Broadcast, Content Provider

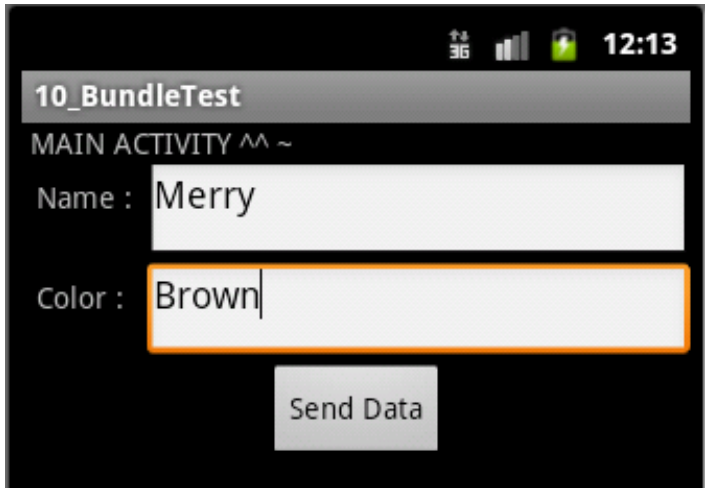
Bundle, Context

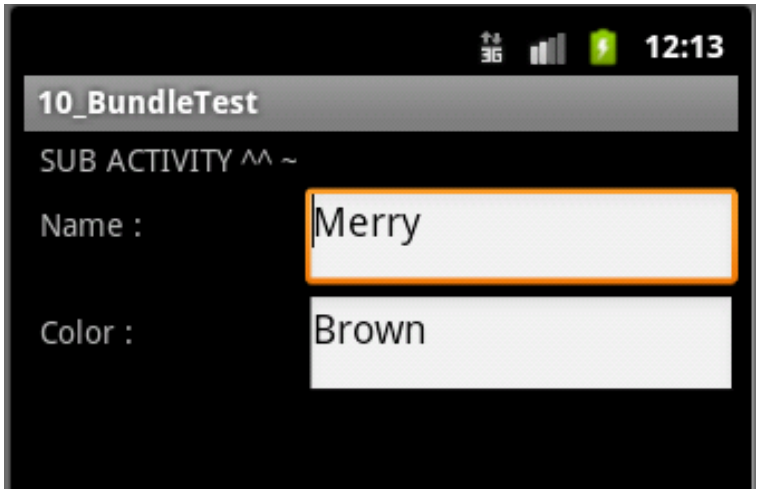
Bundle Class

- 문자열로 된 키(key)와 여러 가지 타입의 값(value)를 저장하는 일종의 Map 클래스 (Bundle = key + value)
- Activity간에 데이터를 주고 받을 때 Bundle 클래스를 사용함
- Intent의 엑스트라 Extras에 필요한 값을 입력하면 안드로이드는 엑스트라값을 번들(bundle) 객체로 변화시킴

- 기본 타입인 int, double, long, String 부터 FloatArray, StringArrayList, Serializable, Parcelable을 구현한 객체를 전송함
- http://developer.android.com/reference/android/os/Bundle.html
- get, getBoolean, getBundle, getByte, getByteArray, .. getParcelable, getSerializable … 등 다양함
- Serializable : Serializable을 구현한 객체를 주고 받을 수 있음
- Parcelable : Parcelable을 구현한 객체를 주고 받을 수 있음

10_BundleTest





10_BundleTest

```
// Bundle 사용
Intent i = new Intent(BundleTestActivity.this, SubActivity.class);
Bundle extras = new Bundle();
extras.putSerializable("mydog1", dog);
i.putExtras(extras);
startActivity(i);

// Bundle 미사용
Intent i = new Intent(BundleTestActivity.this, SubActivity.class);
i.putExtra("mydog1", dog);
startActivity(i);
```

10_BundleTest

```
// Bundle 사용
Bundle extras = this.getIntent().getExtras();
Dog dog = (Dog)extras.getSerializable("mydog1");

// Bundle 미사용
Intent i = getIntent();
Dog dog = (Dog)i.getSerializableExtra("mydog1");
```

Context class, java.io 패키지 이해

public abstract class
Context

extends Object

java.lang.Object
  ↳ android.content.Context

▶Known Direct Subclasses
ContextWrapper, MockContext

abstract File	getDatabasePath (String name) Returns the absolute path on the filesystem
abstract File	getDir (String name, int mode) Retrieve, creating if needed, a new directory
abstract File	getExternalCacheDir () Returns the absolute path to the directory or can place cache files it owns.
abstract File	getExternalFilesDir (String type) Returns the absolute path to the directory or can place persistent files it owns.
abstract FileInputStream	openFileInput (String name) Open a private file associated with this Context's application package for reading.
abstract FileOutputStream	openFileOutput (String name, int mode) Open a private file associated with this Context's application package for writing.
abstract SQLiteDatabase	openOrCreateDatabase (String name, int mode, SQLiteDatabase.CursorFactory factory) Open a new private SQLiteDatabase associated with this Context's application package.

