|  |
| --- |
| 파일에 읽고 쓰기.  내부 저장 공간은 항상 이용 가능.  앱을 설치할 때 해당 앱을 위한 저장공간 할당됨.  이 공간은 해당 앱에서만 접근 가능.  앱이 삭제되며 함께 제거.  외부 저장 공간은 사용하지 못할 수도 있음.  사용할 수 있는지 확인하는 코드 필요.  외부에 공개된 곳이므로 모든 앱이 접근하여 파일 이용 가능.  저장소와 관련된 각종 정보는 Environment 클래스로 얻을 수 있다.  외부 저장 공간 상태, 외부/내부 저장 공간 경로  파일 경로는 문자열로 지정하지 말고, 함수를 이용하여 다르게 대응해야 한다.  외부 저장 공간 순서.  - 외부 저장 공간 이용 가능여부 판단  - 퍼미션 설정  - 파일 읽고 쓰기.  스마트폰 공용 폴더 경로 얻는 함수들. Environment 클래스. |
| 실습9-1.  AndroidManifest.xml 주석 조심.  <!-- -->  <EditText  android:id="@+id/content"  android:layout\_width="match\_parent"  android:layout\_height="0dp"  android:layout\_weight="1"  android:hint="내용을 입력하세요"  android:scrollbars="vertical" /> |
| **// permission 부여 안 된 경우, permission 요청** if(!fileReadPermission || !fileWritePermission){   ActivityCompat.*requestPermissions*(this, new String[]{  Manifest.permission.*READ\_EXTERNAL\_STORAGE*,  Manifest.permission.*WRITE\_EXTERNAL\_STORAGE* }, 200); } |
| **// permission 부여 요청 결과 확인 @Override public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,  @NonNull int[] grantResults) {   super.onRequestPermissionsResult(requestCode, permissions, grantResults);   if(requestCode==200 && grantResults.length>0){   if(grantResults[0] == PackageManager.*PERMISSION\_GRANTED*)  fileReadPermission = true;  if(grantResults[1] == PackageManager.*PERMISSION\_GRANTED*)  fileWritePermission = true;  }  }// end onRequestPermissionsResult** |
| SQLite를 이용한 영속화 :  보통 데이터를 서버에 저장하는데 굳이 로컬 데이터베이스를 사용하는 이유.  서버에서 맏은 데이터를 일정 정도 스마트폰 내부에 저장해야 네트워크 연결이 끊기는 상황에 대응할 수 있다.  이 때문에라도 데이터베이스 프로그램은 자주 이용됨.  안드로이드 데이터베이스 프로그램의 핵심 클래스는 SQLiteDatabase.  openOrCreateDatabase() 함수를 SQLiteDatabase 객체를 얻는다.  데이터베이스에 데이터를 저장하려면 insert 문을 사용한다.  execSQL() 함수로 수행한다.  데이터베이스에 저장된 데이터를 찾아서 가져오려면 select 문을 사용한다.  rawQuery() 함수를 이용한다.  rawQuery() 함수의 결과값은 Cursor 객체.  Cursor는 선택된 행 집합 개념.  열 데이터를 획득하려면 Cursor 객체를 이용해 행을 선택하고,  선택된 행의 열 데이터를 획득하는 구조.  Cursor의 행을 선택하는 함수.  moveToNext() 함수를 이용하여 행을 선택하고, getString() 함수를 이용하여 열 데이터 가져온다.  간단하게 SQLiteDatabase와 Cursor 클래스만 사용해도 모든 SQL 문을 수행할 수 있지만,  SQLiteOpenHelper 클래스를 이용할 수도 있다. 일종의 데이터베이스 관리만을 목적으로 함.  데이터 저장이나 획득은 SQLiteDatabase 클래스를 이용하여 insert(), select() 작업을 하고,  테이블 생성이나 스키마 변경 등의 작업은 SQLiteOpenHelper 클래스로 일원화한다.  