드리밍텍(주) 인턴 황인원 ‘나폴나폴’ 어플리케이션 개발일지

<모바일 폴라로이드 감성의 이미지와 텍스트를 같이 저장하기>

목 차

1. 개발 목적………………………………………………………………………………………………………………2

2. 어플리케이션 이름 및 선정 이유…………………………………………………………………………2

3. 어플리케이션 개발에 대한 정보

1. 핵심 기능 정리……………………………………………………………………………………….2
2. 필수 개발 환경……………………………………………………………………………………….2
3. 어플리케이션 개발 일정 표…………………………………………………………………...2
4. Android Studio 개발 주요 파일 구성 및 상호 관계……………………………2
5. 기본 프로젝트 생성 및 기능 배치………………………………………………………….2
6. 각 구성 별 경로 및 전체 코드………………………………………………………………..2

4. 개발 단계

1. 카메라 기능 구현
2. 개발 목적

개발 목적 : (이미지 + 텍스트) 저장 및 관리용 어플리케이션 개발

2.어플리케이션 이름 및 선정이유

|  |  |
| --- | --- |
| **어플리케이션 이름** | 나폴나폴 |
| **어플리케이션 이름 선정 이유** | 현재 어플리케이션의 기능인 이미지와 텍스트의 결합 자체를 아날로그 감성인 폴라로이드 사진의 기능과 흡사하게 구현 할 것이어서 ‘나만의 폴라로이드’라고 정해 주었다. 이렇게 생각 된 이름에서 줄여 ‘나폴나폴’ 이라는 이름을 정해 주었다.. |

3.어플리케이션 개발에 대한 정보

가. 핵심 기능 정리

본 어플리케이션의 핵심적인 기능으로는 이미지 에 대한 경로 파악 및 활용 기능, 데이터 저장 및 사용 기능 이 있다.

|  |  |
| --- | --- |
| **=>이미지에 대한 경로 파악 및 활용 기능** | 가장 중요한 기능인 이미지 활용 기능을 구현 하기 위해 해줘야 할 것이 이미지에 대한 uri 정보 이용 방법 이다.  Uri란 Uniform Resource Identifier의 준말로써 통합자원 식별자라 불리우는 인테넷에 있는 자원을 나타내는 유일한 주소이다. 하지만 이 식별자는 어떠한 매체를 인식하기 위한 수단으로도 사용 되어질 수도 있다. 따라서 이 uri 정보를 이용해 이미지를 저장한 후 그 uri 정보를 데이터 베이스에 저장하게 된다면 다시 불러와 사용 하고자 할 때 uri 정보를 이용해 해당 주소에 있는 값 또는 이미지를 불러 올 수 있다. |
| **=>데이터 통합 저장 및 사용 기능** | 데이터를 저장 하고 사용 하는 기능은 어느 어플리케이션 에서나 중요하다. 특히 이 어플리케이션의 목적에 따르면 이미지와 데이터를 같이 저장하기 위해 필수적으로 필요한 이다. 따라서 데이터 베이스는 안드로이드 스튜디오가 지원하는 SQLite를 사용한다. |

나. 필수 개발 환경

안드로이드 스튜디오를 설치 및 구동 하기 위해 먼저 구축 되어야 하는 환경에 대한 설명

|  |  |
| --- | --- |
| **개발 환경** | **환경에 대한 설명** |
| JDK | 자바 개발을 위한 도구이다.  자바컴파일러(javac), 자바 가상머신(jvm), 자바 실행 환경(jre), 각종 library를 포함하고 있다. |

|  |  |  |
| --- | --- | --- |
| **단계** | **환경 구축 스크린샷** | **환경 구축에 대한 설명** |
| **a** | 오라클 홈페이지 : <http://www.oracle.com/index.html> | JDK 설치파일을 다운로드 하기 위해 오라클 홈페이지에 접속한다 |
| **b** | https://t1.daumcdn.net/cfile/tistory/21441E475729537329 | 오라클 홈페이지에 접속 후 Downloads 메뉴를 클릭하여 java SE 탭을 누른다. |
| **c** | https://t1.daumcdn.net/cfile/tistory/223FF9415729513215 | Java SE 탭으로 들어가게 되면 다운로드 항목들 중에 JDK를 선택힌다. |
| **d** | https://t1.daumcdn.net/cfile/tistory/224DBD47572954232A | JDK항목을 선택한 후 다운로드 화면으로 들어오게 되면 나타나게 되는 창이다. 다운 받기 전에 Accept License Agreement 를 체크해 주어야 다운로드가 가능해 진다. 그 다음 본인의 운영체제에 맞는 것을 선택해 다운로드 해주면 된다. Mac 을 사용 하기때문에 Mac os 버전으로 선택해 다운로드 한다. |
| **e** | https://t1.daumcdn.net/cfile/tistory/273A23455729555B27 | 다운로드가 완료되면 왼쪽과 같은 화면이 뜬다. 그리고 나서 아이콘을 더블클릭 하면 설치가 시작 된다. |
| **f** | https://t1.daumcdn.net/cfile/tistory/264D9D425729558435 | 화면을 따라서 계속 설치를 진행 해주면 된다. |
| **g** | https://t1.daumcdn.net/cfile/tistory/23068340572955F31F | 설치가 종료 된 화면이다. |
| **h** |  | 설치를 확인하기 위한 단계이다.  터미널 프로그램을 실행하여 프롬프트 라인에  cd /Library/Java/JavaVirtualMachines/ 를 작성해주고 엔터를 친다.  그렇게 되면 위에 cd를 제외한 코드를 따라 그 경로로 이동하게 된다. |
| **i** |  | 경로로 이동 된 상태에서 ls 명령어를 쳐준다.  명령어를 쳤을 때 jdk-?.jdk 형식의 파일이 뜰 경우 jdk 설치가 완료 된 것이다. |

위의 JDK 환경을 구축하게 되면 아래의 두 환경도 같이 설치가 되어진다.

|  |  |
| --- | --- |
| **개발 환경** | **개발 환경에 대한 설명** |
| JRE : 1.8.0\_152-release-1024-b01 x86\_64 | 자바 실행환경(Java Runtime Environment)  JVM이 자바 프로그램을 동작 시킬 때 필요한 라이브러리 파일들과 기타 파일들을 가짐 즉, JRE 는 JVM의 실행환경을 구현 |
| JVM : OpenJDK 64-Bit Server VM by JetBrains s.r.o | 자바 가상머신(Java Virtual Machine)  JVM은 자바 바이너리 파일(.class)를 실행 할 수 있다 |

안드로이드 스튜디오 설치과정에 대한 설명

|  |  |
| --- | --- |
| **개발 환경** | **개발 환경에 대한 설명** |
| Androi한 Studio 3.1.3 | 실제 개발을 진행 하기 위한 툴(코드를 입력 받는 공간) |

|  |  |  |
| --- | --- | --- |
| **단계** | **개발 환경 구축 스크린샷** | **개발 환경 구축에 대한 설명** |
| **a** | 안드로이드스튜디오 다운로드 : <http://developer.android.com/sdk/index.html> | 링크를 통해 안드로이드 스튜디오 다운로드 페이지로 연결한다. |
| **b** | https://t1.daumcdn.net/cfile/tistory/260EA2485729683328 | 웹페이지에 접속 하게 되면 바로 설치가 가능한 페이지로 연결 된다. |
| **c** | https://t1.daumcdn.net/cfile/tistory/234FA84C5729686841 | 다운로드 버튼을 눌러 다운로드 페이지로 접속하게 되면 왼쪽과 같은 화면이 뜨게 된다. 화면 내에서 동의 버튼을 클릭하고 다운로드를 시작 하면 된다. |
| **d** | https://t1.daumcdn.net/cfile/tistory/2353C43757296A0834 | 파일 다운로드가 끝나게 되면 다운로드 한 파일을 더블 클릭 하면 된다. 파일을 열게 되면 왼쪽과 같은 화면 이 뜨는데 화면 내에서 안드로이드 스튜디오 이미지를 application 폴더 이미지로 드래그 해주면 다운로드 단계가 완료 된다. |
| **e** | https://t1.daumcdn.net/cfile/tistory/264BE24C57296A6E1C | 설치된 프로그램을 실행 하면 왼쪽과 같은 화면이 뜨게 된다.  열기를 눌러 진행 하면 된다. |
| **f** | https://t1.daumcdn.net/cfile/tistory/2744F94E57296AB823 | 이전 버전의 설정을 import 하는 것을 묻는 것이지만 처음 하는 설치이므로 import하지 않음을 선택해 준다. |
| **g** | https://t1.daumcdn.net/cfile/tistory/2576484E57296B5C1A | 본인만의 설정으로 설치를 진행 해주면 된다. |
| **h** | https://t1.daumcdn.net/cfile/tistory/260D6036572972A716 | 계속 진행을 하여 설치를 끝내고 나면 왼쪽과 같은 화면이 뜨게 되고 finish를 눌러 설치를 종료 시켜주면 된다. |
| **i** | **안드로이드 스튜디오가 설치 된 후 SDK를 설치 해 주어야 한다.** | SDK는 소프트웨어 개발 키트의 줄임말 이다.  일반적으로 소프트웨어 기술자가 사용하여 특정한 소프트웨어, 소프트웨어 프레임 워크 , 하드웨어 플랫폼, 컴퓨터 시스템, 게임, 운영체제등을 위한 응용 프로그램을 만들 수 있게 해주는 개발 도구의 집합이다. |
| **j** | https://t1.daumcdn.net/cfile/tistory/222993375729812832 | 안드로이드 스튜디오 아이콘을 클릭하여 뜨는 화면에서 오른쪽 아래의 configure 메뉴를 이용 하여 sdk 설정을 해줄 수 있다.  SDK Manager 버튼을 클릭한다. |
| **k** |  | 필요한 sdk 의 경우 체크로 선택을 하여 설치를 진행 해주면 된다. |

데이터베이스 환경

|  |  |
| --- | --- |
| **개발 환경** | **개발 환경에 대한 설명** |
| Database : SQLite | 안드로이드 스튜디오 설치 시 기본적으로 설치 되어지는 데이터 베이스 이다.  데이터 베이스 관리 시스템으로서 서버가 아닌 응용 프로그램에 넣어 사용하는 비교적 가벼운 데이터 베이스 이다. |

<데이터 베이스 초안>

|  |  |  |
| --- | --- | --- |
| **데이터 이름** | **데이터 형식** | **데이터 설명** |
| **Id** | INTEGER  (AUTO\_INCREMENT) | 자동적으로 데이터 행이 한 줄씩 생성 될 때 마다 증가하는 숫자를 부여하는 기능이 바로 AUTO INCREMENT 기능 이다. 자동적으로 숫자가 증가되어 부여되기에 고유한 값이 된다. |
| **tags1** | VARCHAR | VARCHAR은 가변 길이의 문자열 데이터로서 문자열의 길이는 1에서부터 8000사이의 값이 되어 질 수 있다. 글을 저장 하고 보여주기 위해 지정 해놓은 데이터 타입이다. |
| **tags2** |
| **tags3** |
| **Uri(Uniform Resource Identifier,**  **통합 자원 식별자)** | VARCHAR | Uri 정보는 이미지를 불러오기 위한 현재 저장소 위치에 대한 정보 이다. 현재 어플리케이션에서 저장소의 위치는 안드로이드 내의 image(경로 : ./image) 폴더이다. |