Context Class

- 어플리케이션 환경에 관한 글로벌 정보를 접근하기 위한 인터페이스
- Abstract 클래스이며 실제 구현은 안드로이드 시스템에 의해 제공됨
- Context의 역할
  - ✓ 어플리케이션에 관하여 시스템이 관리하고 있는 정보에 접근하기
  - ✓ 안드로이드 시스템 서비스에서 제공하는 API 호출 기능
    (Activity 실행, Intent 브로드캐스팅/수신)
- 안드로이드에서는 어플리케이션과 관련된 정보에 접근하고자 할 때 ActivityManagerService를 통해야만 하며, 어떤 어플리케이션인지에 관한 키 값도 필요함
- 안드로이드 플랫폼상에서의 관점에서의 Context의 역할
  - ✓ 자신이 어떤 어플리케이션을 나타내고 있는지 알려주는 ID 역할
  - ✓ ActivityManagerService에 접근할 수 있도록 하는 통로 역할

Context Class

C# 에서 C#에서 (일반 OS) 어플리케이션 = 프로세스

```
// Get an Application Name
String applicationName =
    System.AppDomain.CurrentDomain.FriendlyName;
// Start a new process(application)
System.Diagnostics.Process.Start("test.exe");
```

Android의 Activity에서 어플리케이션과 프로세스는 독립적 존재

```
// Get an Application Name
String applicationName = this.getPackageName();
// Start a new process(application)
this.startActivity(new Intent(this, Test.class));
```

Context Class

- startActivity();
- startService();
- registerReceiver();
- sendBroadcast();

Broadcast Receiver

Broadcast Receiver

- Application 외부에서 전달되는 Message를 수신하는 용도
- 외부(System, 다른 Application)에서 전달되는 Message에 대해 적절한 대응을 하면 됨