이렇게 함으로써 select, insert 작업을 수행할 때 테이블 생성 여부를 판단하지 않아도 된다는 이점이 있다.  SQLiteOpenHelper 클래스는 추상 클래스이므로 서브 클래스를 만들어 사용해야 한다.  SQLiteOpenHelper를 작성하려면 onCreate()와 onUpgrade() 함수를 재정의해야 한다.  onCreate() 함수는 앱이 설치되어 SQLiteOpenHelper 클래스가 최초로 사용되는 순간 호출된다.  앱 내에서 가장 처음 한 번만 수행하면 되는 코드를 삽입하며, 대부분 테이블을 생성하는 코드.  onUpgrade() 함수는 SQLiteOpenHelper 클래스의 생성자에 전달되는 데이터베이스 버전 정보가 변경될 때마다 반복해서 호출된다.  [버전이 변경된다는 말이 뭐지? 내용물이 바뀐다는 의미인가?]  결국, 테이블의 스키마 부분을 변경하기 위한 용도로 사용된다.  단, version이 변경되지 않으면 호출되지 않는다.  SQLiteOpenHelper를 정의하면, SQL 문 수행을 위해 SQLiteDatabase 객체를 획득한다.  이때 SQLiteDatabase의 getReaderableDatabase(), getWritableDatabase()를 이용한다.  SQL 문을 수행하려면 SQLiteDatabase 클래스의 함수를 이용한다.  rawQuery(), execSQL() 이외에  insert(), update(), delete(), query() 함수가 있다.  rawQuery(), execSQL() 함수는 개발자가 직접 SQL 문을 매개변수로 주어야 하지만,  나머지 함수들은 SQL 문을 만들기 위한 정보만 주면 자동으로 SQL문을 만들어 실행해준다.  insert(),update() 함수는 ContentValues 클래스를 매개변수로 전달하는데,  이는 insert,update 데이터를 표현하는 집합 객체이다.  key-value 형식으로 이용되며 key 값이 실제 테이블의 열이다. |
| 실습 8-1.  만약 테이블 생성을 잘못해서 수정해도 onCreate() 함수는 앱 설치 후 최초 한 번만 호출되므로 수정한 부분이 반영되지 않습니다.  이럴 때 DATABASE\_VERSION 값을 증가하면 onUpgrate() 함수가 자동으로 호출되고,  잘못 만든 테이블을 삭제 후 다시 만들게 작성합니다.  [값 증가는 어떻게 하는거지? final이잖아..] |
| **// 버튼을 누르면** DBHelper helper = new DBHelper(this); SQLiteDatabase db = helper.getWritableDatabase();  db.execSQL("insert into tb\_memo (title,content) values (?,?)",  new String[]{title, content}); db.close(); |
| String으로 입력되는 SQL 띄워쓰기 아주 중요하다!! 조심하기 |
| 9.3 SharedPreferences와 앱 설정 자동화  SharedPreferences는 앱의 데이터를 영속적으로 저장하기 위한 클래스.  DBMS 방식의 데이터 영속화는 테이블 구조로 저장하지만,  SharedPreferences는 데이터를 간단하게 키-값으로 저장한다. XML 파일로 저장.  개발자가 직접 파일을 읽고 쓰는 코드를 작성하지 않으므로 개발이 편해진다.  SharedPrefences 객체 획득하는 함수들. 다른 앱 이용 여부, 파일명 지정 등등 차이.  mode값.  SharedPreferences로 데이터를 저장하려면 Editor 클래스 이용.  각 데이터 타입을 위한 함수가 제공되며, 저장한 데이터 최종 반영하려면 commit() 호출.  앱 설정 자동화.  앱의 환경설정을 위해서는 액티비티에서 설정 화면을 구성하고,  그 화면에서 발생하는 다양한 사용자 이벤트를 처리하여 설정한 데이터를 영속적으로 저장해야 한다.  이러한 부분은 대행하주는 것이 앱 설정 자동화.  직접 UI를 구성하지 않고, PerferenceFragment 클래스를 이용하여 쉽게 작성한다.  설정 화면을 위한 xml 파일. res 하위에 xml 폴더.  태그 따로 마련.  xml 태그에 배열 출력 – 배열 리소스는 res/values 폴더를 이용한다. <string-array> 태그.  카테고리로 묶는 태그도 있다.  XML 파일을 묶는 루트 태그. 