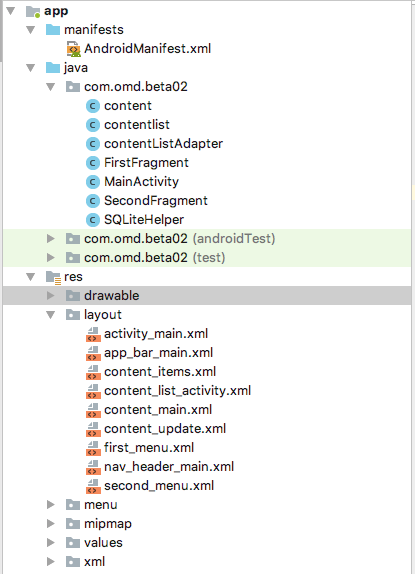
다.어플리케이션 구현 일정 표

|  |  |  |  |
| --- | --- | --- | --- |
| **일정 표 내에서 Percentage 측정 기준** | | | |
| 100% | 80% | 50% | 0% |
| Function Normal operation | Function error | Inability to Implement Function | Do not start yet |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tasks** | **W1** | **W2** | **W3** | **W4** | **W5** | **W6** | **W7** | **W8** | **W9** | **Percentage** |
| **<Initial process of development>** | 1 | 2 |  |  |  |  |  |  |  | 100% |
| 개발 진행에 앞서 먼저 데이터 분석 및 아이디어 구상을 진행 하기 위한 단계 이다.(오재우 학생과 같이 진행) | | | | | | | | | | |
| **The public data analysis** | 1 |  |  |  |  |  |  |  |  | 100% |
| 대표님의 말씀에 따라 공공데이터를 활용 하기 위한 공공 데이터 분석을 진행하였다.(오재우 학생과 같이 진행) | | | | | | | | | | |
| **Think session with teammate** | 1 |  |  |  |  |  |  |  |  | 100% |
| 오재우 학생과 데이터 분석 방법과 분석 된 데이터 활용 방안에 대한 토의를 진행 하였다. | | | | | | | | | | |
| **Idea selection in list** | 1 | 2 |  |  |  |  |  |  |  | 100% |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **어플리케이션 이름** | **구상 목적** | **적용 기술** | **개발 입장** | **강점 및 약점** | | **Sellog (Application)** | 목적: 1인 수공예품 제작자들의 판매 및 전시 기회 제공  Who? 1인 수공예품 판매 또는 구매자  When? 수공예품을 판매, 구매, 전시, 관람하고 싶을 때  Where? 온라인 어플리케이션 상에서  What? 판매자가 직접 만든 수공예품(악세사리, 양초, 비누 등)  How? 온라인 등록을 통해  Why? 1인 수공예품 제작자들의 활발한 활동을 위해 | 기술: 구글 로그인 / 결제 시스템 / 블로그형식의 탭 / 관리 시스템 구축 / 서버 구축 / DB 구축 / 안드로이드 스튜디오 사용 / 사진 기술 | 판매자 입장: 개인의 블로그 형식의 탭을 관리하여 어플리케이션을 사용하는 사용자들에게 자신의 작품을 보여주고, 그 작품을 판매하고자 한다면 판매를 등록하여 판매에 대한 자세한 정보를 적어준다.  구매자, 관람자 입장: 구매자는 본인이 직접 원하는 태그를 검색하여 원하는 상품을 찾는 방법과, 전시회라는 특정한 탭을 통하여 무작위로 보여지는 태그에 대해 관심을 갖게 된다면 그 전시회의 사진을 통해 그 판매자와 작품에 접근할 수 있도록 한다. | 강점: 경쟁자가 적다(현재 관련 어플리케이션이 1개 있다)  약점: 이미 존재하는 어플리케이션 다운로드수가 1000건 정도로 시장 규모가 쉽게 성장하지 못할 것으로 추측된다.  약점 극복 방안: 현재 다른 매체들(블로그, 인스타그램 등)이 사용되고 있어 쉽게 시장 규모가 성장하지 못할 것으로 예상되지만, 판매자 입장을 고려하여 새로운 구매자의 진입을 유도하는 방식으로 양쪽이 서로 점점 증가시키는 방향으로 성장시킬 수 있다. | | **문화 토론 (Application)** | 목적: 문화 생활에 대한 지적 대화 및 공유  Who? 특정 도서나 영상을 본 사람  When? 작품에 대한 감상을 나누고 싶을 때  Where? 온라인 어플리케이션 상에서  What? 내가 적는 수기 또는 그 작품에 대한 해석  How? 자유로운 글 게시를 통해  Why? 누구나 자신이 보거나 읽은 것에 대해 이야기 나누는 것을 선호하고 그에 따른 지식 확장도 가능하기 때문에 | 기술: 카카오톡 로그인 / 구글 로그인 / 서버 구축 / DB 구축 / sns형식의 글과 댓글 시스템 / 안드로이드 스튜디오 사용 |  | 강점: 문화 생활 장려 가능 / 감상 공유  약점: 비슷한 종류의 소프트웨어가 자리 잡고 있음 | | **Customized Diary (Application)** | 목적: 각자의 취향에 따라 필요한 기능을 선택할 수 있는 커스터마이징 플래너  Who? 개인적으로 설정한 다이어리를 사용하고 싶은 사용자  When? 언제나  Where? 어디서든  What? 다이어리 또는 플래너  How? 혼자만의 설정으로  Why? 사람마다 편안하게 느끼는 것과 필요하다 느끼는 것이 다르므로 | 기술: DB 구축 / option 기능 (주요기능 결정) / 입출력 시스템 / 안드로이드 스튜디오 |  | 강점: 본인의 취향에 맞춰 사용이 용이하다  약점: 기존의 어플리케이션들이 다양하다 | | **Korea computer yard (Web)** | 목적: 한국에서 컴퓨터 분야를 공부했거나 하고자 하는 사람들의 지식 공유  Who? 한 분야의(컴퓨터) 전공자들 또는 그 분야를 궁금해하는 어느 누구라도  When? 궁금한 점을 묻고 싶을 때  Where? 이곳에  What? 질문을  How? 업로드 하고 답변 가능 시간 여부를 얻는다  Why? 사용자 마다 요구하는 해결 시간이 다르기 때문에 | 기술: 서버구축, DB 구축 / 주민등록 번호 회원가입(개인 신원 철저히 확인) / 웹 개발 / 안드로이드 스튜디오 개발 | 답변자, 전문가: 본인이 확실히 대답할 수 있는 분야의 소분류 카테고리를 설정하여 놓고 현재 대기 상태 일 시 그에 따른 행동을 취한 후 대기  질문자, 비전문가: 본인이 묻고자 하는 질문이 대략적으로 어느 분야인지 알고 그 분야에 대한 질문을 하였을 때, 그 분류에 대해 답해 줄 수 있는 답변자와 매칭이 되어서 답변을 받고 그 답변에 대한 평가를 내릴 수 있도록 함 | 강점: 한국 내에서 컴퓨터 관련 전문 지식을 공유하고 전달하는데 있어서 편리하다  약점: 한국에 국한된 지식 | | **출근길 이것만 (Application)** | 목적: 출퇴근, 등하교길에 단 하나의 어플리케이션 만으로 사용자가 원하는 모든 정보(날씨, 교통, 일정, 관심 뉴스 등)를 보여줌  Who? 직장인, 학생  When? 출퇴근, 등하교 시간  Where? 출퇴근길, 등하교길  What? 출퇴근, 등하교에 관련된 정보 + 문서 viewer  How? 한눈에 보기 쉽게  Why? 어플리케이션 하나로 원하는 정보를 한번에 보여주기 위해 | 기술: 서버 연동, crawling, 문서 viewing, 안드로이드 스튜디오 |  | 강점: 편리하다  약점: 필요성이 확실 하지 않다 | | | | | | | | | | | |
| **<main development process>** |  | 2 | 3 | 4 | 5 | 6 | 7 | 8 |  |  |
| 새로 선정된 나폴나폴 어플리케이션에 대한 개발 진행 과정  새로 선정 된 아이디어는 개인 개발 진행 | | | | | | | | | | |
| **Main page development** |  | 2 | 3 | 4 | 5 | 6 | 7 |  |  | 50% |
| 어플리케이션 구동을 위한 메인 페이지 제작/ 메인 페이지 기능 구현 | | | | | | | | | | |
| **Google login development** |  | 2 |  |  |  |  |  |  |  | 100% |
| 파이어베이스 기능을 이용한 구글 로그인 구현 | | | | | | | | | | |
| **Search plan for usage of weather data** |  |  |  |  |  |  |  |  |  | Cancel function |
| **Development of weather window by using weather data** |  |  |  |  |  |  |  |  |  |
| 공공데이터 개발 취소/ 공공데이터는 현재 어플리케이션의 목적과 맞지 않기 때문에 기능 구현 취소 | | | | | | | | | | |
| **Basic tool development(page information & data entering tool)** |  |  | 3 | 4 |  |  |  |  |  | 80% |
| 페이지에 넣어줄 데이터의 정보를 입력 받을 기능을 구현 | | | | | | | | | | |
| **Implement camera and gallery interlocking function** |  |  |  | 4 | 5 |  |  |  |  | 80% |
| 이미지를 이용하기 위한 카메라 촬영 기능 / 촬영 된 이미지 저장 기능/ 갤러리로부터 이미지 불러오기 기능 구현 | | | | | | | | | | |
| **Text record process development** |  |  | 3 | 4 | 5 | 6 |  |  |  | 100% |
| 이미지와 함께 저장 해 줄 텍스트 를 입력 받고 저장 해주기 위한 기능 구현 | | | | | | | | | | |
| **Database Development** |  |  | 3 | 4 | 5 | 6 | 7 |  |  | 80% |
| 데이터베이스를 통한 데이터 저장 및 변경 기능 구현  안드로이드 기본 탑재 데이터 베이스인 SQLite 사용 | | | | | | | | | | |
| **Supplementation and modification about beta application** |  |  | 3 | 4 | 5 | 6 | 7 |  |  | 50% |
| 구현된 어플리케이션 코드를 수정 및 보완 작업 | | | | | | | | | | |
| **Design process** |  |  |  |  |  |  |  | 8 |  | 0% |
| 완성된 어플리케이션 디자인 | | | | | | | | | | |
| **<last process of development>** |  |  |  |  |  |  |  | 8 | 9 |  |
| 개발 과정 및 결과를 기록하고 수정 보완 작업 | | | | | | | | | | |
| **Documentation about development process** |  |  |  | 4 | 5 | 6 | 7 | 8 | 9 | 50% |
| 개발 과정에 대한 문서 작업 | | | | | | | | | | |
| **Final document modification** |  |  |  |  |  |  |  |  | 9 | 0% |
| 최종 보고서 작업 | | | | | | | | | | |

라. Android Studio 개발 주요 파일 구성 및 상호 관계

주요 파일 구성 및 상호 관계(app 파일 및 Gradle Scripts)

현재 왼쪽에 보이는 스크린 샷 속의 파일들이 어플리케이션 개발에 쓰이는 app 파일들이다.

|  |  |
| --- | --- |
| **Manifests** | AndroidManifest.xml 이란 파일 하나만 들어있는 파일이며 어플리케이션을 구동하는데 필요한 설정 값 을 관리 해주는 역할을 담당 |
| **Java** | 자바 코드가 들어있는 클래스를 관리하는 폴더 |
| **Res** | Resource 폴더. UI 와 관련 된 파일과 디자인 리소스, 문자열 리소스를 담고 있다 |

**<Manifests>**

매니페스트 파일에서는 안드로이드 시스템이 앱의 코드를 실행하기 전에 확보해야 하는 앱에 대한 필수 정보를 시스템에 제공한다.(구글 개발자 문서)

어플리케이션에 대한 자바 패키지의 이름을 지정. 이 패키지 이름은 어플리케이션에 대한 고유한 식별자 역할을 한다

액티비티, 서비스, 브로드캐스트 수신기 및 컨텐츠 제공자등 어플리케이션을 이루는 구성요소를 설명한다. 또한 각 구성 요소를 구현하는 클래스의 이름을 지정하고 클래스가 처리할 수 있는 기능을 게시한다(ex. Intent 메시지). 이러한 선언은 안드로이드에 구성 요소와 그 요소가 실행 될 수 있는 조건을 알린다

어플리케이션이 연결되어야 하는 라이브러리를 나열한다

**메니페스트 파일이 수행하는 작업( 구글 개발자 문서)**

어플리케이션이 필요로 하는 안드로이드 API의 최소 레벨을 선언한다

어플리케이션 구성 요소를 호스팅 하는 프로세스를 결정 한다

어플리케이션이 API의 보호된 부분에 액세스 하여 다른 어플리케이션과 상호 작용 하는 데 보유해야 하는 권한을 선언한다

<Manifest 파일의 필수 요소>

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| <manifest>  ~  <application>  ~  </application>  </manifest> | 두가지 요소 <manifest> 와 <application> 은 반드시 제공되어야 하며 한 번만 나올 수 있다. 다른 요소들은 여러 번 또는 전혀 나오지 않아도 된다. 하지만 이러한 여러 요소 중 적어도 일부가 제공 되어야 manifest파일이 유효하다.  여러 요소는 순서가 지정되지 않는다. 하지만 이러한 규칙에는 아래의 두 가지 중요한 예외 사항이 있다.   |  |  | | --- | --- | | **1.** | <activity-alias>요소는 이 요소를 별칭으로 사용하는 <activity>다음에 와야 한다 | | **2.** | <application>요소는 <manifest>요소 내부에 있는 마지막 요소여야 한다. 즉, </application>이 </manifest>태그 바로 앞에 와야 한다. | | |
| 나폴나폴 어플리케이션의 Manifest 현재까지의 전체 코드 | |
| *<?***xml version="1.0" encoding="utf-8"***?>* <**manifest xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:tools="http://schemas.android.com/tools"  package="com.omd.beta02"**>   //카메라 기능에 대한 권한 설정  <**uses-permission android:name="android.permission.CAMERA"**/>   //저장소 쓰기 기능에 대한 권한 설정  <**uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"** />   //저장소 읽기 기능에 대한 권한 설정  //갤러리로부터의 데이터 불러오기 기능의 오류로 인한 허가권 설정  <**uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE"** />   //android:hardwareAccelerated="false" 를 통한 기기작동 오류에 대한 보완, 기기마다 갤러리 불러오기 및  //두번째 촬영부터 이미지 뷰에 이미지가 표시 되지않는 오류가 발생하였기에 코드를 추가 시켜 주었다.  <**application  android:allowBackup="true"  android:icon="@mipmap/ic\_launcher"  android:label="@string/app\_name"  android:roundIcon="@mipmap/ic\_launcher\_round"  android:supportsRtl="true"  android:theme="@style/AppTheme"  android:hardwareAccelerated="false"**>   //촬영된 사진에 대한 provider 즉, 공급자 역할을 해준다 resource 코드의 file\_paths는 이미지 파일이 저장될 위치를 지정 해주는 역할 을 한다.  <**provider  android:name="android.support.v4.content.FileProvider"  android:authorities="com.omd.beta02.fileprovider"  android:exported="false"  android:grantUriPermissions="true"**>  <**meta-data  android:name="android.support.FILE\_PROVIDER\_PATHS"  android:resource="@xml/file\_paths"**></**meta-data**>  </**provider**>   <**activity  android:name=".MainActivity"  android:label="@string/app\_name"  android:theme="@style/AppTheme.NoActionBar"**>  <**intent-filter**>  <**action android:name="android.intent.action.MAIN"** />  <**category android:name="android.intent.category.LAUNCHER"** />  </**intent-filter**>  </**activity**>   <**activity android:name=".contentlist"  android:label="content list"  android:parentActivityName=".MainActivity"**>  <**intent-filter**>  <**action android:name="android.intent.action.PICK"** />  <**category android:name="android.intent.category.LAUNCHER"** />  </**intent-filter**>  </**activity**>  </**application**>  </**manifest**> | |

c

d-1

d-2

d-3

b

a

다음페이지에서 선택 된 영역을 설명 한다

‘나폴나폴’ 어플리케이션 Manifest파일 안의 코드에 대한 설명

|  |  |
| --- | --- |
| **a** | xmlns |
| Xml namespace의 약자로 xml 문서 작성시 네임 스페이스를 지정해 줌으로써 이 문서는 어떤 형태로 작성 되었다는 걸 명시하고있다.(네임 스페이스란 xml 요소간의 이름 충돌을 방지해주는 방법을 제공)  항상 http://schemas.android.com/apk/res/android 으로 세팅 되어져야 한다.(구글 개발자 페이지 참고) |

|  |  |
| --- | --- |
| **b** | <manifest>내의 package=”com.omd.beta02” 의 의미는 com 하위의 omd 하위의 beta02라는 프로젝트 경로를 가리키는 것이다. |
|  |

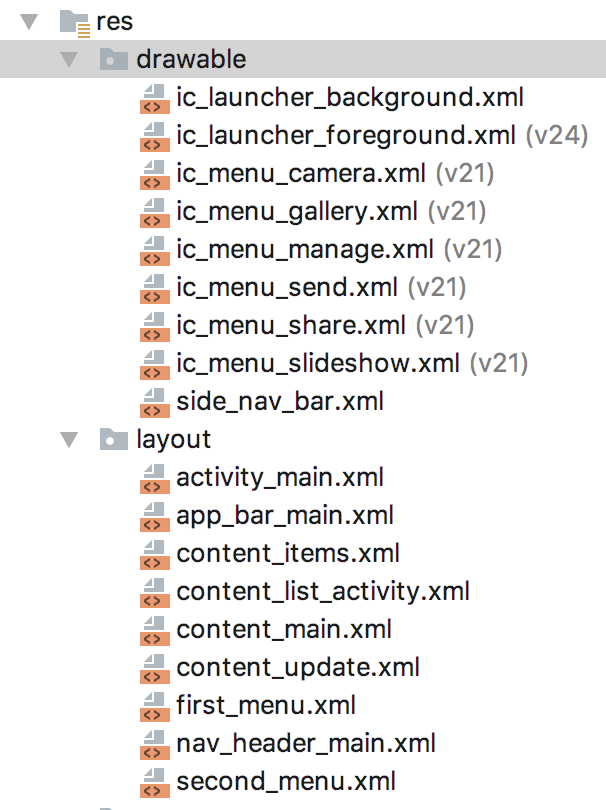
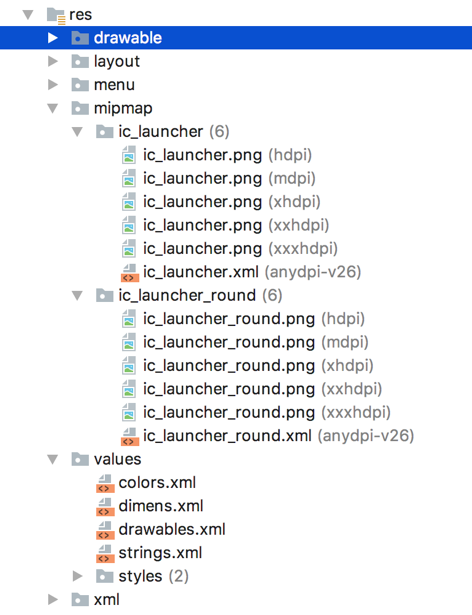
|  |  |
| --- | --- |
| **c** | 카메라 기능을 위한 카메라 접근과 갤러리 이미지 불러오기를 위한 갤러리 접근 권한 에 대한 설정을 담당하는 코드이다 |
|  |

|  |  |  |
| --- | --- | --- |
| <application>내의 세가지 코드 | | |
| **d-1** | **android:icon="@mipmap/ic\_launcher"** | mipimap폴더 하위에 ic\_launcher라는 파일을 앱의 아이콘으로 설정 해주는 코드 |
| **d-2** | **android:theme="@style/AppTheme"** | 어플리케이션에 AppTheme이라는 테마를 적용시키는 코드 |
| **d-3** | **android:label="@string/app\_name"** | 어플리케이션 아이콘에 표시 될 앱이름을 정해 줄 수 있는 코드 |

**<java Package>**

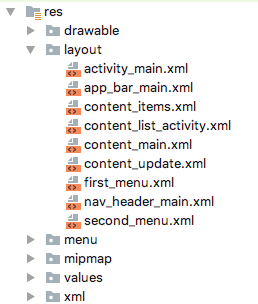
|  |
| --- |
| 어플리케이션이 작동하는 방식에 대해 정의하는 모든 java 파일이 이곳에 들어있다    이 폴더 내부에 있는 MainActivity.java 는 모든 프로젝트에 필수적으로 있어야 하는 기본 파일이다. 이 자바 파일의 본문에는 onCreate()라는 함수가 정의 되어 있고 그 안에는 setContentView(R.layout.***activity\_main***);  이라는 코드가 있다. setContentView의 기능은 layout패키지 아래에 있는 activity\_main이라는 파일을 view로 연결 해준다는 의미가 담겨 있다. 즉, 자바 코드에 의해 Activity파일과 xml파일은 언제나 한 쌍이며 서로 연결되어있다.. |

**<res>**

**** ****

Resource 파일 의 줄임말로써 기본적으로 네 가지의 package를 포함 하고 있습니다

|  |  |  |
| --- | --- | --- |
| **Layout** | 레이아웃 파일 저장 패키지 | |
| **Drawable** | 이미지 파일 저장 패키지 | |
| **Mipmap** | launcher 이미지(=앱 아이콘) 저장 패키지 | |
| **Values** | 문자열, 색상값, 수치값, 스타일 저장 패키지  아래의 세가지 파일 포함 | |
| **Color.xml** | 컬러값 저장 파일 |
| **String.xml** | 문자열값 저장 파일 |
| **Styles.xml** | 스타일 값 저장 파일 |

****

어플리케이션 개발 중 생성된 레이아웃 파일

|  |  |  |
| --- | --- | --- |
| **레이아웃 파일 name** | **파일에 대한 설명** | **파일로 인해 생성되는 UI** |
| **Activity\_main** | 네비게이션 뷰를 관리하는 xml 파일 |  |
| **Content\_items** | grid뷰를 통해 리스트를 하나 씩 추가시켜줄 형태의 뷰를 정의  오른쪽의 ui가 리스트 객체 하나당 하나씩 생성 |  |
| **Content\_list\_activity** | grid뷰를 정의 하여서 격자형태의 리스트를 생성 해주는 xml파일 |  |
| **Content\_main** | 이 어플리케이션의 main page 이다 메인 페이지 내에 필요한 기능 들이 구현 됨 |  |
| **Content\_update** | dialog기능을 통해 업데이트 기능 연결시 보여 줄 업데이트 기능 화면 정의 |  |
| **Nav\_header\_main** | 네비게이션 뷰 안의 헤더 파일을 설정하는 xml파일(미사용) |  |
| **First\_menu, second\_menu** | 네비게이션 뷰 기능 확인을 위해 정의 한 xml(미사용) | (미사용) |

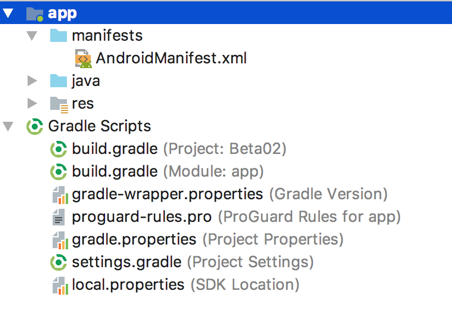
|  |
| --- |
| Xml 파일이란 우리가 보는 UI를 구현하는 파일이다. 쉽게 말해 디자인이라고 할 수 있다.  Xml 파일 내에서는 <>로 코드를 구성해주게 되고 Design 탭을 이용해 실제 디자인이 진행되고있는 것에 대한 미리 보기 기능을 제공한다 |

**<gradle>**

|  |
| --- |
| 안드로이드 빌드 시스템은 앱 리소스 및 소스 코드를 컴파일 하고 테스트, 구축, 서명 및 배포할 수 있는 APK로 패키징 한다. 안드로이드 스튜디오는 고급 빌드 툴킷인 Gradle을 사용하여 빌드 프로세스를 자동화 하고 관리 한다. 사용자는 스스로 지정 빌드 구성을 정의 할 수 있다. 각 빌드 구성에서는 자체적인 코드 및 리소스 세트를 정의 할 수 있고, 모든 어플리케이션 버전에 공통되는 부분을 재 사용 하기 도 한다. |

APK : Android application package(안드로이드 응용 프로그램 패키지) – 안드로이드 소프트웨어 배포에 사용되는 패키지 파일. APK 파일이 만들어지려면 안드로이드용 프로그램을 먼저 컴파일 된 후, 모든 파일들이 하나의 패키지 파일로 만들어 지면 된다.

|  |
| --- |
| Gradle 이란 빌드 배포 도구 이다. 하지만 안드로이드 스튜디오(IDE)와 빌드 시스템이 서로 독립적이기 때문에 안드로이드 스튜디오가 설지 최지 않은 컴퓨터에서도 안드로이드 앱을 빌드 할 수 있다. 즉, 어디서 빌드하든 간에 빌드의 출력은 동일하다.  왼쪽의 이미지에서 gradle script는 app폴더와 따로 생성 되어진 것을 볼 수있다. |



안드로이드 스튜디오는 코드들의 편집만을 담당하고, 빌드는 gradle이 전적으로 수행하게 된다. 이렇게 기능이 분리 되어져 있으므로 인해 프로젝트 자체가 더 깔끔하게 관리 할 수 있게 되어진다.

|  |  |
| --- | --- |
|  | 상자 친 build.gradle 은 모듈의 빌드 방법이 정의 된 빌드 스크립트 이다. 모듈은 쉽게 말해 프로그램의 집합체인데 프로그램을 작성할 때 미리 작성해둔 어떤 기능을 불러와 사용 할 수 있게 해주는 기능이다. |

**모듈 레벨 빌드 파일**

<project>/<module>/ 디렉토리에 있는 모듈 레벨 build.gradle 파일을 사용하면 이 파일이 위치하는 특정 모듈의 빌드 설정을 구성할 수 있다. 이러한 빌드 설정을 구성하면 main/ 어플리케이션 매니페스트 또는 최상위 build.gradle 파일에 있는 설정을 재정의 할 수 있다.

|  |
| --- |
| Build.gradle(Module: app) code |
| apply **plugin**: **'com.android.application'** android {  compileSdkVersion 28  defaultConfig {  applicationId **"com.omd.beta02"** minSdkVersion 15  targetSdkVersion 28  versionCode 1  versionName **"1.0"** testInstrumentationRunner **"android.support.test.runner.AndroidJUnitRunner"** }  buildTypes {  release {  minifyEnabled **false** proguardFiles getDefaultProguardFile(**'proguard-android.txt'**), **'proguard-rules.pro'** }  } }  dependencies {  implementation fileTree(**dir**: **'libs'**, **include**: [**'\*.jar'**])  implementation **'com.android.support:appcompat-v7:28.0.0-alpha3'** implementation **'com.android.support:design:28.0.0-alpha3'** implementation **'com.android.support.constraint:constraint-layout:1.1.2'** testImplementation **'junit:junit:4.12'** androidTestImplementation **'com.android.support.test:runner:1.0.2'** androidTestImplementation **'com.android.support.test.espresso:espresso-core:3.0.2'** implementation **'com.android.support:gridlayout-v7:28.0.0-alpha3'** } |

-안드로이드 앱 모듈 build.gradle 파일에서의 기본 요소 설명(설명은 구글 개발자 문서에서 발췌)

|  |
| --- |
| apply **plugin**: **'com.android.application'** |
| The first line in the build configuration applies the Android plugin for Gradle to this build and makes the android {} block available to specify Android-specific build options |
| android {~} |
| The android {} block is where you configure all your Android-specific build options |
| compileSdkVersion 28 |
| compileSdkVersion specifies the Android API level Gradle should use to compile your app. This means your app can use the API features included in this API level and lower. |
| defaultConfig {~} |
| The defaultConfig {} block encapsulates default settings and entries for all build variants, and can override some attributes in main/AndroidManifest.xml dynamically from the build system. You can configure product flavors to override these values for different versions of your app. |
| applicationId **"com.omd.beta02"** |
| applicationId uniquely identifies the package for publishing. However, your source code should still reference the package name defined by the package attribute in the main/AndroidManifest.xml file. |
| minSdkVersion 15 |
| Defines the minimum API level required to run the app. |
| targetSdkVersion 28 |
| Specifies the API level used to test the app. |
| versionCode 1 |
| Defines the version number of your app. |
| buildTypes {~} |
| Defines a user-friendly version name for your app. |
| release {  minifyEnabled **false** proguardFiles getDefaultProguardFile(**'proguard-android.txt'**), **'proguard-rules.pro'** } |
| By default, Android Studio configures the release build type to enable code shrinking, using minifyEnabled, and specifies the Proguard settings file.   |  | | --- | | minifyEnabled true | | Enables code shrinking for the release build type. | |
| dependencies {~} |
| The dependencies {} block in the module-level build configuration file only specifies dependencies required to build the module itself. |

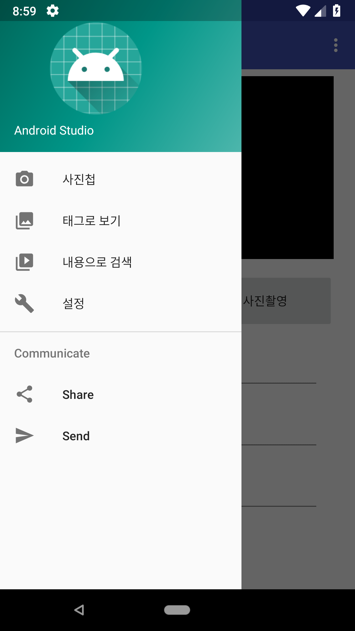
마. 기본 프로젝트 생성 및 기능 배치

새로운 프로젝트 생성 및 기본 화면 구성을 위한 진행

새로운 프로젝트를 생성 하여 개발을 시작 하게 된다. 프로젝트를 생성 할 때 액티비티를 함께 구성 하여 주는데 본인이 원하는 액티비티를 선택 하여 앞으로 사용 될 메인 액티비티를 구현 해 주게 된다. 이 어플리케이션의 메인 액티비티로 사용 될 화면은 네비게이션 기능 화면이다.

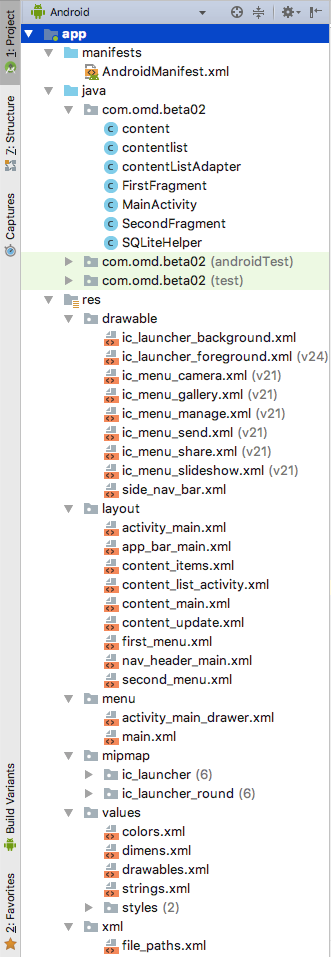
-Navigation Drawer란?

|  |
| --- |
| 메인 화면 내에서 좌측 상단의 버튼을 눌렀을 시에 동일 화면내의 좌측에서 나오는 또다른 기능 화면을 말한다.  Navigation drawer 내에서 구현 하고자 하는 기능은 사진첩 불러오기 및 저장 된 데이터 리스트로 보기 기능이 있다. |



|  |  |
| --- | --- |
| 새로운 프로젝트 생성 및 기본 화면 구성 진행 과정 | |
|  | 설치 된 안드로이드 스튜디오를 실행 한 후 메뉴에서 새로운 안드로이드 스튜디오 프로젝트 시작을 눌러 준다. |
|  | 다음 화면에서 앞으로 쓰게 될 안드로이드 스튜디오의 어플리케이션 이름을 설정 한 후 다음 화면으로 넘어가 준다.  설정 된 어플리케이션 이름은 ‘Beta02’이다.  또한 이 과정에서 프로젝트의 위치도 지정 할 수 있다. |
|  | 다음으로는 앞으로 가상머신 에서 쓰게 될 핸드폰 또는 태블릿의 버전을 선택 해 주게 된다. |
|  | 다음으로 넘어가게 되면 시작 할 때 사용할 액티비티를 선택 하게 되는데 navigation drawer activity를 선택하여 바로 네비게이션 액티비티를 만들어 주게 된다. |
|  | 마지막으로 메인 액티비티의 이름을 설정하고 나서 finish를 눌러 설정을 종료 해주면 프로젝트와 기본적인 액티비티, 레이아웃이 설정 되어지게 된다. |