public abstract class

BroadcastReceiver

extends Object

java.lang.Object
  ↳ android.content.BroadcastReceiver

▶Known Direct Subclasses
AppWidgetProvider, DeviceAdminReceiver

Broadcast Receiver 사용 방법

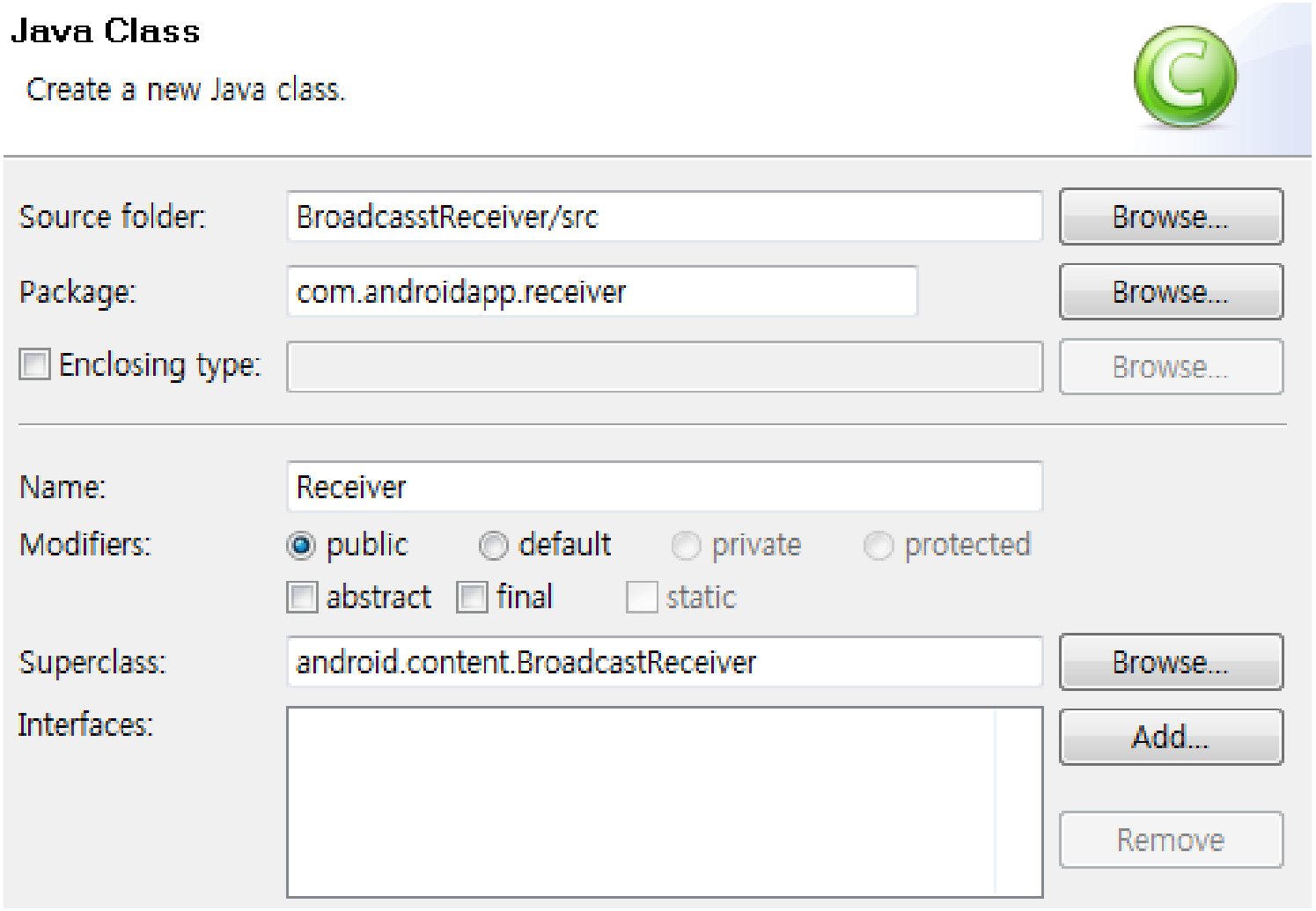
1. BroadcastReceiver class 상속해서 구현
   - Application 요소가 됨, AndroidManifest.xml에 등록해서 사용 함
2. BroadcastReceiver class Instance 생성
   - Application 요소가 아님, AndroidManifest.xml에 등록할 필요 없음

abstract void	onReceive (Context context, Intent intent) This method is called when the BroadcastReceiver is receiving an Intent broadcast.

BoradcastReceiver class에서 **onReceive** method는 외부에서 전달되는 Message를 수신했을 때 호출되는 **Callback Method.**

주의 : onReceive()내에서 처리하는 시간이 10초 이내 처리되는 내용이어야 함, ANR error 발생됨 시간이 많이 걸리는 동작은 다른 곳에서 처리하도록 해야 함, 모든 application은 5초 내에 반응이 있어야 하며, 반응이 없으면 ANR error 발생

외부에서 문자메시지 왔을 때 Toast로 왔다는 것을 알리는 법



문자 알림 NewSMSCheck

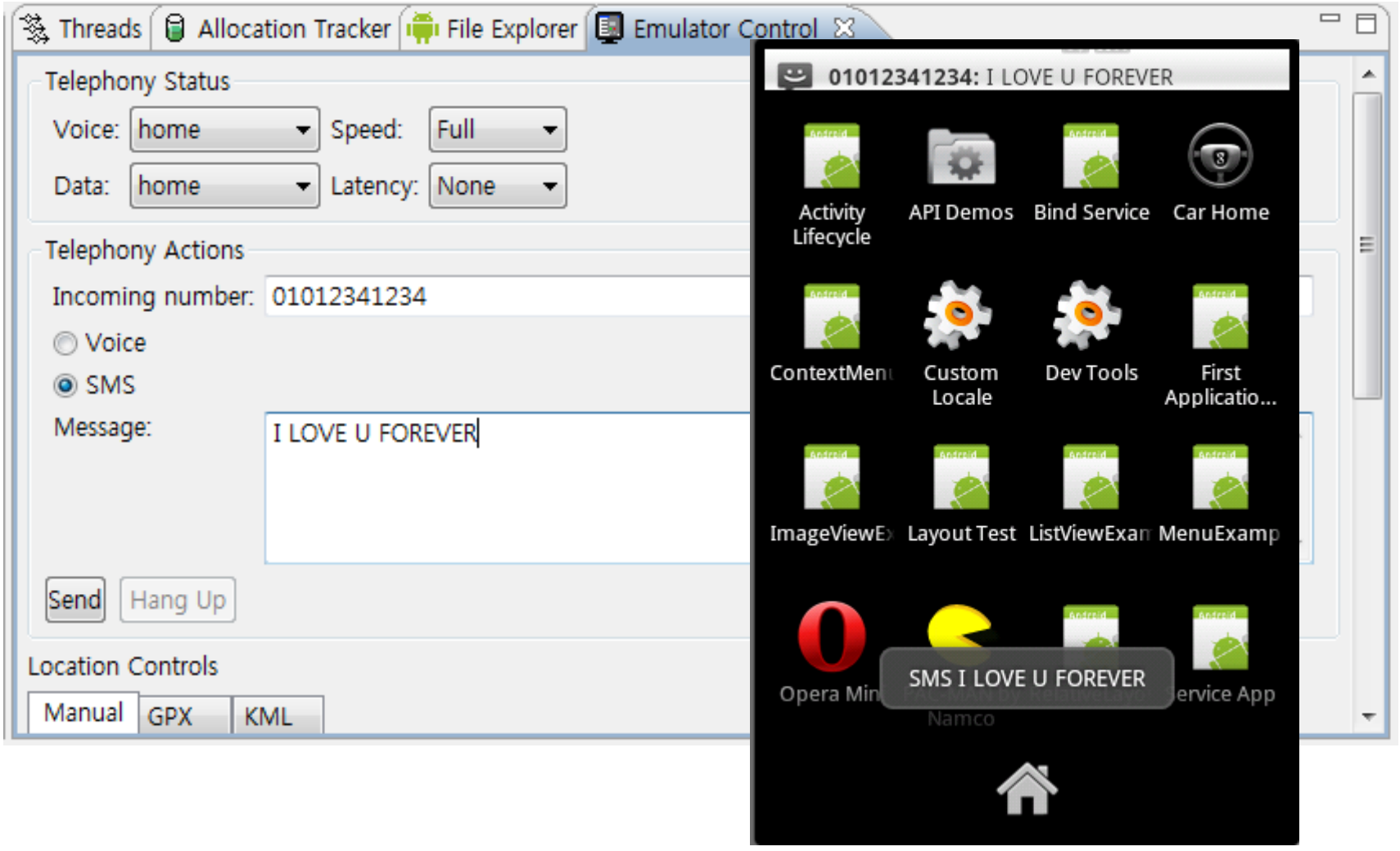
```
public class NewSMSCheck extends BroadcastReceiver {

    @Override
    public void onReceive(Context arg0, Intent arg1) {
        // TODO Auto-generated method stub
        Bundle bundle = arg1.getExtras();
        Object message[] = (Object[]) bundle.get("pdus");
        SmsMessage sms[] = new SmsMessage[message.length];
        int n=0;
        for(Object msg : message){
            sms[n++] = SmsMessage.createFromPdu((byte[])msg);
        }
        Toast.makeText(arg0, "SMS " +
        sms[0].getMessageBody(), Toast.LENGTH_LONG).show();
    }
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.androidapp.receiver"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon"
          android:label="@string/app_name">
        <receiver android:name=".NewSMSCheck
                android:enabled="true"
                android:label="@string/app_name">
            <intent-filter>
                <action
                    android:name=" android.provider.Telephony.SMS_RECEIVED"/>
            </intent-filter>
        </receiver>
    </application>
    <uses-permission android:name="android.permission.RECEIVE_SMS" />
</manifest>
```