중첩되어 사용되면 별도의 화면으로 출력.  설정의 서브 화면을 루트 태그 말고 전혀 다른 액티비티로 출력하려면,  <intent>태그 이용한다.  앱 설정 자동화 적용.  구성한 XML 파일 적용하려면 PerferenceFragment를 상속받는 개발자 서브 클래스를 만들어야 한다.  [위에서 앱 설정 자동화를 대행해주는 역할을 한다고 말했던듯]  onCreate() 함수 내에 addPreferencesFromResource() 함수만 이용해도 적용됨.  화면을 보여주려면 Activity 클래스가 필요한데, 다른 클래스를 상속받아 버렸네.  액티비티의 레이아웃 XML 파일에서 <fragment> 태그를 이용하여  PreferenceFragment 서브 클래스를 등록합니다.  name 속성에 서브 클래스의 전체 패키지명을 등록  전체적인 구조는 액티비티가 실행되면서 PreferenceFragment 클래스를 화면에 출력하며  PreferenceFragment 클래스에서 설정을 위한 XML을 적용하는 구조.  코드에서 SharedPreferneces 획득. 이벤트 등록. |
| 10.1 Adapter와 AdapterView  AdapterView는 항목을 나열하는 뷰.  하나의 뷰에 여러 데이터를 나열하고 그중 하나를 사용자에게 선택받는 뷰.  Adapterview들은 구조적으로 라이브러리의 AdapterView를 상속받아 작성된 뷰를 의미한다.  대표ex. ListView, GridView, Spinner 등  AdapterView는 자체로 화면에 나오지 않는다.  뷰 부분이 빈 상태.  Adapter 클래스를 이용해야 화면에 출력된다.  Adapter 클래스는 뷰는 아니지만 AdapterView를 만들어주는 클래스.  화면 출력을 위해 액티비티가 실행되어야 한다.  액티비티가 뷰를 화면에 출력하는 구조.  AdapterView는 액티비티에서 직접 항목을 나열하는 게 아니라,  Adapter에게 일을 시키고, Adapter가 AdapterView를 완성해주는 구조.  결국 Adapteriew를 이용하려면, Adapter를 이용해야 한다.  Adapter 클래스는 Adapter 타입으로 표현되는 클래스.  라이브러리에서 ArrayAdapter, SimpleAdapter, CursorAdapter가 제공된다.  ArrayAdapter는 각 항목에 문자열 데이터를 순서대로 하나씩 나열할 대 사용된다.  액티비티에서 AdapterView-ListView를 화면에 출력하고,  코드에서 Adapter를 적용한다.  AdapterView의 라이브러리 XML이 제공되는 모양.  좀 더 복잡한 항목으로 구성하고 싶으면 직접 만들면 된다.  직접 만든 레이아웃 XML을 이용할 때는 데이터를 출력할 TextView의 id를 전달하면 된다.  이벤트. OnItemClickListener, OnItemLongClickListener  setOnItemClickListener()로 등록.  항목이 동적으로 추가/제거될 때는  Adapter에게 전달한 객체를 변경하고 notifyDataSetChanged() 호출하면 된다.  SimpleAdapter는 한 항목에 문자열 데이터 여러 개 나열할 때 이용.  ArrayAdapter는 데이터 구성을 문자열 배열이나 ArrayList<String>으로도 가능하지만,  SimpleAdapter는 ArrayList<HashMap<String, String>>으로 구성한다.  CursorAdapter 클래스는,  안드로이드 DBMS 프로그램의 select 결과값을 그대로 이용해 항목을 구성한다.  Cursor 객체를 CursorAdapter에게 넘겨주면 알아서 행의 열 데이터를 추출해서 정의된 규칙대로 항목을 구성한다.  데이터베이스에서 항목 데이터를 획득해 목록을 구성할 때 편리하다. |
| Context 객체.  Context 객체를 안드로이드 기본 객체.  Activity, Service는 Context의 하위 클래스.  Context 클래스는 안드로이드 시스템의 글로벌 정보나 애플리케이션 환경 등에 접근하는 방법을 제공한다.  애플리케이션 레벨의 코드에서 수행되는 기능이 아니라,  시스템 레벨에서 수행되는 기능을 지원한다.  결국, 애플리케이션 레벨에서 시스템 레벨의 기능 및 정보를 이용해야 할 때,  Context 객체를 이용한다. |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |