLoginHandler() {  @Override  public void run(boolean success) {  if (success) {  String accessToken = *mOAuthLoginInstance*.getAccessToken(*mContext*);  String refreshToken = *mOAuthLoginInstance*.getRefreshToken(*mContext*);  long expiresAt = *mOAuthLoginInstance*.getExpiresAt(*mContext*);  String tokenType = *mOAuthLoginInstance*.getTokenType(*mContext*);   redirectSignupActivity();  } else {  String errorCode = *mOAuthLoginInstance*.getLastErrorCode(*mContext*).getCode();  String errorDesc = *mOAuthLoginInstance*.getLastErrorDesc(*mContext*);  Toast.*makeText*(*mContext*, "errorCode:" + errorCode + ", errorDesc:" + errorDesc, Toast.*LENGTH\_SHORT*).show();  }  };  };   // 네이버 성공 후 이동할 액티비티  protected void redirectSignupActivity() {   new RequestApiTask().execute();   finish();  }   // 네이버 로그인 성공시 토큰 받아오기    //각 Activity 마다 Task 작성  public class InnerTask extends AsyncTask<Map, Integer, String> {   //doInBackground 실행되기 이전에 동작  @Override  protected void onPreExecute() {  super.onPreExecute();  }   //작업을 쓰레드로 처리  @Override  protected String doInBackground(Map... maps) {  //HTTP 요청 준비 - Post 로 바꿀것  HttpClient.Builder http = new HttpClient.Builder("POST", Web.*servletURL* + "androidSignIn"); //스프링 url  //파라미터 전송  http.addAllParameters(maps[0]);  System.*out*.println("http ㅁㄴㅇㄻㄴㅇㄹ: " + http.getParameter());  //HTTP 요청 전송  HttpClient post =http.create();  post.request();   String body = post.getBody(); //Web의 Controller에서 리턴한 값  return body;  }   //doInBackground 종료되면 동작  */\*\*  \** ***@param*** *s : doInBackground에서 리턴한 body. JSON 데이터  \*/* @Override  protected void onPostExecute(String s) {  Log.*d*("JSON\_RESULT ㅁㄴㅇㄹ", s);  //JSON으로 받은 데이터를 VO Obejct로 바꿔준다.  if(s.length() > 0) {  Gson gson = new Gson();  UserVO m = gson.fromJson(s, UserVO.class);  if (m.getId() != null && m.getId() != "") {  // 페이지 이동  Intent intent = new Intent(SignInActivity.this, MainActivity.class);   //로그인 유지  userVO.setId(m.getId());  userVO.setEnabled(m.getEnabled());  userVO.setCustomer\_echeck(m.getCustomer\_echeck());   //아이디 저장 & 자동로그인  if(idRemember.isChecked()){  sh.keepId(m.getId());  }else{  sh.clear();  }    StyleableToast.*makeText*(getApplicationContext(), m.getId()+"님 로그인 되었습니다.", Toast.*LENGTH\_LONG*, R.style.*mytoast*).show();  startActivity(intent);  } else if (m.getEnabled() != "1") {  Toast.*makeText*(getApplicationContext(), "회원 정보가 올바르지 않습니다.", Toast.*LENGTH\_LONG*).show();  } else {  Toast.*makeText*(getApplicationContext(), "가입 인증이 필요한 회원입니다.", Toast.*LENGTH\_LONG*).show();  }  } else {  Toast.*makeText*(getApplicationContext(), "로그인 실패", Toast.*LENGTH\_SHORT*).show();  }  }  }   @Override  public void onBackPressed() {  Intent intent = new Intent(SignInActivity.this, MainActivity.class);  startActivity(intent);  }   @Override  protected void onDestroy() {   System.*out*.println("=========================== sign destroy");  super.onDestroy();  }   private class RequestApiTask extends AsyncTask<Void, Void, String> {   @Override  protected String doInBackground(Void... params) {  String url = "https://openapi.naver.com/v1/nid/me";  String at = *mOAuthLoginInstance*.getAccessToken(*mContext*);  return *mOAuthLoginInstance*.requestApi(*mContext*, at, url);  }   protected void onPostExecute(String content) {  System.*out*.println("====================== content : "+content);  System.*out*.println("====================== type : "+content.getClass());  final Intent intent = new Intent(SignInActivity.this, NaverLogin.class);  Gson gson = new Gson();  Map<String,Object> responseInfo = gson.fromJson(content, Map.class);  System.*out*.println("================================= a-b : "+(Map<String, Object>)responseInfo.get("response"));  Map<String, Object> responseUserInfo=(Map<String, Object>)responseInfo.get("response");  System.*out*.println("============== 정답 : " + (String)responseUserInfo.get("id"));   intent.putExtra("id", (String)responseUserInfo.get("id"));  startActivity(intent);  }  }  HttpClient 를 이용하여 JSON 형식으로 정보를 주고 받고, Gson 라이브러리를 이용하여 json형식의 파일을 파싱합니다.  웹과 연결은 동기화 방식(Thread)형식으로 진행할 시 변수에 값이 할당되기 전에 다음 메서드가 실행 될 수 있기 때문에 AsyncTask를 통해서 비동기화방식으로 진행하였습니다.  로그인이 성공하면 public static 형식의 UserVO에 값을 저장하고 로그인 상태를 유지하도록 구성했습니다.  네이버와 카카오 로그인 연동도 같이 구성되어, 웹뿐만아니라 안드로이드에서도 간편 로그인을 할 수 있습니다.  또한 | |
| activity | com.example.lifecare.login.NaverLogin |
| public class NaverLogin extends AppCompatActivity {  String id;  UserVO userVO = UserVO.*getInstance*();   @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  Intent intent = getIntent();   id = intent.getExtras().getString("id");  }   @Override  protected void onStart() {  super.onStart();  InnerTask task = new InnerTask();  Map<String, String> map = new HashMap<>();  map.put("naverId", id);   task.execute(map);  }   public class InnerTask extends AsyncTask<Map, Integer, String> {   //doInBackground 실행되기 이전에 동작  @Override  protected void onPreExecute() {  super.onPreExecute();  }   //작업을 쓰레드로 처리  @Override  protected String doInBackground(Map... maps) {  //HTTP 요청 준비 - Post 로 바꿀것  HttpClient.Builder http = new HttpClient.Builder("POST", Web.*servletURL* + "naverSignIn"); //스프링 url  //파라미터 전송  http.addAllParameters(maps[0]);  System.*out*.println("http ㅁㄴㅇㄻㄴㅇㄹ: " + http.getParameter());  //HTTP 요청 전송  HttpClient post = http.create();  post.request();   String body = post.getBody(); //Web의 Controller에서 리턴한 값  return body;  }   //doInBackground 종료되면 동작  */\*\*  \** ***@param*** *s : doInBackground에서 리턴한 body. JSON 데이터  \*/* @Override  protected void onPostExecute(String s) {  Log.*d*("JSON\_RESULT ㅁㄴㅇㄹ", s);  SharedPreferenceHandler sh = new SharedPreferenceHandler(getApplicationContext());  //JSON으로 받은 데이터를 VO Obejct로 바꿔준다.  if(s.length() > 0) {  Gson gson = new Gson();  UserVO m = gson.fromJson(s, UserVO.class);  if (m.getId() != null && m.getId() != "") {  // 페이지 이동  Intent intent = new Intent(NaverLogin.this, MainActivity.class);   //로그인 유지  userVO.setId(m.getId());  userVO.setEnabled(m.getEnabled());  userVO.setCustomer\_echeck(m.getCustomer\_echeck());   //아이디 저장 & 자동로그인  sh.keepId(m.getId());   StyleableToast.*makeText*(getApplicationContext(), m.getId()+"님 로그인 되었습니다.", Toast.*LENGTH\_LONG*, R.style.*mytoast*).show();  startActivity(intent);  } else if (m.getEnabled() != "1") {  Toast.*makeText*(getApplicationContext(), "회원 정보가 올바르지 않습니다.", Toast.*LENGTH\_LONG*).show();  } else {  Toast.*makeText*(getApplicationContext(), "가입 인증이 필요한 회원입니다.", Toast.*LENGTH\_LONG*).show();  }  } else {  Toast.*makeText*(getApplicationContext(), "연동되어있지 않은 아이디입니다.", Toast.*LENGTH\_SHORT*).show();  }  }  } }  앞선 SignInActivity에서 NaverLogin을 성공하면 관련 id에 특정한 key값을 반환하는데 웹과 연동하여 DB를 참조해서 key값에 해당하는 id 정보를 가져오는 역할을 하는 페이지 입니다. | |
| Activity | com.example.lifecare.login.KakaoLoginCheck |
| public class KakaoLoginCheck extends AppCompatActivity {  private SessionCallback sessionCallback = new SessionCallback();  Session session;  UserVO userVO = UserVO.*getInstance*();   @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_kakao\_login*);   session = Session.*getCurrentSession*();  session.addCallback(sessionCallback);   session.open(AuthType.*KAKAO\_LOGIN\_ALL*, KakaoLoginCheck.this);   try {  Thread.*sleep*(1000);  } catch (InterruptedException e) {  e.printStackTrace();  }   Intent intent = new Intent(KakaoLoginCheck.this, KakaoLoginCheck2.class);  startActivity(intent);  }   @Override  protected void onDestroy() {  super.onDestroy();   // 세션 콜백 삭제  Session.*getCurrentSession*().removeCallback(sessionCallback);  }   @Override  protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {  // 카카오톡|스토리 간편로그인 실행 결과를 받아서 SDK로 전달  if (Session.*getCurrentSession*().handleActivityResult(requestCode, resultCode, data)) {  return;  }   super.onActivityResult(requestCode, resultCode, data);  } }  kakao 어플과 연동을 통해서 로그인을 진행하고, 로그인 성공시 token을 받아오는 역할을 합니다. | |
| Activity | com.example.lifecare.login.kakoLoginCheck2 |