**-프로젝트내의 app 폴더 구성**



**현재 프로젝트의 app 파일 단계의 구성은 크게 세 가지로 나뉜다.**

|  |  |  |
| --- | --- | --- |
| **a** | **manifest** | **매니페스트 파일에서는 안드로이드 시스템이 앱의 코드를 실행하기 전에 확보해야 하는 앱에 대한 필수 정보를 시스템에 제공한다.** |

|  |  |  |
| --- | --- | --- |
| **b** | **java** | **어플리케이션이 작동하는 방식에 대해 정의하는 모든 java 파일** |

|  |  |
| --- | --- |
| **생성된 주요 java 파일** | **간단한 설명** |
| MainActivity | 메인 xml에 있는 레이아웃 |
| SQLiteHelper | 데이터베이스 및 메소드 구현 |
| content | 데이터 베이스 값 전달 클래스 |
| contenlist | 데이터 리스트화 및 업데이트,삭제 메소드 구현 |
| contentListAdapter | 데이터 리스트화 동적할당 메소드 |

|  |  |  |
| --- | --- | --- |
| **c** | **res** | **어플리케이션 개발에 이용할 레이아웃을 설정하는 파일** |

|  |  |
| --- | --- |
| **생성된 주요 res 파일** | **간단한 설명** |
| Activity\_main.xml | 네비게이션 뷰 구성 xml |
| Content\_items.xml | 동적할당시 데이터 구성 뷰 구현 xml |
| Content\_main.xml | 메인 화면 구성 xml |
| Content\_list\_activity.xml | Grid view 구성 xml |
| Content\_update.xml | 업데이트 기능 구현 dialog 구현 XML |
| Activity\_main\_drawer.xml | 네비게이션 뷰 메뉴 구성 xml |
| File\_paths.xml | 촬영 이미지 저장 위치 할당 xml |

바. 각 구성 별 경로 및 전체 코드

**경로 : app -> manifests**

**파일 이름 : AndroidManifest.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 package="com.omd.beta02"**>  
  
 //카메라 기능에 대한 권한 설정  
 <**uses-permission android:name="android.permission.CAMERA"**/>  
  
 //저장소 쓰기 기능에 대한 권한 설정  
 <**uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"** />  
  
 //저장소 읽기 기능에 대한 권한 설정  
 //갤러리로부터의 데이터 불러오기 기능의 오류로 인한 허가권 설정  
 <**uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE"** />  
  
 //android:hardwareAccelerated="false" 를 통한 기기작동 오류에 대한 보완, 기기마다 갤러리 불러오기 및  
 //두번째 촬영부터 이미지 뷰에 이미지가 표시 되지않는 오류가 발생하였기에 코드를 추가 시켜 주었다.  
 <**application  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="NPNP"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/AppTheme"  
 android:hardwareAccelerated="false"**>  
  
 //촬영된 사진에 대한 provider 즉, 공급자 역할을 해준다 resource 코드의 file\_paths는 이미지 파일이 저장될 위치를 지정 해주는 역할 을 한다.  
 <**provider  
 android:name="android.support.v4.content.FileProvider"  
 android:authorities="com.omd.beta02.fileprovider"  
 android:exported="false"  
 android:grantUriPermissions="true"**>  
 <**meta-data  
 android:name="android.support.FILE\_PROVIDER\_PATHS"  
 android:resource="@xml/file\_paths"**></**meta-data**>  
 </**provider**>  
  
 <**activity  
 android:name=".MainActivity"  
 android:label="NPNP"  
 android:theme="@style/AppTheme.NoActionBar"**>  
 <**intent-filter**>  
 <**action android:name="android.intent.action.MAIN"** />  
 <**category android:name="android.intent.category.LAUNCHER"** />  
 </**intent-filter**>  
 </**activity**>  
  
 <**activity android:name=".contentlist"  
 android:label="content list"  
 android:parentActivityName=".MainActivity"**>  
 <**intent-filter**>  
 <**action android:name="android.intent.action.MAIN"** />  
 </**intent-filter**>  
 </**activity**>  
 </**application**>

**경로 : app -> java file -> com.omd.beta02**

**파일 이름 : MainActivity**

**package** com.omd.beta02;  
  
**import** android.Manifest;  
**import** android.content.Intent;  
**import** android.content.pm.PackageManager;  
**import** android.content.pm.ResolveInfo;  
**import** android.database.Cursor;  
**import** android.graphics.Bitmap;  
**import** android.graphics.BitmapFactory;  
**import** android.net.Uri;  
**import** android.os.AsyncTask;  
**import** android.os.Bundle;  
**import** android.os.Environment;  
**import** android.provider.MediaStore;  
**import** android.support.design.widget.FloatingActionButton;  
**import** android.support.design.widget.Snackbar;  
**import** android.support.v4.app.ActivityCompat;  
**import** android.support.v4.content.ContextCompat;  
**import** android.support.v4.content.FileProvider;  
**import** android.util.Log;  
**import** android.view.View;  
**import** android.support.design.widget.NavigationView;  
**import** android.support.v4.view.GravityCompat;  
**import** android.support.v4.widget.DrawerLayout;  
**import** android.support.v7.app.ActionBarDrawerToggle;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.support.v7.widget.Toolbar;  
**import** android.view.Menu;  
**import** android.view.MenuItem;  
**import** android.widget.Button;  
**import** android.widget.EditText;  
**import** android.widget.ImageView;  
**import** android.widget.Toast;  
  
**import** java.io.File;  
**import** java.io.IOException;  
**import** java.net.URI;  
**import** java.text.SimpleDateFormat;  
**import** java.util.ArrayList;  
**import** java.util.Date;  
**import** java.util.List;  
  
**import static** android.os.Environment.*getExternalStoragePublicDirectory*;  
  
**public class** MainActivity **extends** AppCompatActivity  
 **implements** NavigationView.OnNavigationItemSelectedListener {  
  
 *//코드의 숫자에 따라 기능이 실행 되도록 하기 위한 코드 설정* **private static final int *CAMERA\_CODE*** = 10;  
 **private static final int *GALLERY\_CODE*** = 0;  
  
 *//전역 으로 선언 해주어야 하는 변수 또는 상수 들이다.  
 //전역 으로 선언 해줄 시 데이터 값을 불러와 사용하기 편해지기 때문에 전역으로 선언해 주면 좋다  
 //모든 변수를 전역으로 선언하게 되는 것은 좋지 않다* **private** String **mCurrentPhotoPath**;  
 EditText **editText\_tags1**, **editText\_tags2**, **editText\_tags3**;  
 Button **button\_save2**;  
 **int i** = 0;  
 **public static** SQLiteHelper *sqLiteHelper*;  
 **private** String **TAG**;  
 **private** Uri **contentUri** , **photoUri**, **galleryUri**, **newUri**, **uri2**;  
 **private** String **getmCurrentPhotoPath**;  
  
 *//자바로 따지면 main class의 기능을 하는 onCreate 메소드 이다.  
 //자바의 main class와 차이점 이라면 onCreate 메소드는 클래스당 필요 하다는 점이다.* @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
  
 *//가상머신에서 구동 했을시 문제 발생에 대한 문제점 파악을 위한 로그 추적 기능이다.* Log.*d*(**TAG**,**"Problem"**);  
  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 Toolbar toolbar = (Toolbar) findViewById(R.id.***toolbar***);  
 setSupportActionBar(toolbar);  
  
 *//사진 촬영에 대한 허가권을 얻는 기능  
 //onCreate 밖에 선언해준 메소드를 이용 하여 기능 구현을 도와준다.* requirePermission();  
  
 init();  
  
 *//sqLiteHelper class로 부터 DB 생성에 관련한 메소드들과 DB를 호출 하고  
 //Table을 생성해주는 코드를 입력 하여 준다.  
 sqLiteHelper* = **new** SQLiteHelper(**this**, **"Example01.sqlite"**, **null**, 1);  
 *sqLiteHelper*.queryData(**"CREATE TABLE IF NOT EXISTS EXAMPLEn (id INTEGER PRIMARY KEY AUTOINCREMENT, tags1 VARCHAR, tags2 VARCHAR,tags3 VARCHAR, uri VARCHAR)"**);  
  
 *//앞으로 사진출력용으로 사용 될 xml에 대한 선언을 해준다.* **final** ImageView imageView = (ImageView) findViewById(R.id.***picture\_view***);  
  
 */\*//갤러리 불러오기 기능으로 사용하고자 하였던 버튼 기능  
 Button gallery = (Button)findViewById(R.id.button\_gallery);  
 gallery.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 imageView.setImageResource(0);  
 //pickUpPicture();  
 }  
 });\*/  
  
 //button\_save2는 현재 view에 표시되고있는 사진과 추가된 태그를 데이터 베이스로 넘겨주는 기능을 한다.* Button button\_save2 = (Button) findViewById(R.id.***button\_save2***);  
  
 button\_save2.setOnClickListener(**new** View.OnClickListener(){  
  
 @Override  
 **public void** onClick(View view){  
 **try** {  
 *//i로 설정해준 변수 값이 1일 경우는 갤러리에서 불러온 파일에 대한 uri를 받아온 경우이다* **if**(**i**==1) {  
 **newUri** = **galleryUri**;  
 *sqLiteHelper*.insertData(  
 **editText\_tags1**.getText().toString().trim(),  
 **editText\_tags2**.getText().toString().trim(),  
 **editText\_tags3**.getText().toString().trim(),  
 **newUri**.toString().trim()  
 );  
  
 **editText\_tags1**.setText(**""**);  
 **editText\_tags2**.setText(**""**);  
 **editText\_tags3**.setText(**""**);  
  
  
 **galleryUri** = **null**;  
 }  
 *//i로 설정해준 변수 값이 2 일 경우는 카메라 촬영을 통해 얻은 이미지 파일의 uri이다* **if**(**i**==2){  
 **newUri** = **photoUri**;  
 *sqLiteHelper*.insertData(  
 **editText\_tags1**.getText().toString().trim(),  
 **editText\_tags2**.getText().toString().trim(),  
 **editText\_tags3**.getText().toString().trim(),  
 **newUri**.toString().trim()  
 );  
  
 **editText\_tags1**.setText(**""**);  
 **editText\_tags2**.setText(**""**);  
 **editText\_tags3**.setText(**""**);  
  
 **photoUri** = **null**;  
 }  
  
 *//uri를 받아온 후 i 변수값 초기화 과정* **i**=0;  
  
 Toast.*makeText*(getApplicationContext(), **"기록 저장 성공"**,Toast.***LENGTH\_SHORT***).show();  
 }  
 **catch**(Exception e){  
 e.printStackTrace();  
 }  
 }  
 });  
  
  
 **try** {  
 *//button\_save 버튼은 촬영 된 사진에 대해 넘어온 정보 값들을 받아 와 실제 기기 내의 저장 공간에 저장 해주는 형식이다* Button button\_save = (Button) findViewById(R.id.***button\_save***);  
  
 *//Button button\_save = (Button) findViewById(R.id.button\_save);* button\_save.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
  
 *//사진 저장을 위해 galleryAddPic이라는 메소드를 구현 하였고  
 //메소드를 호출 하게 되면 카메라 기능으로 부터 정보를 받아와 galleryAddPic으로 정보를 넘겨 주게 된다.* galleryAddPic();  
 }  
 });  
 }**catch**(NullPointerException e){  
 Toast.*makeText*(**this**, **"이미 저장되거나 없는 이미지입니다"**, Toast.***LENGTH\_SHORT***).show();  
 }  
  
  
 *//camera\_button을 이용한 카메라 기능과의 연동* Button button = (Button) findViewById(R.id.***camera\_button***);  
 button.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
  
  
 **boolean** camera = ContextCompat.*checkSelfPermission* (view.getContext(), Manifest.permission.***CAMERA***) == PackageManager.***PERMISSION\_GRANTED***;  
  
 **boolean** write = ContextCompat.*checkSelfPermission* (view.getContext(), Manifest.permission.***WRITE\_EXTERNAL\_STORAGE***) == PackageManager.***PERMISSION\_GRANTED***;  
  
 **if** (camera && write) {  
 *//사진찍은 인텐트 코드 넣기* takePicture();  
 } **else** {  
 Toast.*makeText*(MainActivity.**this**, **"카메라 권한 및 쓰기 권한을 주지 않았습니다."**, Toast.***LENGTH\_SHORT***).show();  
 }  
  
 }  
 });  
  
 */\*//아직 구현 되지 않은 기능  
 FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);  
 fab.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH\_LONG)  
 .setAction("Action", null).show();  
 }  
 });\*/* DrawerLayout drawer = (DrawerLayout) findViewById(R.id.***drawer\_layout***);  
 ActionBarDrawerToggle toggle = **new** ActionBarDrawerToggle(  
 **this**, drawer, toolbar, R.string.***navigation\_drawer\_open***, R.string.***navigation\_drawer\_close***);  
 drawer.addDrawerListener(toggle);  
 toggle.syncState();  
  
 NavigationView navigationView = (NavigationView) findViewById(R.id.***nav\_view***);  
 navigationView.setNavigationItemSelectedListener(**this**);  
 }  
  
  
 *//////////////// oncreate 끝지점\\\\\\\\\\\\\\\\\\\\  
 //현재 지점 부터는 외부에 선언되는 메소드 또는 변수들이다.  
  
