|  |  |
| --- | --- |
|  |  |
| 프로보노 ICT멘토링 프로젝트  결과보고서 | |
|  |  |

|  |  |
| --- | --- |
| 프로젝트명 : | 지하철 배려석 효과적 운영관리 시스템 |

|  |  |  |  |
| --- | --- | --- | --- |
| 프로젝트 참여자 | | | |
| **멘 토** | 김 성 훈 |  | |
| 멘 티 | 김 민 성 | 안양대학교 | 컴퓨터공학과 |
| 멘 티 | 김 성 표 | 안양대학교 | 컴퓨터공학과 |
| 멘 티 | 김 주 환 | 안양대학교 | 컴퓨터공학과 |
| 멘 티 | 김 태 욱 | 안양대학교 | 컴퓨터공학과 |
| 멘 티 | 이 선 경 | 안양대학교 | 컴퓨터공학과 |

|  |
| --- |
| 목 차 |

|  |
| --- |
| **Ⅰ. 프로젝트 개요** |
| 1. 프로젝트 추진배경 |
| 2. 월별 프로젝트 추진내용 |
| 3. 팀원별 역할 |
| 4. 주요 결과물 |
|  |
| **Ⅱ. 프로젝트 결과** |
| 1. 프로젝트 결과물 |
| 2. 프로젝트 기대효과 및 활용분야 |
| 3. 프로젝트 개선(보완)점 및 해결방안 |
| 4. 프로젝트 소감 |
| 5. 기타(참고 및 인용자료) |

**Ⅰ. 프로젝트 개요**

**1. 프로젝트 추진배경**

지하철 배려석(장애인, 임산부)의 실제 대상자가 앉아야 하지만,

앉아있는 사람이 대상자여부를 확인하기가 어려워 실제 운영이 잘 되고 있는지

확인이 어렵다. 또한 다른칸에 자리가 비어있어도 가서 확인하기가 어렵기 때문에

칸마다 배려석의 효율이 떨어진다.

**2. 월별 프로젝트 추진내용**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **구분** | **추진내용** | **추진일정(월)** | | | | | | |
| **5** | **6** | **7** | **8** | **9** | **10** | **11** |
| **계획** | 요구사항 계획서 작성 | **V** |  |  |  |  |  |  |
| **설계** | 수용할 기능을 기반으로  전체적인 결과물의 그림 그리기 |  | **V** | **V** |  |  |  |  |
| **개발** | 프로토타입 설정, 1차 개발 |  |  | **V** | **V** | **V** |  |  |
| **테스트** |  |  |  |  |  | **V** | **V** |  |
| **종료** |  |  |  |  |  |  |  | **V** |

**3. 팀원별 역할**

|  |  |  |  |
| --- | --- | --- | --- |
| **성명** | **전공** | **학년** | **담당역할** |
| **김민성** | 컴퓨터공학과 | 3 | 아두이노, 서버 개발 팀원 |
| **김성표** | 컴퓨터공학과 | 3 | 앱 디자인 |
| **김주환** | 컴퓨터공학과 | 3 | 앱 개발 분야 팀장 |
| **김태욱** | 컴퓨터공학과 | 3 | 서버, 앱 팀원 |
| **이선경** | 컴퓨터공학과 | 3 | 아두이노, 서버 개발 팀장 |

**4. 주요 결과물**

ㅇ 결과물 이미지

|  |
| --- |
|  |

ㅇ 소외계층의 삶의 질 향상에 기여한 점

- 배려석이 분명히 존재하나 실제로 좌석에 착석한 임산부는 보기 힘들었으나 프로젝트를 통해 조금이라도 비임산부의 착석을 줄이고, 임산부의 착석을 권장하도록 하였다.

ㅇ 주요 특징

- SW 특징

• 탑승을 희망하는 지하철의 임산부 배려좌석(아두이노)의 실시간 정보를 어플리케이션을 통해 시각적으로 정보를 확인할 수 있으며 지하철 도착시간 등 지하철에 대한 정보를 얻을 수 있다. .

- HW 특징

• 임산부 배려좌석에 설치된 아두이노 와이파이 모듈을 통해서 열차내의 와이파이와 연결되어 임산부 배려좌석에 대한 착석여부를 어플리케이션에서 확인 할 수 있도록 웹DB에 정보를 넘겨준다.

**Ⅱ. 프로젝트 결과**

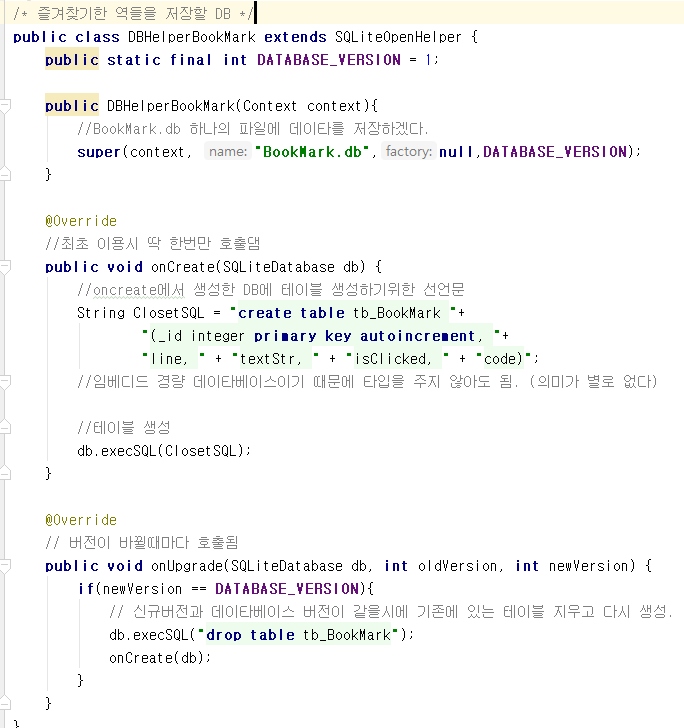
**1. 프로젝트 결과물**

ㅇ 결과물 이미지

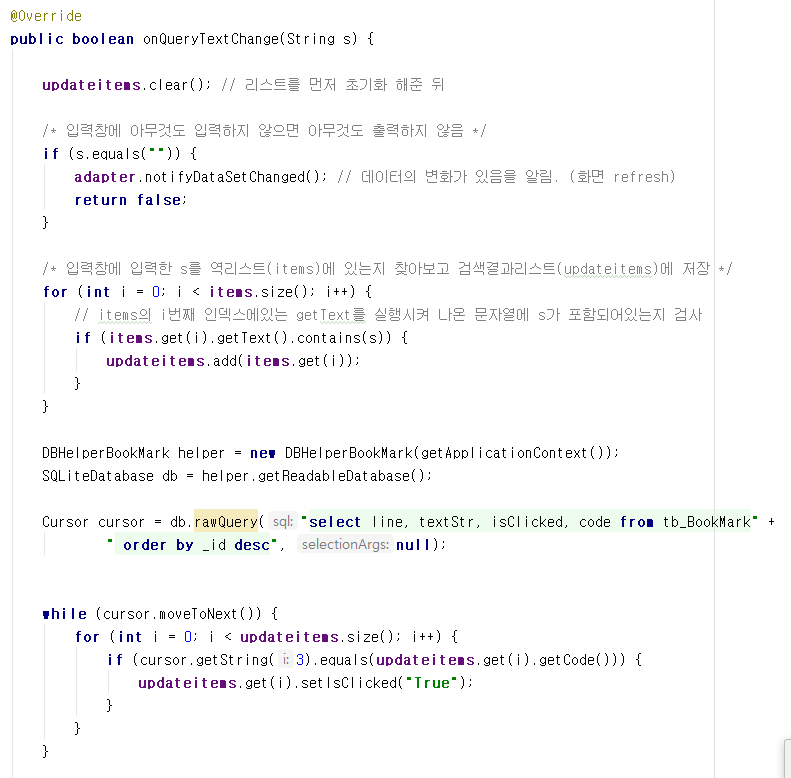
|  |
| --- |
| EMB000023942b0cEMB000023942b0d   * 메인화면 : 배려석 예약, 민원 문자 전송, 문의/긴급전화 선택 가능 * 검색기능 : 지하철 API 기반 검색 기능, 즐겨찾기 가능 |

|  |
| --- |
| EMB000023942b0eEMB000023942b0f   * 도착정보 : 지하철 실시간 도착정보 확인 가능 (시간표 기반, 초 단위) * 좌석조회 : 선택한 열차의 번호와, 실시간 좌석 공석 여부 조회 |