| public class KakaoLoginCheck2 extends AppCompatActivity {  UserVO userVO = UserVO.*getInstance*();   @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_kakao\_login*);   System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  if(userVO.getId() != ""){  KakaoLoginCheck2.InnerTask task = new KakaoLoginCheck2.InnerTask();  Map<String, String> map = new HashMap<>();   map.put("kakaoId",userVO.getId());   task.execute(map);  }else{  Intent intent = new Intent(KakaoLoginCheck2.this, SignInActivity.class);  StyleableToast.*makeText*(getApplicationContext(), "카카오로그인에 실패하였습니다.", Toast.*LENGTH\_LONG*, R.style.*mytoast*).show();  startActivity(intent);  }  }   //각 Activity 마다 Task 작성  public class InnerTask extends AsyncTask<Map, Integer, String> {   //doInBackground 실행되기 이전에 동작  @Override  protected void onPreExecute() {  super.onPreExecute();  }   //작업을 쓰레드로 처리  @Override  protected String doInBackground(Map... maps) {  //HTTP 요청 준비 - Post 로 바꿀것  HttpClient.Builder http = new HttpClient.Builder("POST", Web.*servletURL* + "kakaoSignIn"); //스프링 url  //파라미터 전송  http.addAllParameters(maps[0]);  System.*out*.println("http ㅁㄴㅇㄻㄴㅇㄹ: " + http.getParameter());  //HTTP 요청 전송  HttpClient post = http.create();  post.request();   String body = post.getBody(); //Web의 Controller에서 리턴한 값  return body;  }   //doInBackground 종료되면 동작  */\*\*  \** ***@param*** *s : doInBackground에서 리턴한 body. JSON 데이터  \*/* @Override  protected void onPostExecute(String s) {  Log.*d*("JSON\_RESULT ㅁㄴㅇㄹ", s);  SharedPreferenceHandler sh = new SharedPreferenceHandler(getApplicationContext());  //JSON으로 받은 데이터를 VO Obejct로 바꿔준다.  if(s.length() > 0) {  Gson gson = new Gson();  UserVO m = gson.fromJson(s, UserVO.class);  if (m.getId() != null && m.getId() != "") {  // 페이지 이동  Intent intent = new Intent(KakaoLoginCheck2.this, MainActivity.class);   //로그인 유지 // userVO.setId(m.getId()); // userVO.setEnabled(m.getEnabled()); // userVO.setCustomer\_echeck(m.getCustomer\_echeck());   //아이디 저장 & 자동로그인  //sh.keepId(m.getId());   StyleableToast.*makeText*(getApplicationContext(), m.getId()+"님 로그인 되었습니다.", Toast.*LENGTH\_LONG*, R.style.*mytoast*).show();  startActivity(intent);  } else if (m.getEnabled() != "1") {  Intent intent = new Intent(KakaoLoginCheck2.this, SignInActivity.class);  Toast.*makeText*(getApplicationContext(), "회원 정보가 올바르지 않습니다.", Toast.*LENGTH\_LONG*).show();  startActivity(intent);  } else {  Intent intent = new Intent(KakaoLoginCheck2.this, SignInActivity.class);  Toast.*makeText*(getApplicationContext(), "가입 인증이 필요한 회원입니다.", Toast.*LENGTH\_LONG*).show();  startActivity(intent);  }  } else {  Intent intent = new Intent(KakaoLoginCheck2.this, SignInActivity.class);  Toast.*makeText*(getApplicationContext(), "카카오에 연동되어 있는 아이디가 없습니다.", Toast.*LENGTH\_SHORT*).show();  startActivity(intent);  }  }  } }  kakao 어플에서 로그인 성공시 받아오는 key값을 가지고 웹과 연동하여 DB를 참조하고 해당 id 정보를 가져오는 페이지 입니다. | |
| service | com.example.lifecare.EclipseConnect.SharedPreferenceHandler |
| public class SharedPreferenceHandler {  private final String PREF\_NAME = "com.example.lifecare";  static Context *mContext*;   public SharedPreferenceHandler(Context c) {  *mContext* = c;  }   public void keepId(String id){  SharedPreferences pref = *mContext*.getSharedPreferences(PREF\_NAME, Activity.*MODE\_PRIVATE*);  SharedPreferences.Editor editor = pref.edit();  editor.putString("id", id);  editor.commit();  System.*out*.println("keepId 완료");  }   public String getId() {  SharedPreferences pref = *mContext*.getSharedPreferences(PREF\_NAME, Activity.*MODE\_PRIVATE*);   System.*out*.println("getId 진입");  try {  return pref.getString("id", null);  } catch (Exception e) {  return null;  }  }   public void clear(){  SharedPreferences pref = *mContext*.getSharedPreferences(PREF\_NAME, Activity.*MODE\_PRIVATE*);  SharedPreferences.Editor editor = pref.edit();  editor.clear();  editor.commit();  } }  SharedPreferenceHandler를 이용해서 아이디 저장 클릭 시 다음 로그인 때 아이디를 저장할 수 있도록 구현. SharedPreference는 어플을 지우기 전까지 저장되는 저장 공간이기 때문에 어플을 종료해도 그 데이터는 유지된다. | |