  
 //글과 사진 저장을 위해 선언한 버튼 및 edittext들의 view에 대한 initial 선언 메소드* **private void** init(){  
 **editText\_tags1** = (EditText) findViewById(R.id.***editText\_tags1***);  
 **editText\_tags2** = (EditText) findViewById(R.id.***editText\_tags2***);  
 **editText\_tags3** = (EditText) findViewById(R.id.***editText\_tags3***);  
 **button\_save2** = (Button) findViewById(R.id.***button\_save2***);  
 }  
  
  
 @Override  
 **public void** onBackPressed() {  
 DrawerLayout drawer = (DrawerLayout) findViewById(R.id.***drawer\_layout***);  
 **if** (drawer.isDrawerOpen(GravityCompat.***START***)) {  
 drawer.closeDrawer(GravityCompat.***START***);  
 } **else** {  
 **super**.onBackPressed();  
 }  
 }  
  
 @Override  
 **public boolean** onCreateOptionsMenu(Menu menu) {  
 *// Inflate the menu; this adds items to the action bar if it is present.* getMenuInflater().inflate(R.menu.***main***, menu);  
 **return true**;  
 }  
  
 @Override  
 **public boolean** onOptionsItemSelected(MenuItem item) {  
 *// Handle action bar item clicks here. The action bar will  
 // automatically handle clicks on the Home/Up button, so long  
 // as you specify a parent activity in AndroidManifest.xml.* **int** id = item.getItemId();  
  
 *//noinspection SimplifiableIfStatement* **if** (id == R.id.***action\_settings***) {  
 **return true**;  
 }  
  
 **return super**.onOptionsItemSelected(item);  
 }  
  
 @SuppressWarnings(**"StatementWithEmptyBody"**)  
 @Override  
 **public boolean** onNavigationItemSelected(MenuItem item) {  
 *// Handle navigation view item clicks here.* **int** id = item.getItemId();  
 *//FragmentManager fragmentManager = getFragmentManager();* **if** (id == R.id.***nav\_gallery***) {  
 Intent intent = **new** Intent(Intent.***ACTION\_PICK***);  
 intent.setType(MediaStore.Images.Media.***CONTENT\_TYPE***);  
 intent.setType(**"image/\*"**);  
 startActivityForResult(intent, ***GALLERY\_CODE***);  
  
 */\*setTitle("camera");  
 FirstFragment fragment = new FirstFragment();  
 FragmentTransaction fragmentTransaction = getSupportFragmentManager().beginTransaction();  
 fragmentTransaction.replace(R.id.content\_frame, fragment, "FirstFragment");  
 fragmentTransaction.commit();\*/* } **else if** (id == R.id.***nav\_tags***) {  
  
  
 Intent intent = **new** Intent(MainActivity.**this**, contentlist.**class**);  
 startActivity(intent);  
  
  
 */\*setTitle("gallery");  
 SecondFragment fragment = new SecondFragment();  
 FragmentTransaction fragmentTransaction = getSupportFragmentManager().beginTransaction();  
 fragmentTransaction.replace(R.id.content\_frame, fragment, "SecondFragment");  
 fragmentTransaction.commit();\*/* } **else if** (id == R.id.***nav\_slideshow***) {  
  
  
  
 } **else if** (id == R.id.***nav\_manage***) {  
  
 } **else if** (id == R.id.***nav\_share***) {  
  
 } **else if** (id == R.id.***nav\_send***) {  
  
 }  
  
 DrawerLayout drawer = (DrawerLayout) findViewById(R.id.***drawer\_layout***);  
 drawer.closeDrawer(GravityCompat.***START***);  
 **return true**;  
 }  
  
  
 **void** requirePermission() {  
  
 String[] permissions = **new** String[]{Manifest.permission.***CAMERA***, Manifest.permission.***WRITE\_EXTERNAL\_STORAGE***};  
 ArrayList<String> listPermissionsNeeded = **new** ArrayList<>();  
  
 **for** (String permission : permissions) {  
  
 **if** (ContextCompat.*checkSelfPermission*(**this**, permission) == PackageManager.***PERMISSION\_DENIED***) {  
 *//권한이 허가가 안됬을 경우 요청할 권한을 모집하는 부분* listPermissionsNeeded.add(permission);  
 }  
  
 }  
  
 **if** (!listPermissionsNeeded.isEmpty()) {  
  
 *//권한 요청 하는 부분* ActivityCompat.*requestPermissions*(**this**, listPermissionsNeeded.toArray(**new** String[listPermissionsNeeded.size()]), 1);  
 }  
  
  
 }  
 **void** takePicture(){  
  
 Intent intent = **new** Intent(MediaStore.***ACTION\_IMAGE\_CAPTURE***);  
 **try** {  
 File photoFile = createImageFile();  
 **photoUri** = FileProvider.*getUriForFile*(**this**, **"com.omd.beta02.fileprovider"**, photoFile);  
 intent.putExtra(MediaStore.***EXTRA\_OUTPUT***,**photoUri**);  
 startActivityForResult(intent, ***CAMERA\_CODE***);  
 **i**=2;  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 }  
  
  
 **private** File createImageFile() **throws** IOException {  
 *// Create an image file name* String timeStamp = **new** SimpleDateFormat(**"yyyyMMdd\_HHmmss"**).format(**new** Date());  
 String imageFileName = **"JPEG\_"** + timeStamp + **"\_"**;  
 *//File storageDir = getExternalFilesDir(Environment.DIRECTORY\_PICTURES);* File storageDir = *getExternalStoragePublicDirectory*(Environment.*DIRECTORY\_DCIM*);  
 File image = File.*createTempFile*(  
 imageFileName, */\* prefix \*/* **".jpg"**, */\* suffix \*/* storageDir */\* directory \*/* );  
  
 *// Save a file: path for use with ACTION\_VIEW intents* **mCurrentPhotoPath** = image.getAbsolutePath();  
 **return** image;  
 }  
  
 **private void** setPic() {  
 *// Get the dimensions of the View* }  
  
 @Override  
 **protected void** onActivityResult(**int** requestCode, **int** resultCode, Intent data){  
 *//super.onActivityResult(resultCode, resultCode, data);* **if**(requestCode==***CAMERA\_CODE***){  
 Button button\_save1 = (Button) findViewById(R.id.***button\_save***);  
 ImageView imageView = (ImageView) findViewById(R.id.***picture\_view***);  
 imageView.setImageBitmap(BitmapFactory.*decodeFile*(**mCurrentPhotoPath**));  
 imageView.setScaleType(ImageView.ScaleType.***FIT\_XY***);  
 button\_save1.setVisibility(View.***VISIBLE***);  
 }  
 **if**(requestCode == ***GALLERY\_CODE*** && resultCode == ***RESULT\_OK***){  
 *//절대경로로 받아오기  
 //바로 아래의 주석 처리 된 코드가 그냥 uri를 받아와 그 uri 정보를 통해 갤러리로부터 이미지 불러오기 기능을  
 //구현하고자 하였던 코드이다.  
 //uri = data.getData();  
 //따라서 아래의 두줄과 같이 uri를 똑같이 선언 해준후 그 uri를 절대 경로를 얻는 메소드(미리 선언해 놓은)를 이용해  
 //절대 경로를 얻어 언제 어디서나 같은 경로로 이미지 파일을 얻어올 수 있게 해 준 것이다.* **uri2** = data.getData();  
 **galleryUri** = Uri.*parse*(getRealPathFromURI(**uri2**));  
  
 Button button\_save2 = (Button) findViewById(R.id.***button\_save***);  
 ImageView imageView = (ImageView) findViewById(R.id.***picture\_view***);  
 imageView.setScaleType(ImageView.ScaleType.***FIT\_XY***);  
 imageView.setImageURI(**galleryUri**);  
 button\_save2.setVisibility(View.***GONE***);  
 **i**=1;  
  
 }  
 }  
  
 **private void** galleryAddPic() {  
 **try** {  
 Intent mediaScanIntent = **new** Intent(Intent.***ACTION\_MEDIA\_SCANNER\_SCAN\_FILE***);  
 File f = **new** File(**mCurrentPhotoPath**);  
 **contentUri** = Uri.*fromFile*(f);  
 mediaScanIntent.setData(**contentUri**);  
 **this**.sendBroadcast(mediaScanIntent);  
 Toast.*makeText*(**this**, **"사진 저장 완료!"**, Toast.***LENGTH\_SHORT***).show();  
 }**catch**(Exception e){  
 e.printStackTrace();  
 }  
 }  
  
 *//uri를 통한 절대 경로를 알아오는 메소드  
 //카메라를 통한 이미지 촬영은 촬영 되는 순간 저장 되는 경로를 알 수 있으며 기본적인 이미지 내에 같이 저장 되어 지기 때문에 카메라를 통한 이미지 촬영에서는 오류가 나지 않았다.  
 //갤러리로 부터의 이미지 불러오기는 uri를 이용해 불러오기 기능을 사용 하기 때문에 실제 기기나 가상 머신 환경에서마다의 경로 설정 차이로 인해  
 //두번째 불러오기 부터 오류가 났었기 때문에 uri를 통해 절대 경로를 얻어와 어디서나 오류 없이 절대적 경로를 이용한 갤러리로부터의 이미지 불러오기 기능을 구현 하고자 하였다.* **private** String getRealPathFromURI(Uri galleryUri) {  
 **int** column\_index=0;  
 String[] result={MediaStore.Images.Media.***DATA***};  
 Cursor cursor = getContentResolver().query(galleryUri, result, **null**, **null**, **null**);  
  
 **if** (cursor.moveToFirst()) { *// Source is local file path* column\_index = cursor.getColumnIndexOrThrow(MediaStore.Images.Media.***DATA***);  
 }  
 **return** cursor.getString(column\_index);  
 }  
}

**경로 : app -> java file -> com.omd.beta02**

**파일 이름 : SQLiteHelper**

**package** com.omd.beta02;  
  
**import** android.content.Context;  
**import** android.database.Cursor;  
**import** android.database.sqlite.SQLiteDatabase;  
**import** android.database.sqlite.SQLiteOpenHelper;  
**import** android.database.sqlite.SQLiteStatement;  
  
  
*//DB에 대한 HELPER class를 선언 해줌으로서 Main 클래스 내에서 간편하게 호출 하여 생성 및 디비 관리를 할 수 있다.***public class** SQLiteHelper **extends** SQLiteOpenHelper {  
  
 **public** SQLiteHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, **int** version) {  
 **super**(context, name, factory, version);  
 }  
  
  
 **public void** queryData(String sql){  
 SQLiteDatabase database = getWritableDatabase();  
 database.execSQL(sql);  
 }  
  
  
 *//데이터 삽입 관련 메소드  
 //전달 받은 인자값을 통해 데이터를 입력 받는다* **public void** insertData(String tags1 , String tags2,String tags3, String uri){  
 SQLiteDatabase database = getWritableDatabase();  
 String sql = **"INSERT INTO EXAMPLEn VALUES (NULL, ?, ?, ?, ?)"**;  
  
 SQLiteStatement statement = database.compileStatement(sql);  
 statement.clearBindings();  
  
 statement.bindString(1, tags1);  
 statement.bindString(2, tags2);  
 statement.bindString(3, tags3);  
 statement.bindString(4, uri);  
  
 statement.executeInsert();  
  
  
 }  
  
 *//업데이트 기능을 구현해 주기 위한 메소드* **public void** updateData(String tags1, String tags2, String tags3, String uri, **int** id){  
 SQLiteDatabase database = getWritableDatabase();  
  
 String sql = **"UPDATE EXAMPLEn SET tags1 = ?, tags2 = ?, tags3 = ?, uri = ? WHERE id = ?"**;  
 SQLiteStatement statement = database.compileStatement(sql);  
  
 statement.bindString(1, tags1);  
 statement.bindString(2, tags2);  
 statement.bindString(3, tags3);  
 statement.bindString(4, uri);  
 statement.bindDouble(5, (**double**)id);  
  
 statement.execute();  
 database.close();  
 }  
  
 **public void** deleteData(**int** id){  
 SQLiteDatabase database = getWritableDatabase();  
  
 String sql = **"DELETE FROM EXAMPLEn WHERE id = ?"**;  
 SQLiteStatement statement = database.compileStatement(sql);  
  
 statement.clearBindings();  
 statement.bindDouble(1, (**double**)id);  
  
 statement.execute();  
 database.close();  
 }  
  
 *//데이터를 읽기 위한 cursor 메소드를 정의 한 것이다.  
 //cursor메소드를 통해 주소값을 불러와 원하는 위치의 값을 읽어 올 수 있게 된다.* **public** Cursor getData(String sql){  
 SQLiteDatabase database = getReadableDatabase();  
 **return** database.rawQuery(sql,**null**);  
 }  
  
 @Override  
 **public void** onCreate(SQLiteDatabase sqLiteDatabase) {  
  
 }  
  
 @Override  
 **public void** onUpgrade(SQLiteDatabase sqLiteDatabase, **int** i, **int** i1) {  
  
 }  
}

**경로 : app -> java file -> com.omd.beta02**

**파일 이름 : content**

**package** com.omd.beta02;  
  
**public class** content {  
  
 **private int id**;  
 **private** String **tags1**;  
 **private** String **tags2**;  
 **private** String **tags3**;  
 **private** String **uri**;  
  
 **public** content(String tags1, String tags2,String tags3, String uri, **int** id){  
 **this**.**tags1** = tags1;  
 **this**.**tags2** = tags2;  
 **this**.**tags3** = tags3;  
 **this**.**uri** = uri;  
 **this**.**id** = id;  
 }  
  
 **public int** getId(){  
 **return id**;  
 }  
 **public void** setId(**int** id){  
 **this**.**id** = id;  
 }  
  
 **public** String getTags1(){  
 **return tags1**;  
 }  
 **public void** setTags1(String tags1){  
 **this**.**tags1** = tags1;  
 }  
  
 **public** String getTags2(){  
 **return tags2**;  
 }  
 **public void** setTags2(String tags2){  
 **this**.**tags2** = tags2;  
 }  
  
 **public** String getTags3(){  
 **return tags3**;  
 }  
 **public void** setTags3(String tags3){  
 **this**.**tags3** = tags3;  
 }  
  
 **public** String getUri(){  
 **return uri**;  
 }  
 **public void** setUri(String uri){  
 **this**.**uri** = uri;  
 }  
  
}

**경로 : app -> java file -> com.omd.beta02**

**파일 이름 : contentlist**

**package** com.omd.beta02;  
  
**import** android.Manifest;  
**import** android.app.Activity;  
**import** android.app.Dialog;  
**import** android.content.DialogInterface;  
**import** android.content.Intent;  
**import** android.content.pm.PackageManager;  
**import** android.database.Cursor;  
**import** android.net.Uri;  
**import** android.os.Bundle;  
**import** android.provider.MediaStore;  
**import** android.support.annotation.Nullable;  
**import** android.support.v4.app.ActivityCompat;  
**import** android.support.v4.content.ContextCompat;  
**import** android.support.v7.app.AlertDialog;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.util.Log;  
**import** android.view.View;  
**import** android.widget.AdapterView;  
**import** android.widget.Button;  
**import** android.widget.EditText;  
**import** android.widget.GridView;  
**import** android.widget.ImageView;  
**import** android.widget.Toast;  
  
**import** java.text.BreakIterator;  
**import** java.util.ArrayList;  
  
**import static** android.text.Selection.*setSelection*;  
  
**public class** contentlist **extends** AppCompatActivity{  
  
 **private static** String *tags11*;  
 **private static** String *tags22*;  
 **private static** String *tags33*;  
 **private static** String *uri1*;  
 **private static int** *id1*;  
 **private static int** *id2*;  
 **private** String **TAG**;  
 GridView **gridView**;  
 ArrayList<content> **list**;  
 contentListAdapter **adapter** = **null**;  
 **private static final int *GALLERY\_CODE*** = 0;  
 Uri **galleryUri**, **uri**;  
 ImageView **imageView\_update**;  
 Button **button\_update**;  
  
  
 @Override  
 **protected void** onCreate(@Nullable Bundle savedInstaceState) {  
 **super**.onCreate(savedInstaceState);  
 setContentView(R.layout.***content\_list\_activity***);  
 **gridView** = (GridView) findViewById(R.id.***gridView***);  
  