ㅇ 프로그램 소스



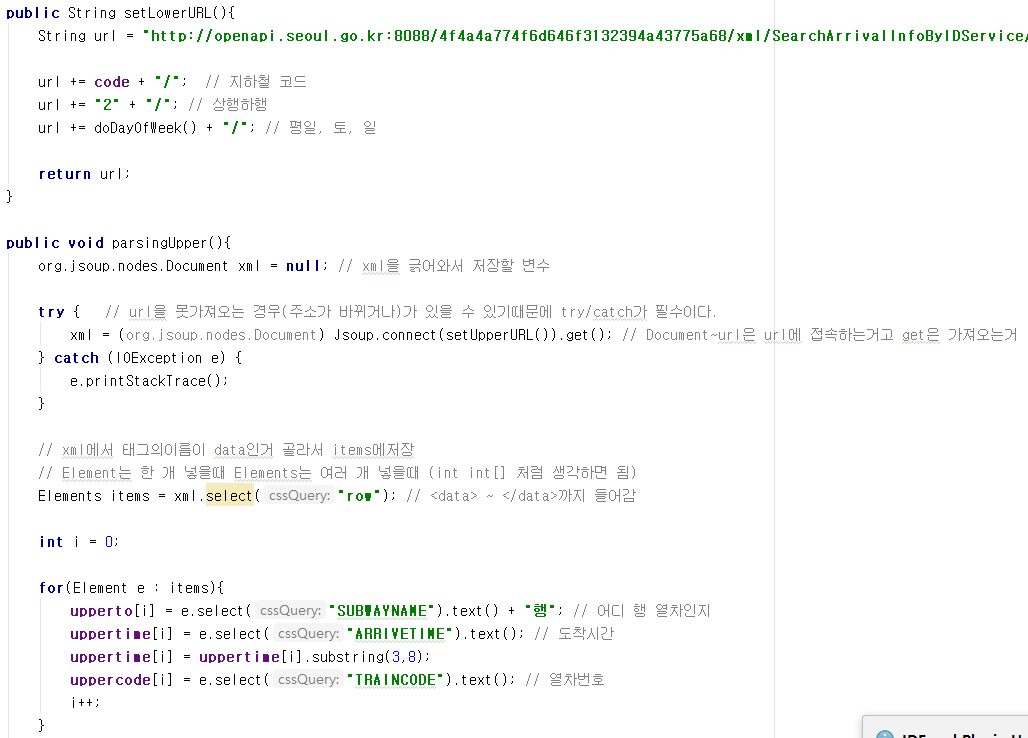
(내부 DB관리)



(DB를 기반으로한 검색)



(즐겨찾기 관리)



(인터넷 API를 통한 데이터 파싱)

|  |
| --- |
| /\*아두이노 소스 \*/  /\* WIFI \*/  #include <SoftwareSerial.h>  #include <ESP8266.h>  /\*\*\*\*\*\*\*\*/  /\* NFC \*/  #include <SPI.h>  #include <MFRC522.h>  /\*\*\*\*\*\*\*/  /\*\*\*\*\*\*\*\*\*\*\* WIFI \*\*\*\*\*\*\*\*\*\*\*\*\*/  SoftwareSerial esp8266Serial = SoftwareSerial(3, 2);  ESP8266 wifi = ESP8266(esp8266Serial);  #define ID "SILab"  #define PW "si12341234"  #define PORT 8088  #define IP "http://218.148.147.25:8088"  #define IP1 218  #define IP2 148  #define IP3 147  #define IP4 25  String URL = URL "/ArduinoTest/NFCtest.jsp";  String sendURL = "";  String data = "";    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\*\*\*\*\*\*\*\*\*\*\* NFC \*\*\*\*\*\*\*\*\*\*\*\*\*/  #define SS\_PIN 10  #define RST\_PIN 9    const byte KEYB[] = {  0xDE, 0xAD, 0xBE, 0xEB, 0xDE, 0xAD };    MFRC522 mfrc522(SS\_PIN, RST\_PIN); // MFRC522 인스턴스 생성    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void setup() {  Serial.begin(9600);    /\* WIFI \*/  esp8266Serial.begin(9600);  wifi.begin();  wifi.setTimeout(1000);    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\*\*\*\*\*\* WiFi commands \*\*\*\*\*\*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  // setWifiMode  Serial.print("setWifiMode: ");  //Serial.println(getStatus(wifi.setMode(ESP8266\_WIFI\_ACCESSPOINT)));  Serial.println(getStatus(wifi.setMode(1)));  // getWifiMode  /\*ESP8266WifiMode mode;  Serial.print("getWifiMode: ");  Serial.println(getStatus(wifi.getMode(&mode)));\*/  // joinAP  Serial.print("joinAP: ");  Serial.println(getStatus(wifi.joinAP(ID, PW)));  // getAP  /\*char ap[32] = {};  Serial.print("getAP: ");  Serial.print(getStatus(wifi.getAP(ap)));  Serial.print(" : ");  Serial.println(ap);\*/  // quitAP  /\*Serial.print("quitAP: ");  Serial.println(getStatus(wifi.quitAP()));\*/  /\*\*\*\*\*\*\*\*/  /\* NFC \*/  SPI.begin(); // SPI 버스 사용을 위한 초기화를 합니다  mfrc522.PCD\_Init(); // MFRC522를 초기화 합니다.  Serial.println("NFC READY");  /\*\*\*\*\*\*\*/  }  void loop() {  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\*\*\*\*\*\* NFC \*\*\*\*\*\*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  byte buffer[18];  byte size = sizeof(buffer);    // 새로운 카드가 접촉되었는지 확인합니다:  if ( ! mfrc522.PICC\_IsNewCardPresent()) {  return;  }    // 카드들 중 하나를 선택합니다:  if ( ! mfrc522.PICC\_ReadCardSerial()) {  return;  }    // 카드가 선택되었고, mfrc522.uid에 UID와 SAK 정보가 들어 있습니다.  // UID 정보를 16 진수 형태로 출력:  Serial.print("카드 UID:");  for (byte i = 0; i < mfrc522.uid.size; i++) {  Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");  Serial.print(mfrc522.uid.uidByte[i], HEX);  data += String(mfrc522.uid.uidByte[i], HEX);  }  data.toUpperCase();  Serial.println();  Serial.print("UID: ");  Serial.println(data);  Serial.println();  wifiServerConn();  data = "";  delay(10000);  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\*\*\*\*\*\*\*\* WIFI 함수 \*\*\*\*\*\*\*\*/  void wifiServerConn() {  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\*\*\*\*\*\* Basic commands \*\*\*\*\*\*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  // test  Serial.print("test: ");  Serial.println(getStatus(wifi.test()));  // restart  /\*Serial.print("restart: ");  Serial.println(getStatus(wifi.restart()));\*/  // getVersion  /\*  char version[16] = {};  Serial.print("getVersion: ");  Serial.print(getStatus(wifi.getVersion(version, 16)));  Serial.print(" : ");  Serial.println(version); \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\*\*\*\*\*\* TCP/IP commands \*\*\*\*\*\*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  // connect  Serial.print("connect: ");  Serial.println(getStatus(wifi.connect(ESP8266\_PROTOCOL\_TCP, IPAddress(IP1,IP2,IP3,IP4), PORT)));  getConnectionStatue(wifi);  // send  Serial.print("send: ");  sendURL = "GET " + URL + "?uid=" + data + " HTTP/1.0\r\n\r\n";  Serial.println(sendURL);  Serial.println(getStatus(wifi.send(sendURL)));  // close  Serial.print("close: ");  Serial.println(getStatus(wifi.close(1)));  }  void getConnectionStatue(ESP8266 wifi) {  ESP8266ConnectionStatus connectionStatus;  ESP8266Connection connections[5];  unsigned int connectionCount;  Serial.print("getConnectionStatus: ");  Serial.print(getStatus(wifi.getConnectionStatus(connectionStatus, connections, connectionCount)));  Serial.print(" : ");  Serial.println(connectionCount);  for (int i = 0; i < connectionCount; i++) {  Serial.print(" - Connection: ");  Serial.print(connections[i].id);  Serial.print(" - ");  Serial.print(getProtocol(connections[i].protocol));  Serial.print(" - ");  Serial.print(connections[i].ip);  Serial.print(":");  Serial.print(connections[i].port);  Serial.print(" - ");  Serial.println(getRole(connections[i].role));  }  }  String getStatus(bool status) {  if (status)  return "OK";  return "KO";  }  String getStatus(ESP8266CommandStatus status) {  switch (status) {  case ESP8266\_COMMAND\_INVALID:  return "INVALID";  break;  case ESP8266\_COMMAND\_TIMEOUT:  return "TIMEOUT";  break;  case ESP8266\_COMMAND\_OK:  return "OK";  break;  case ESP8266\_COMMAND\_NO\_CHANGE:  return "NO CHANGE";  break;  case ESP8266\_COMMAND\_ERROR:  return "ERROR";  break;  case ESP8266\_COMMAND\_NO\_LINK:  return "NO LINK";  break;  case ESP8266\_COMMAND\_TOO\_LONG:  return "TOO LONG";  break;  case ESP8266\_COMMAND\_FAIL:  return "FAIL";  break;  default:  return "UNKNOWN COMMAND STATUS";  break;  }  }  String getRole(ESP8266Role role) {  switch (role) {  case ESP8266\_ROLE\_CLIENT:  return "CLIENT";  break;  case ESP8266\_ROLE\_SERVER:  return "SERVER";  break;  default:  return "UNKNOWN ROLE";  break;  }  }  String getProtocol(ESP8266Protocol protocol) {  switch (protocol) {  case ESP8266\_PROTOCOL\_TCP:  return "TCP";  break;  case ESP8266\_PROTOCOL\_UDP:  return "UDP";  break;  default:  return "UNKNOWN PROTOCOL";  break;  }  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |

|  |
| --- |
| <!-- NFC INSERT -->  <%@ page contentType="text/html; charset=UTF-8"%>  <%@ page import="java.sql.\*, java.util.\*, java.text.\*"%>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  <title>NFCtest</title>  </head>  <body>  <%  request.setCharacterEncoding("UTF-8");  String uid = request.getParameter("uid");  String sno = "회원고유번호";  Connection conn = null;  PreparedStatement pstmt = null;  try {  String jdbcUrl = "jdbc:oracle:thin:@218.148.147.25:1521:xe";  String dbId = "hr2";  String dbPass = "1234";  Class.forName("oracle.jdbc.driver.OracleDriver");  conn = DriverManager.getConnection(jdbcUrl, dbId, dbPass);  String sql = "INSERT INTO NFC VALUES(?, ?, ?)";  pstmt = conn.prepareStatement(sql);  java.text.SimpleDateFormat formatter = new java.text.SimpleDateFormat("yyyyMMddHHmmss");  String date = formatter.format(new java.util.Date());  pstmt.setString(1, uid);  pstmt.setString(2, sno);  pstmt.setString(3, date);  pstmt.executeUpdate();  %>  <%=uid%><p>  <%=sno%>  <%=date%>  <%  } catch (Exception e) {  e.printStackTrace();  } finally {  if (pstmt != null)  try {  pstmt.close();  } catch (SQLException sqle) {  }  if (conn != null)  try {  conn.close();  } catch (SQLException sqle) {  }  }  %>  </body>  </html> |

ㅇ 프로젝트 결과 상세내용

개발 툴

- Android Studio 3.1.3 (JRE 1.8.0\_152-release)

- Oracle Database 11g Release 2

- Eclipse Jee Oxygen

- Arduino IDE 1.8.7

하드웨어

- Arduino UNO

- 갤럭시S7 엣지 (Android 8.0.0 오레오)

**2. 프로젝트 기대효과 및 활용분야**

ㅇ 기대효과

∙ 인식표를 매번 들고 다닐 필요 없이 스마트폰만 있으면 임산부 좌석을

보다 편리하게 이용할 수 있다.

∙ 임산부 좌석을 무조건 적으로 비어두어 좌석의 효율을 낮출 필요 없이

앱과 아두이노를 통해 주변에 임산부가 있는지 체크하여 일반인도 앉을 수 있어

좌석의 효율을 높일 수 있다.

∙ 임산부/장애인 벼러석 관련 조회 기능을 기반으로 지하철의 좌석 관련한 플랫폼 확보가 가능 하다.

이를 기반으로 버스로도 확대할 수 있다.

ㅇ 활용분야

- 지정된 사용자에게 좌석을 제공하는 인증 시스템으로 활용 가능하다.

**3. 프로젝트 개선(보완)점 및 해결방안**

ㅇ 프로젝트 개선(보완)점

- 처음에 휴대폰 NFC에 있는 고유 UID값을 사용하려고 했으나 개발을 시작하고 테스트를 하던 중 UID값이 매번 마다 바뀐다는 것을 알게 되었는데, 고유한 값을 찾아내지 못해 NFC 활용을 하지 못했다.

ㅇ 해결방안

- 회원마다 저희가 임의로 특정한 값을 부여하는 방식으로 해결하였다.

**4. 프로젝트 소감**

ㅇ 프로젝트에 참여하기 전에는 임산부 배려석에 대해 평소 인식을 하지 못하였는데, 이번 기회를 통하여 전철을 이용할 때 임산부 배려석이 비어져 있는지, 임산부가 앉아 있는지, 임산부가 아닌 사람이 앉아 있는지 주의깊게 보게 되었다.

**5. 기타(참고 및 인용자료)**

http://cocomo.tistory.com/409

SQLite DB 설명 블로그3

http://www.u2iot.com/bbs/board.php?bo\_table=project&wr\_id=76&sca=%ED%97%AC%EC%8A%A4%EC%BC%80%EC%96%B4

NFC에 대한 정보

<http://blog.naver.com/PostView.nhn?blogId=n_jihyeon&logNo=221196820428>

searchview에 대한 정보