 Log.*d*(**TAG**,**"Problem2"**);  
 requirePermission();  
  
 **list** = **new** ArrayList<>();  
 **adapter** = **new** contentListAdapter(**this**, R.layout.***content\_items***, **list**);  
 **gridView**.setAdapter(**adapter**);  
  
 *//get all data from sqlite  
  
 //cursor 기능은 DB에서 가져온 데이터를 쉽게 처리 하기 위해서 제공해주는 인터페이스 이다.  
 //cursor는 가져온 DB데이터를 한행 씩 참조하는 것 처럼 사용하여 개발 할 수 있도록 도와준다.* Cursor cursor = MainActivity.*sqLiteHelper*.getData(**"SELECT \* FROM EXAMPLEn"**);  
  
 **list**.clear();  
  
 *//cursor.moveToNext() 는 Cursor를 다음행으로 이동 시켜주는 메소드이다.  
 //그 외에도 moveToFirst(), moveToPrevious(), moveToPosition(position), moveToLast()등의 메소드 들이 있다.* **while** (cursor.moveToNext()){  
  
 **int** id = cursor.getInt(0);  
 String tags1 = cursor.getString(1);  
 String tags2 = cursor.getString(2);  
 String tags3 = cursor.getString(3);  
 String uri = cursor.getString(4);  
  
 *//Arraylist로 선언되어 동적 할당 받는 배열에 값을 하나씩 추가 시켜 주는 코드이다.* **list**.add(**new** content(tags1, tags2, tags3, uri, id));  
 }  
  
 *//adapter에 연결된 listview를 갱신하는 코드이다.* **adapter**.notifyDataSetChanged();  
  
 **gridView**.setOnItemClickListener(**new** AdapterView.OnItemClickListener() {  
 @Override  
 **public void** onItemClick(AdapterView<?> adapterView, View view, **final int** i, **long** l) {  
  
 *//dialog에 띄워 줄 두가지 아이템 이름 배열의 인자로 선언* CharSequence[] items = {**"업데이트"**, **"삭제"**};  
  
 *//dialog라는 객체를 만들어 다이얼로그 기능을 띄워준다, 괄호안의 위치가 다이얼로그를 생성해줄 위치* AlertDialog.Builder dialog = **new** AlertDialog.Builder(contentlist.**this**);  
  
  
 dialog.setTitle(**"선택하세요"**);  
  
 *//dialog 클릭시 행동에 관한 기능 구현* dialog.setItems(items, **new** DialogInterface.OnClickListener(){  
 @Override  
 **public void** onClick(DialogInterface dialog, **int** item){  
 *//item은 배열 이기 때문에 0 이면 첫번째 인자를 선택한 것이다.* **if**(item == 0){  
 *//update* Cursor c = MainActivity.*sqLiteHelper*.getData(**"SELECT \* FROM EXAMPLEn"**);  
  
 c.moveToPosition(i);  
  
 *id1* = c.getInt(0);  
 *tags11* = c.getString(1);  
 *tags22* = c.getString(2);  
 *tags33* = c.getString(3);  
 *uri1* = c.getString(4);  
  
 *//showDialogUpdate(contentlist.this);* showDialogUpdate(contentlist.**this**);  
  
 }**else**{  
 *//delete* Cursor c = MainActivity.*sqLiteHelper*.getData(**"SELECT id FROM EXAMPLEn"**);  
 c.moveToPosition(i);  
  
 *id2* = c.getInt(0);  
 showDialogDelete(*id2*);  
  
 }  
 }  
 });  
 dialog.show();  
 }  
 });  
  
  
 }  
  
 *////////////////\\\\\\\\\\\\\\\\\\onCreate 끝지점//////////////////\\\\\\\\\\\\\\\\* **private void** showDialogUpdate(Activity activity){  
  
 **final** Dialog dialog = **new** Dialog(activity);  
 *//content\_update xml 파일 을 dialog기능을 통해 불러온다.* dialog.setContentView(R.layout.***content\_update***);  
 dialog.setTitle(**"Update"**);  
  
 **imageView\_update** = (ImageView) dialog.findViewById(R.id.***imageView\_update***);  
 **final** EditText editText\_tags1 = (EditText) dialog.findViewById(R.id.***editText\_tags1***);  
 **final** EditText editText\_tags2 = (EditText) dialog.findViewById(R.id.***editText\_tags2***);  
 **final** EditText editText\_tags3 = (EditText) dialog.findViewById(R.id.***editText\_tags3***);  
 Button button\_update = (Button) dialog.findViewById(R.id.***button\_update***);  
  
 *//set width for dialog* **int** width = (**int**) (activity.getResources().getDisplayMetrics().**widthPixels**\*0.95);  
 *//set height for dialog* **int** height = (**int**)(activity.getResources().getDisplayMetrics().**heightPixels**\*0.85);  
 dialog.getWindow().setLayout(width, height);  
  
  
 editText\_tags1.setText(contentlist.*tags11*);  
 editText\_tags2.setText(contentlist.*tags22*);  
 editText\_tags3.setText(contentlist.*tags33*);  
 **imageView\_update**.setImageURI(Uri.*parse*(contentlist.*uri1*));  
 **imageView\_update**.setScaleType(ImageView.ScaleType.***FIT\_XY***);  
  
 *//EditText에서 커서를 가장 끝에 위치시키는 방법* editText\_tags1.setSelection(editText\_tags1.length());  
 editText\_tags2.setSelection(editText\_tags2.length());  
 editText\_tags3.setSelection(editText\_tags3.length());  
  
 dialog.show();  
 *//업데이트 다이얼로그에서 이미지 부분을 클릭할 시 다른 이미지로 변경 할 수 있는 코드* **imageView\_update**.setOnClickListener(**new** View.OnClickListener(){  
 **public void** onClick(View v){  
 *//request photo library* Intent intent = **new** Intent(Intent.***ACTION\_PICK***);  
 intent.setType(MediaStore.Images.Media.***CONTENT\_TYPE***);  
 intent.setType(**"image/\*"**);  
 startActivityForResult(intent, ***GALLERY\_CODE***);  
 }  
 });  
 *//새로 작성되거나 지워진 내용을 데이터베이스로 업데이트 시키기 위한 버튼 코드* button\_update.setOnClickListener(**new** View.OnClickListener(){  
 @Override  
 **public void** onClick(View v){  
 **try** {  
 MainActivity.*sqLiteHelper*.updateData(  
 editText\_tags1.getText().toString().trim(),  
 editText\_tags2.getText().toString().trim(),  
 editText\_tags3.getText().toString().trim(),  
 *uri1*.toString().trim(),  
 *id1* );  
 dialog.dismiss();  
 Toast.*makeText*(getApplicationContext(),**"업데이트 성공!"**,Toast.***LENGTH\_SHORT***).show();  
 }  
 **catch**(Exception error){  
 Log.*e*(**"Update error"**, error.getMessage());  
 }  
 updateContentlist();  
 }  
 });  
  
 }  
  
 **private void** showDialogDelete(**final int** i){  
 **final** AlertDialog.Builder dialogDelete = **new** AlertDialog.Builder(contentlist.**this**);  
  
 dialogDelete.setTitle(**"Warning!!"**);  
 dialogDelete.setMessage(**"Are you sure you want to this delete?"**);  
 dialogDelete.setPositiveButton(**"OK"**, **new** DialogInterface.OnClickListener() {  
 @Override  
 **public void** onClick(DialogInterface dialog, **int** which) {  
 **try** {  
 MainActivity.*sqLiteHelper*.deleteData(i);  
 Toast.*makeText*(getApplicationContext(), **"삭제 성공!"**, Toast.LENGTH\_SHORT).show();  
 }**catch** (Exception e){  
 Log.e(**"error"**, e.getMessage());  
 }  
 updateContentlist();  
 }  
 });  
  
 dialogDelete.setNegativeButton(**"Cancel"**, **new** DialogInterface.OnClickListener() {  
 @Override  
 **public void** onClick(DialogInterface dialog, **int** which) {  
 dialog.dismiss();  
 }  
 });  
 dialogDelete.show();  
  
 }  
  
 **private void** updateContentlist(){  
 Cursor cursor = MainActivity.sqLiteHelper.getData(**"SELECT \* FROM EXAMPLEn"**);  
  
 list.clear();  
  
 **while** (cursor.moveToNext()){  
  
 **int** id = cursor.getInt(0);  
 String tags1 = cursor.getString(1);  
 String tags2 = cursor.getString(2);  
 String tags3 = cursor.getString(3);  
 String uri = cursor.getString(4);  
  
 list.add(**new** content(tags1, tags2, tags3, uri, id));  
 }  
  
 adapter.notifyDataSetChanged();  
 }  
  
 **protected void** onActivityResult(**int** requestCode, **int** resultCode, Intent data) {  
 **if** (requestCode == GALLERY\_CODE) {  
  
 uri1 = String.valueOf(Uri.parse(getRealPathFromURI(data.getData())));  
 *//galleryUri = Uri.parse(getRealPathFromURI(Uri.parse(uri1)));  
  
 //ImageView imageView = (ImageView) findViewById(R.id.imageView\_update);* imageView\_update.setScaleType(ImageView.ScaleType.FIT\_XY);  
  
 imageView\_update.setImageURI(Uri.parse(uri1));  
 }  
 }  
  
 **private** String getRealPathFromURI(Uri uri) {  
 **int** column\_index=0;  
 String[] result={MediaStore.Images.Media.DATA};  
 Cursor cursor = getContentResolver().query(uri, result, **null**, **null**, **null**);  
  
 **if** (cursor.moveToFirst()) { *// Source is local file path* column\_index = cursor.getColumnIndexOrThrow(MediaStore.Images.Media.DATA);  
 }  
 **return** cursor.getString(column\_index);  
 }  
  
 **void** requirePermission() {  
  
 String[] permissions = **new** String[]{Manifest.permission.CAMERA, Manifest.permission.WRITE\_EXTERNAL\_STORAGE};  
 ArrayList<String> listPermissionsNeeded = **new** ArrayList<>();  
  
 **for** (String permission : permissions) {  
  
 **if** (ContextCompat.checkSelfPermission(**this**, permission) == PackageManager.PERMISSION\_DENIED) {  
 *//권한이 허가가 안됬을 경우 요청할 권한을 모집하는 부분* listPermissionsNeeded.add(permission);  
 }  
  
 }  
  
 **if** (!listPermissionsNeeded.isEmpty()) {  
  
 *//권한 요청 하는 부분* ActivityCompat.requestPermissions(**this**, listPermissionsNeeded.toArray(**new** String[listPermissionsNeeded.size()]), 1);  
 }  
 }  
}

**경로 : app -> java -> com.omd.beta02**

**파일 이름 : contentListAdapter**

**package** com.omd.beta02;  
  
**import** android.content.Context;  
**import** android.net.Uri;  
**import** android.view.LayoutInflater;  
**import** android.view.View;  
**import** android.view.ViewGroup;  
**import** android.widget.BaseAdapter;  
**import** android.widget.ImageView;  
**import** android.widget.TextView;  
  
**import** java.util.ArrayList;  
  
**public class** contentListAdapter **extends** BaseAdapter {  
  
 **private** Context **context**;  
 **private int layout**;  
 **private** ArrayList<content> **contentlist**;  
  
 **public** contentListAdapter(Context context, **int** layout, ArrayList<content> contentlist){  
 **this**.**context** = context;  
 **this**.**layout** = layout;  
 **this**.**contentlist** = contentlist;  
 }  
  
 @Override  
 **public int** getCount() {  
 **return contentlist**.size();  
 }  
  
 @Override  
 **public** Object getItem(**int** position) {  
 **return contentlist**.get(position);  
 }  
  
 @Override  
 **public long** getItemId(**int** position) {  
 **return** position;  
 }  
  
 **private class** ViewHolder{  
 ImageView **imageView2**;  
 TextView **textView\_retags1**;  
 TextView **textView\_retags2**;  
 TextView **textView\_retags3**;  
 }  
  
 @Override  
 **public** View getView(**int** position, View view, ViewGroup viewGroup) {  
  
 View row = view;  
 ViewHolder holder = **new** ViewHolder();  
  
 **if**(row == **null**){  
 LayoutInflater inflater = (LayoutInflater) **context**.getSystemService(Context.***LAYOUT\_INFLATER\_SERVICE***);  
 row = inflater.inflate(**layout**,**null**);  
  
 holder.**textView\_retags1** = (TextView) row.findViewById(R.id.***textView\_retags1***);  
 holder.**textView\_retags2** = (TextView) row.findViewById(R.id.***textView\_retags2***);  
 holder.**textView\_retags3** = (TextView) row.findViewById(R.id.***textView\_retags3***);  
 holder.**imageView2**= (ImageView) row.findViewById(R.id.***imageView2***);  
 holder.**imageView2**.setScaleType(ImageView.ScaleType.***FIT\_XY***);  
 row.setTag(holder);  
 }**else**{  
 holder = (ViewHolder) row.getTag();  
 }  
  
 content content = **contentlist**.get(position);  
  
 holder.**textView\_retags1**.setText(content.getTags1());  
 holder.**textView\_retags2**.setText(content.getTags2());  
 holder.**textView\_retags3**.setText(content.getTags3());  
 holder.**imageView2**.setImageURI(Uri.*parse*(content.getUri()));  
 holder.**imageView2**.setScaleType(ImageView.ScaleType.***FIT\_XY***);  
 **return** row;  
  
 *//layout inflater 는 레이아웃의 동적 할당을 도와주는 역할을 한다  
 //즉, layout inflater는 xml에 미리 정의된 자원 들을 view의 형태로 반환해 준다.  
 //content에서 받아온 정보를 토대로 새로운 view를 동적 할당 해주는 코드이다.  
 //holder 에서는 불러온 view에 content에서 받아온 정보를 넘겨주게 된다* }  
}

**경로 : app -> res -> layout**

**파일 이름 : activity\_main.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**android.support.v4.widget.DrawerLayout 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:id="@+id/drawer\_layout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:fitsSystemWindows="true"  
 tools:openDrawer="start"**>  
  
 <**include  
 layout="@layout/app\_bar\_main"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"** />  
  
 <**android.support.design.widget.NavigationView  
 android:id="@+id/nav\_view"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="match\_parent"  
 android:layout\_gravity="start"  
 android:fitsSystemWindows="true"  
 app:headerLayout="@layout/nav\_header\_main"  
 app:menu="@menu/activity\_main\_drawer"** />  
  
</**android.support.v4.widget.DrawerLayout**>

**경로 : app -> res -> layout**

**파일 이름 : content\_items.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**RelativeLayout  
 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"**>  
  
  
 <**TextView  
 android:id="@+id/textView\_retags3"  
 android:layout\_width="120dp"  
 android:layout\_height="40dp"  
 android:layout\_alignParentEnd="true"  
 android:layout\_alignParentTop="true"  
 android:layout\_marginEnd="20dp"  
 android:layout\_marginTop="120dp"  
 android:text="TextView"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="1.0"  
 app:layout\_constraintStart\_toEndOf="@+id/imageView2"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView\_retags2"  
 app:layout\_constraintVertical\_bias="0.4"  
 android:layout\_alignParentRight="true"  
 android:layout\_marginRight="20dp"** />  
  
 <**TextView  
 android:id="@+id/textView\_retags2"  
 android:layout\_width="120dp"  
 android:layout\_height="40dp"  
 android:layout\_alignParentEnd="true"  
 android:layout\_alignParentTop="true"  
 android:layout\_marginEnd="20dp"  
 android:layout\_marginTop="70dp"  
 android:text="TextView"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.513"  
 app:layout\_constraintStart\_toEndOf="@+id/imageView2"  
 tools:ignore="MissingConstraints"  
 tools:layout\_editor\_absoluteY="75dp"  
 android:layout\_alignParentRight="true"  
 android:layout\_marginRight="20dp"** />  
  
 <**TextView  
 android:id="@+id/textView\_retags1"  
 android:layout\_width="120dp"  
 android:layout\_height="40dp"  
 android:layout\_alignParentEnd="true"  
 android:layout\_alignParentTop="true"  
 android:layout\_marginEnd="20dp"  
 android:layout\_marginRight="10dp"  
 android:layout\_marginTop="20dp"  
 android:text="TextView"  
 app:layout\_constraintBottom\_toTopOf="@+id/textView\_retags2"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toEndOf="@+id/imageView2"  
 app:layout\_constraintTop\_toTopOf="parent"  
 android:layout\_alignParentRight="true"** />  
  
 <**ImageView  
 android:id="@+id/imageView2"  
 android:layout\_width="210dp"  
 android:layout\_height="163dp"  
 android:layout\_alignParentLeft="true"  
 android:layout\_alignParentStart="true"  
 android:layout\_marginLeft="16dp"  
 android:layout\_marginStart="16dp"  
 android:layout\_marginTop="10dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:srcCompat="?attr/colorButtonNormal"** />  
  
 //이 xml파일 내의 view들이 content\_list\_activity의 Gridview내에 하나의 아이템으로써 동적 할당 되어 들어 가게 된다.  
  
</**RelativeLayout**>

**경로 : app -> res -> layout**

**파일 이름 : content\_main.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout 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:orientation="vertical"  
 app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior"  
 tools:context=".MainActivity"  
 tools:showIn="@layout/app\_bar\_main"**>  
  
 <**android.support.constraint.ConstraintLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"**>  
  
 <**Button  
 android:id="@+id/button\_save"  
 android:layout\_width="160dp"  
 android:layout\_height="65dp"  
 android:layout\_marginEnd="8dp"  
 android:layout\_marginLeft="24dp"  
 android:layout\_marginRight="8dp"  
 android:layout\_marginStart="24dp"  
 android:layout\_marginTop="16dp"  
 android:text="저장하기"  
 app:layout\_constraintEnd\_toStartOf="@+id/camera\_button"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/imageView3"** />  
  
 <**Button  
 android:id="@+id/button\_save2"  
 android:layout\_width="143dp"  
 android:layout\_height="41dp"  
 android:layout\_marginBottom="8dp"  
 android:layout\_marginEnd="16dp"  
 android:layout\_marginLeft="16dp"  
 android:layout\_marginRight="16dp"  
 android:layout\_marginStart="16dp"  
 android:text="글그림 저장하기"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.502"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/editText\_tags3"** />  
  
 <**EditText  
 android:id="@+id/editText\_tags1"  
 android:layout\_width="330dp"  
 android:layout\_height="40dp"  
 android:layout\_marginBottom="8dp"  
 android:layout\_marginEnd="8dp"  
 android:layout\_marginLeft="8dp"  
 android:layout\_marginRight="8dp"  
 android:layout\_marginStart="8dp"  
 android:layout\_marginTop="16dp"  
 android:ems="10"  
 android:hint="tag1"  
 android:inputType="textPersonName"  
 app:layout\_constraintBottom\_toTopOf="@+id/editText\_tags2"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/camera\_button"** />  
  
 <**EditText  
 android:id="@+id/editText\_tags2"  
 android:layout\_width="330dp"  
 android:layout\_height="40dp"  
 android:layout\_marginBottom="8dp"  
 android:layout\_marginEnd="8dp"  
 android:layout\_marginLeft="8dp"  
 android:layout\_marginRight="8dp"  
 android:layout\_marginStart="8dp"  
 android:layout\_marginTop="8dp"  
 android:ems="10"  
 android:hint="tag2"  
 android:inputType="textPersonName"  
 app:layout\_constraintBottom\_toTopOf="@+id/editText\_tags3"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/editText\_tags1"** />  
  
 <**EditText  
 android:id="@+id/editText\_tags3"  
 android:layout\_width="330dp"  
 android:layout\_height="40dp"  
 android:layout\_marginBottom="8dp"  
 android:layout\_marginEnd="8dp"  
 android:layout\_marginLeft="8dp"  
 android:layout\_marginRight="8dp"  
 android:layout\_marginStart="8dp"  
 android:layout\_marginTop="8dp"  
 android:ems="10"  
 android:hint="tag3"  
 android:inputType="textPersonName"  
 app:layout\_constraintBottom\_toTopOf="@+id/button\_save2"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/editText\_tags2"** />  
  
 <**Button  
 android:id="@+id/camera\_button"  
 android:layout\_width="160dp"  
 android:layout\_height="65dp"  
 android:layout\_marginEnd="24dp"  
 android:layout\_marginRight="24dp"  
 android:layout\_marginTop="16dp"  
 android:text="사진촬영"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/imageView3"** />  
  
 <**android.support.constraint.Guideline  
 android:id="@+id/guideline"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 app:layout\_constraintGuide\_percent="0.5"** />  
  
 <**ImageView  
 android:id="@+id/imageView3"  
 android:layout\_width="363dp"  
 android:layout\_height="212dp"  
 android:layout\_marginEnd="8dp"  
 android:layout\_marginLeft="8dp"  
 android:layout\_marginRight="8dp"  
 android:layout\_marginStart="8dp"  
 android:layout\_marginTop="8dp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:srcCompat="@android:color/background\_dark"** />  
  
 <**ImageView  
 android:id="@+id/picture\_view"  
 android:layout\_width="348dp"  
 android:layout\_height="199dp"  
 android:layout\_centerInParent="true"  
 android:layout\_marginBottom="8dp"  
 android:layout\_marginEnd="8dp"  
 android:layout\_marginLeft="8dp"  
 android:layout\_marginRight="8dp"  
 android:layout\_marginStart="8dp"  
 android:layout\_marginTop="8dp"  
 android:scaleType="centerInside"  
 app:layout\_constraintBottom\_toBottomOf="@+id/imageView3"  
 app:layout\_constraintEnd\_toEndOf="@+id/imageView3"  
 app:layout\_constraintStart\_toStartOf="@+id/imageView3"  
 app:layout\_constraintTop\_toTopOf="@+id/imageView3"  
 app:srcCompat="@android:color/transparent"** />  
  
 </**android.support.constraint.ConstraintLayout**>  
  
</**LinearLayout**>

**경로 : app -> res -> layout**

**파일 이름 : content\_list\_activity.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**RelativeLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"**>  
  
 <**GridView  
 android:id="@+id/gridView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_alignParentStart="true"  
 android:layout\_alignParentTop="true"  
 android:layout\_marginTop="0dp"  
 android:gravity="center"**></**GridView**>  
 //Grid view란 격자 모양으로 아이템을 배치 할 수 있는 컨테이너 역할 을 해주는 view이다  
 //이 view 안에 격자 모양으로 넣어줄 content에 대해서 content\_items에서 구현 해줄 것이다.  
</**RelativeLayout**>

**경로 : app -> res -> layout**

**파일 이름 : content\_update**

*<?***xml version="1.0" encoding="utf-8"***?>*<**android.support.constraint.ConstraintLayout  
 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"**>  
  
  
 <**ImageView  
 android:id="@+id/imageView\_update"  
 android:layout\_width="346dp"  
 android:layout\_height="274dp"  
 android:layout\_alignParentTop="true"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginEnd="8dp"  
 android:layout\_marginLeft="8dp"  
 android:layout\_marginRight="8dp"  
 android:layout\_marginStart="8dp"  
 android:layout\_marginTop="8dp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:srcCompat="@android:color/darker\_gray"** />  
  
 <**EditText  
 android:id="@+id/editText\_tags1"  
 android:layout\_width="match\_parent"  
 android:layout\_height="45dp"  
 android:layout\_alignParentLeft="true"  
 android:layout\_alignParentStart="true"  
 android:layout\_centerVertical="true"  
 android:layout\_marginTop="8dp"  
 android:ems="10"  
 android:inputType="textPersonName"  
 app:layout\_constraintTop\_toBottomOf="@+id/imageView\_update"  
 tools:ignore="MissingConstraints"  
 tools:layout\_editor\_absoluteX="0dp"** />  
  
 <**EditText  
 android:id="@+id/editText\_tags2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="45dp"  
 android:layout\_alignParentBottom="true"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginTop="8dp"  
 android:ems="10"  
 android:inputType="textPersonName"  
 app:layout\_constraintTop\_toBottomOf="@+id/editText\_tags1"  
 tools:ignore="MissingConstraints"  
 tools:layout\_editor\_absoluteX="0dp"** />  
  
 <**Button  
 android:id="@+id/button\_update"  
 android:layout\_width="match\_parent"  
 android:layout\_height="48dp"  
 android:layout\_alignParentBottom="true"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginBottom="8dp"  
 android:layout\_marginTop="8dp"  
 android:text="Button"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/editText\_tags3"  
 app:layout\_constraintVertical\_bias="1.0"  
 tools:ignore="MissingConstraints"  
 tools:layout\_editor\_absoluteX="0dp"** />  
  
 <**EditText  
 android:id="@+id/editText\_tags3"  
 android:layout\_width="match\_parent"  
 android:layout\_height="45dp"  
 android:layout\_alignParentBottom="true"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginTop="8dp"  
 android:ems="10"  
 android:inputType="textPersonName"  
 app:layout\_constraintTop\_toBottomOf="@+id/editText\_tags2"  
 tools:ignore="MissingConstraints"  
 tools:layout\_editor\_absoluteX="0dp"** />  
</**android.support.constraint.ConstraintLayout**>

**경로 : app -> res -> menu**

**파일 이름 : activity\_main\_drawer.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**menu xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 tools:showIn="navigation\_view"**>  
  
 <**group android:checkableBehavior="single"**>  
 <**item  
 android:id="@+id/nav\_gallery"  
 android:icon="@drawable/ic\_menu\_camera"  
 android:title="사진첩"** />  
 <**item  
 android:id="@+id/nav\_tags"  
 android:icon="@drawable/ic\_menu\_gallery"  
 android:title="태그로 보기"** />  
 <**item  
 android:id="@+id/nav\_slideshow"  
 android:icon="@drawable/ic\_menu\_slideshow"  
 android:title="내용으로 검색"** />  
 <**item  
 android:id="@+id/nav\_manage"  
 android:icon="@drawable/ic\_menu\_manage"  
 android:title="설정"** />  
 </**group**>  
  
 <**item android:title="Communicate"**>  
 <**menu**>  
 <**item  
 android:id="@+id/nav\_share"  
 android:icon="@drawable/ic\_menu\_share"  
 android:title="Share"** />  
 <**item  
 android:id="@+id/nav\_send"  
 android:icon="@drawable/ic\_menu\_send"  
 android:title="Send"** />  
 </**menu**>  
 </**item**>  
  
</**menu**>

**경로 : app -> res -> xml**

**파일 이름 : file\_paths.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**paths xmlns:android="http://schemas.android.com/apk/res/android"**>  
  
 //path로 설정 해준 곳에 파일이 생성 되고 그 안에 이미지가 저장 되어진다  
 <**external-path name="my\_images" path="DCIM"** />  
  
</**paths**>

기능 개발 단계

가.카메라 기능 구현

**-어플리케이션 데이터베이스 업데이트 기능 구현**

**데이터 베이스 업데이트 기능이란?** 데이터 베이스에 저장 된 정보를 바탕으로 그 정보를 새로운 정보로 또는 삭제 함으로써 데이터 베이스의 정보를 바꾸어 주는 것이라 할 수 있다.

|  |
| --- |
| **기능 동작을 위해 필요한 java 파일 및 xml 파일** |
| SQLiteHelper.java클래스 내에서 update와 관련된 쿼리문 메소드를 작성해 준다. |
| Contentlist.java 파일 내에서 형성된 list를 통한 업데이트 기능 접근(dialog기능 이용) |
| Content\_update.xml 파일을 통해 업데이트 화면 UI 구성 |
| Contentlist.java 파일 내의 updateContentlist 메소드를 통한 리스트 최신화 |

|  |
| --- |
| **SQLiteHelper.java클래스 내에서 update 관련 쿼리문 메소드 작성** |
| *//업데이트 기능을 구현해 주기 위한 메소드* **public void** updateData(String tags1, String tags2, String tags3, String uri, **int** id){  SQLiteDatabase database = getWritableDatabase();   String sql = **"UPDATE EXAMPLEn SET tags1 = ?, tags2 = ?, tags3 = ?, uri = ? WHERE id = ?"**;  SQLiteStatement statement = database.compileStatement(sql);   statement.bindString(1, tags1);  statement.bindString(2, tags2);  statement.bindString(3, tags3);  statement.bindString(4, uri);  statement.bindDouble(5, (**double**)id);   statement.execute();  database.close(); } |

|  |
| --- |
| **Content list.java 파일 내에서 형성된 list를 통한 업데이트 기능 접근(dialog기능 이용)** |
| 데이터베이스의 정보가 리스트로 보여지는 화면에 대해서 각각의 리스트를 클릭 하였을때 기능을 동작 하도록 해주는 메소드를 구현 하여 주었다. 기본적으로 사용할 기능의 수에 맞추어 배열을 선언하여 주고 그 배열 원소에 따라 선택 되어진 각각의 기능이 동작 하도록 해준다. 기능은 dialog 뷰를 이용하여 구현 해주고 업데이트 기능 또한 dialog 기능을 이용하여 화면을 띄워 준다. |
| **gridView**.setOnItemClickListener(**new** AdapterView.OnItemClickListener() {  @Override  **public void** onItemClick(AdapterView<?> adapterView, View view, **final int** i, **long** l) {   *//dialog에 띄워 줄 두가지 아이템 이름 배열의 인자로 선언* CharSequence[] items = {**"업데이트"**, **"삭제"**};   *//dialog라는 객체를 만들어 다이얼로그 기능을 띄워준다, 괄호안의 위치가 다이얼로그를 생성해줄 위치* AlertDialog.Builder dialog = **new** AlertDialog.Builder(contentlist.**this**);     dialog.setTitle(**"선택하세요"**);    *//dialog 클릭시 행동에 관한 기능 구현* dialog.setItems(items, **new** DialogInterface.OnClickListener(){  @Override  **public void** onClick(DialogInterface dialog, **int** item){  *//item은 배열 이기 때문에 0 이면 첫번째 인자를 선택한 것이다.***if**(item == 0){  *//update* Cursor c = MainActivity.*sqLiteHelper*.getData(**"SELECT \* FROM EXAMPLEn"**);   c.moveToPosition(i);   *//showDialogUpdate(contentlist.this);* showDialogUpdate(contentlist.**this**);   }**else**{  *//delete* Cursor c = MainActivity.*sqLiteHelper*.getData(**"SELECT id FROM EXAMPLEn"**);  c.moveToPosition(i);   *id2* = c.getInt(0);  showDialogDelete(*id2*);   }  }  });  dialog.show();  } });  *id1* = c.getInt(0); *tags11* = c.getString(1); *tags22* = c.getString(2); *tags33* = c.getString(3); *uri1* = c.getString(4); |
|  |



원래 가지고 있던 내용을

업데이트 화면에 표시해 주어

비교하면서 수정 할 수 있도록 해준다.

**imageView\_update** = (ImageView) dialog.findViewById(R.id.***imageView\_update***);  
**final** EditText editText\_tags1 = (EditText) dialog.findViewById(R.id.***editText\_tags1***);  
**final** EditText editText\_tags2 = (EditText) dialog.findViewById(R.id.***editText\_tags2***);  
**final** EditText editText\_tags3 = (EditText) dialog.findViewById(R.id.***editText\_tags3***);  
Button button\_update = (Button) dialog.findViewById(R.id.***button\_update***);

*id1* = c.getInt(0);  
 *tags11* = c.getString(1);  
 *tags22* = c.getString(2);  
 *tags33* = c.getString(3);  
 *uri1* = c.getString(4);

|  |
| --- |
| **Content list.java 파일 내에서 형성된 list를 통한 업데이트 기능 접근(dialog기능 이용)** |
| showDialogUpdate 메소드를 통해 업데이트를 위한 dialog를 띄워 줄 수 있다. 메소드 안에 들어간 기능으로는 기존의 데이터를 불러와 다시 표시해 줌으로 써 기존의 데이터와 비교 하면서 업데이트가 가능 하다는 점이다. 또한 글을 수정함에 있어서 기존의 데이터의 맨 끝 쪽에 커서를 위치시켜 사용자의 편리성을 도모하였다. 마지막으로 업데이트 할 정보의 수정을 끝마치고 버튼을 눌러 새로운 정보를 데이터 베이스에 저장 시키도록 하였다. |
| **private void** showDialogUpdate(Activity activity){   **final** Dialog dialog = **new** Dialog(activity);  *//content\_update xml 파일 을 dialog기능을 통해 불러온다.* dialog.setContentView(R.layout.***content\_update***);  dialog.setTitle(**"Update"**);   *//set width for dialog* **int** width = (**int**) (activity.getResources().getDisplayMetrics().**widthPixels**\*0.95);  *//set height for dialog* **int** height = (**int**)(activity.getResources().getDisplayMetrics().**heightPixels**\*0.85);  dialog.getWindow().setLayout(width, height);   editText\_tags1.setText(contentlist.*tags11*);  editText\_tags2.setText(contentlist.*tags22*);  editText\_tags3.setText(contentlist.*tags33*);  **imageView\_update**.setImageURI(Uri.*parse*(contentlist.*uri1*));  **imageView\_update**.setScaleType(ImageView.ScaleType.***FIT\_XY***);   *//EditText에서 커서를 가장 끝에 위치시키는 방법*  dialog.show();  *//업데이트 다이얼로그에서 이미지 부분을 클릭할 시 다른 이미지로 변경 할 수 있는 코드* **imageView\_update**.setOnClickListener(**new** View.OnClickListener(){  **public void** onClick(View v){  *//request photo library* Intent intent = **new** Intent(Intent.***ACTION\_PICK***);  intent.setType(MediaStore.Images.Media.***CONTENT\_TYPE***);  intent.setType(**"image/\*"**);  startActivityForResult(intent, ***GALLERY\_CODE***);   }  });  *//새로 작성되거나 지워진 내용을 데이터베이스로 업데이트 시키기 위한 버튼 코드* button\_update.setOnClickListener(**new** View.OnClickListener(){  @Override  **public void** onClick(View v){  **try** {  MainActivity.*sqLiteHelper*.updateData(  editText\_tags1.getText().toString().trim(),  editText\_tags2.getText().toString().trim(),  editText\_tags3.getText().toString().trim(),  *uri1*.toString().trim(),  *id1* );  dialog.dismiss();  Toast.*makeText*(getApplicationContext(),**"업데이트 성공!"**,Toast.***LENGTH\_SHORT***).show();  }  **catch**(Exception error){  Log.*e*(**"Update error"**, error.getMessage());  }  updateContentlist();  }  });  } |
|  |
| **Content\_update.xml 파일을 통해 업데이트 화면 UI 구성** |
| 기본적인 업데이트 기능 화면 구성을 위한 xml파일(각 구성에 Id를 지정해 주어 기능을 부여해 준다.) |
| *<?***xml version="1.0" encoding="utf-8"***?>* <**android.support.constraint.ConstraintLayout  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"**>   <**ImageView  android:id="@+id/imageView\_update"  android:layout\_width="346dp"  android:layout\_height="274dp"  android:layout\_alignParentTop="true"  android:layout\_centerHorizontal="true"  android:layout\_marginEnd="8dp"  android:layout\_marginLeft="8dp"  android:layout\_marginRight="8dp"  android:layout\_marginStart="8dp"  android:layout\_marginTop="8dp"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toTopOf="parent"  app:srcCompat="@android:color/darker\_gray"** />   <**EditText  android:id="@+id/editText\_tags1"  android:layout\_width="match\_parent"  android:layout\_height="45dp"  android:layout\_alignParentLeft="true"  android:layout\_alignParentStart="true"  android:layout\_centerVertical="true"  android:layout\_marginTop="8dp"  android:ems="10"  android:inputType="textPersonName"  app:layout\_constraintTop\_toBottomOf="@+id/imageView\_update"  tools:ignore="MissingConstraints"  tools:layout\_editor\_absoluteX="0dp"** />   <**EditText  android:id="@+id/editText\_tags2"  android:layout\_width="match\_parent"  android:layout\_height="45dp"  android:layout\_alignParentBottom="true"  android:layout\_centerHorizontal="true"  android:layout\_marginTop="8dp"  android:ems="10"  android:inputType="textPersonName"  app:layout\_constraintTop\_toBottomOf="@+id/editText\_tags1"  tools:ignore="MissingConstraints"  tools:layout\_editor\_absoluteX="0dp"** />    <**EditText  android:id="@+id/editText\_tags3"  android:layout\_width="match\_parent"  android:layout\_height="45dp"  android:layout\_alignParentBottom="true"  android:layout\_centerHorizontal="true"  android:layout\_marginTop="8dp"  android:ems="10"  android:inputType="textPersonName"  app:layout\_constraintTop\_toBottomOf="@+id/editText\_tags2"  tools:ignore="MissingConstraints"  tools:layout\_editor\_absoluteX="0dp"** />  <**Button  android:id="@+id/button\_update"  android:layout\_width="match\_parent"  android:layout\_height="48dp"  android:layout\_alignParentBottom="true"  android:layout\_centerHorizontal="true"  android:layout\_marginBottom="8dp"  android:layout\_marginTop="8dp"  android:text="Button"  app:layout\_constraintBottom\_toBottomOf="parent"  app:layout\_constraintTop\_toBottomOf="@+id/editText\_tags3"  app:layout\_constraintVertical\_bias="1.0"  tools:ignore="MissingConstraints"  tools:layout\_editor\_absoluteX="0dp"** />  </**android.support.constraint.ConstraintLayout**> |
| **Contentlist.java 파일 내의 updateContentlist 메소드를 통한 리스트 최신화** |
| 업데이트 기능을 구동 한 후에 새로워진 데이터 베이스를 이용한 list 최신화 기능을 구현 한다. 리스트를 비운 뒤 커서를 이용하여 처음의 데이터베이스부터 다시 리스트를 생성 해준다. |
| **private void** updateContentlist(){  Cursor cursor = MainActivity.*sqLiteHelper*.getData(**"SELECT \* FROM EXAMPLEn"**);   **list**.clear();   **while** (cursor.moveToNext()){   **int** id = cursor.getInt(0);  String tags1 = cursor.getString(1);  String tags2 = cursor.getString(2);  String tags3 = cursor.getString(3);  String uri = cursor.getString(4);   **list**.add(**new** content(tags1, tags2, tags3, uri, id));  }   **adapter**.notifyDataSetChanged(); } |

**-어플리케이션 데이터베이스 삭제 기능 구현**

**데이터 베이스 삭제 기능이란?** 데이터 베이스에 저장 된 정보를 어떠한 요소를 기준으로 선택된 데이터가 기준을 충족 시킬 경우 그 데이터를 삭제 해주는 기능을 말한다.

|  |
| --- |
| **기능 동작을 위해 필요한 java 파일 및 xml 파일** |
| SQLiteHelper.java클래스 내에서 delete와 관련된 쿼리문 메소드를 작성해 준다. |
| Content list.java 파일 내에서 형성된 list를 통한 삭제 기능 접근(dialog기능 이용) |
| Contentlist.java 파일 내의 updateContentlist 메소드를 통한 리스트 최신화 |

|  |
| --- |
| **SQLiteHelper.java클래스 내에서 delete 관련 쿼리문 메소드 작성** |
| **public void** deleteData(**int** id){  SQLiteDatabase database = getWritableDatabase();   String sql = **"DELETE FROM EXAMPLEn WHERE id = ?"**;  SQLiteStatement statement = database.compileStatement(sql);   statement.clearBindings();  statement.bindDouble(1, (**double**)id);   statement.execute();  database.close(); } |

|  |
| --- |
| **Content list.java 파일 내에서 형성된 list를 통한 삭제 기능 접근(dialog기능 이용)** |
| 업데이트 기능과 마찬가지로 삭제 기능 또한 dialog를 이용해 접근 해준다. |
| **gridView**.setOnItemClickListener(**new** AdapterView.OnItemClickListener() {  @Override  **public void** onItemClick(AdapterView<?> adapterView, View view, **final int** i, **long** l) {   *//dialog에 띄워 줄 두가지 아이템 이름 배열의 인자로 선언* CharSequence[] items = {**"업데이트"**, **"삭제"**};   *//dialog라는 객체를 만들어 다이얼로그 기능을 띄워준다, 괄호안의 위치가 다이얼로그를 생성해줄 위치* AlertDialog.Builder dialog = **new** AlertDialog.Builder(contentlist.**this**);    dialog.setTitle(**"선택하세요"**);   *//dialog 클릭시 행동에 관한 기능 구현* dialog.setItems(items, **new** DialogInterface.OnClickListener(){  @Override  **public void** onClick(DialogInterface dialog, **int** item){  *//item은 배열 이기 때문에 0 이면 첫번째 인자를 선택한 것이다.* **if**(item == 0){  *//update* Cursor c = MainActivity.*sqLiteHelper*.getData(**"SELECT \* FROM EXAMPLEn"**);   c.moveToPosition(i);   *id1* = c.getInt(0);  *tags11* = c.getString(1);  *tags22* = c.getString(2);  *tags33* = c.getString(3);  *uri1* = c.getString(4);   *//showDialogUpdate(contentlist.this);* showDialogUpdate(contentlist.**this**);   }**else**{  *//delete* Cursor c = MainActivity.*sqLiteHelper*.getData(**"SELECT id FROM EXAMPLEn"**);  c.moveToPosition(i);   *id2* = c.getInt(0);  showDialogDelete(*id2*);   }  }  });  dialog.show();  }  });   } |
|  |
| **Content list.java 파일 내에서 형성된 showDialogDelete메소드를 통한 삭제 기능 구현(dialog기능 이용)** |
| showDialogDelete 메소드를 구현해 줌으로 써 SQLiteHelper 자바 클래스 내에 구현 된 데이터 베이스 삭제 메소드에 접근 하여 원하는 데이터를 삭제 해줄 수 있다. 데이터를 삭제 하고자 할 때 삭제 하길 원치 않은 경우를 생각 하여 취소 버튼도 항목도 추가 시켜 준다. 삭제가 완료되면 데이터 베이스는 최신화를 통해 현재의 변화 된 데이터베이스를 다시 리스트화 하여 보여준다 |
| **private void** showDialogDelete(**final int** i){  **final** AlertDialog.Builder dialogDelete = **new** AlertDialog.Builder(contentlist.**this**);   dialogDelete.setTitle(**"Warning!!"**);  dialogDelete.setMessage(**"Are you sure you want to this delete?"**);  dialogDelete.setPositiveButton(**"OK"**, **new** DialogInterface.OnClickListener() {  @Override  **public void** onClick(DialogInterface dialog, **int** which) {  **try** {  MainActivity.*sqLiteHelper*.deleteData(i);  Toast.*makeText*(getApplicationContext(), **"삭제 성공!"**, Toast.***LENGTH\_SHORT***).show();  }**catch** (Exception e){  Log.*e*(**"error"**, e.getMessage());  }  updateContentlist();  }  });   dialogDelete.setNegativeButton(**"Cancel"**, **new** DialogInterface.OnClickListener() {  @Override  **public void** onClick(DialogInterface dialog, **int** which) {  dialog.dismiss();  }  });  dialogDelete.show();  } |
|  |

|  |
| --- |
| **Contentlist.java 파일 내의 updateContentlist 메소드를 통한 리스트 최신화** |
| 삭제 기능을 구동 한 후에 새로워진 데이터 베이스를 이용한 list 최신화 기능을 구현 한다. 리스트를 비운 뒤 커서를 이용하여 처음의 데이터베이스부터 다시 리스트를 생성 해준다. |
| **private void** updateContentlist(){  Cursor cursor = MainActivity.*sqLiteHelper*.getData(**"SELECT \* FROM EXAMPLEn"**);   **list**.clear();   **while** (cursor.moveToNext()){   **int** id = cursor.getInt(0);  String tags1 = cursor.getString(1);  String tags2 = cursor.getString(2);  String tags3 = cursor.getString(3);  String uri = cursor.getString(4);   **list**.add(**new** content(tags1, tags2, tags3, uri, id));  }   **adapter**.notifyDataSetChanged(); } |