DDMS – Emulator Control



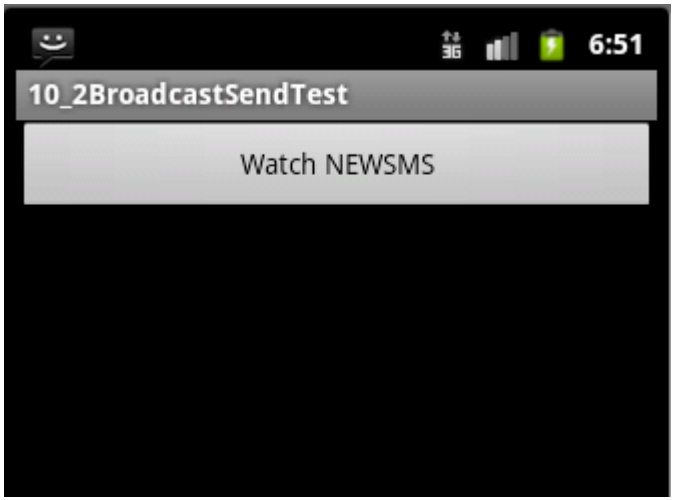
Broadcast Receiver 사용 방법

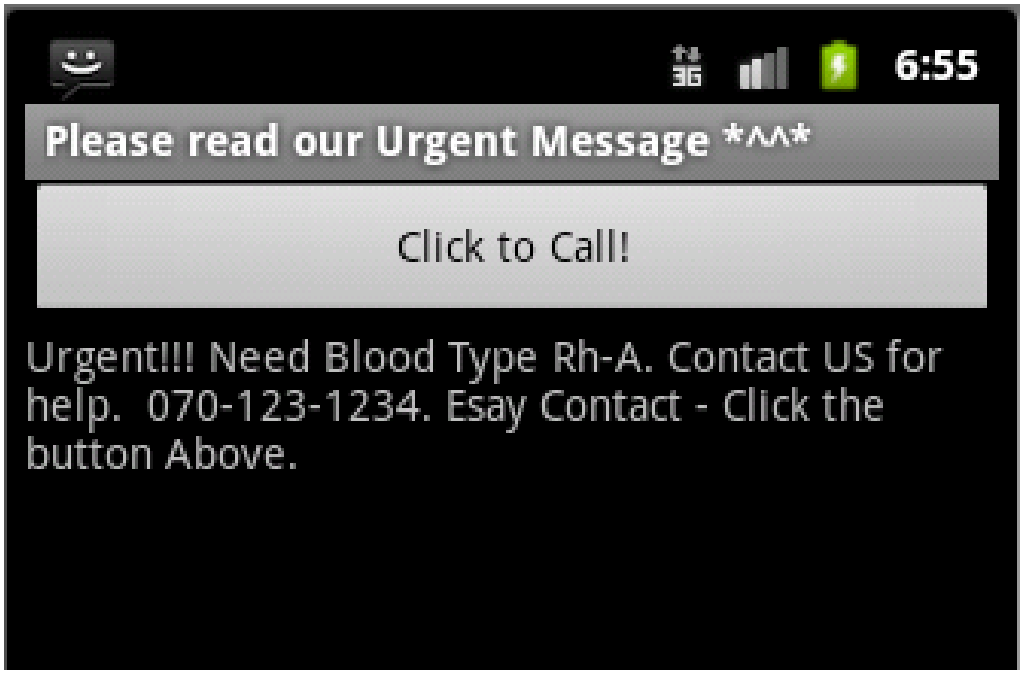
- BR은 메인 스레드에서 실행됨 (10초내로 onReceiver가 return해야 함)
- 방송은 시스템이 보내고 응용 프로그램은 수신만 하는 것이 일반적
- 만일 응용 프로그램도 특별한 변화를 유발시켰다거나 특이한 변화를 발견했다면 다른 응용 프로그램에게 방송을 보낼 수 있음
- **응용 프로그램이 방송을 할 때 사용하는 메서드**
  - ✓ void sendBroadcast(Intent intent [String receiverPermission])
  - ✓ void sendOrderedBroadcast (Intent intent, String receiverPermission)

  intent : 전달하고자 하는 방송 내용, 액션에 방송의 주 내용을 대입

  receiverPermission : 허가 받은 수신자에게만 방송을 보내고 싶을 때 사용, 필요 없을 경우 생략하거나 null로 전달함

ShowBR (Broadcast Recever)





- MainActivity.java → 방송을 보냄
- ShowBR → 방송을 받으면 SubActivity.java를 실행함 (화면 표시)

ShowBR (Broadcast Recever)

```
//  MainActivity.java 에서 버튼 클릭시
public void onClick(View v) {
      Intent i = new Intent();
      i.setAction("com.androidapp.broadcast.CALLBOARD");
      i.putExtra("brmsg", "Urgent!!! Need Blood Type Rh-A. Contact US
          for help.  070-123-1234. Esay Contact - Click the button
          Above.");
      sendBroadcast(i);
}
```

ShowBR (Broadcast Recever)

```
// ShowBR.java에서
public class ShowBR extends BroadcastReceiver {
    public void onReceive(Context context, Intent i) {
        if(context != null){
          Intent backintent=new Intent("com.android.broadcast.SHOWBR");
          backintent.putExtra("showmsg", i.getStringExtra("brmsg"));
          backintent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
          // Falgs 등록 필수 사항!!!
          context.startActivity(backintent);
        }
    }
}
```

ShowBR (Broadcast Recever)

```
// SubActivity.java에서
public class SubActivity extends Activity {

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.sub);
        TextView t = (TextView)findViewById(R.id.board);
        Intent i = getIntent();
        t.setText(i.getStringExtra("showmsg"));
    }
}
```

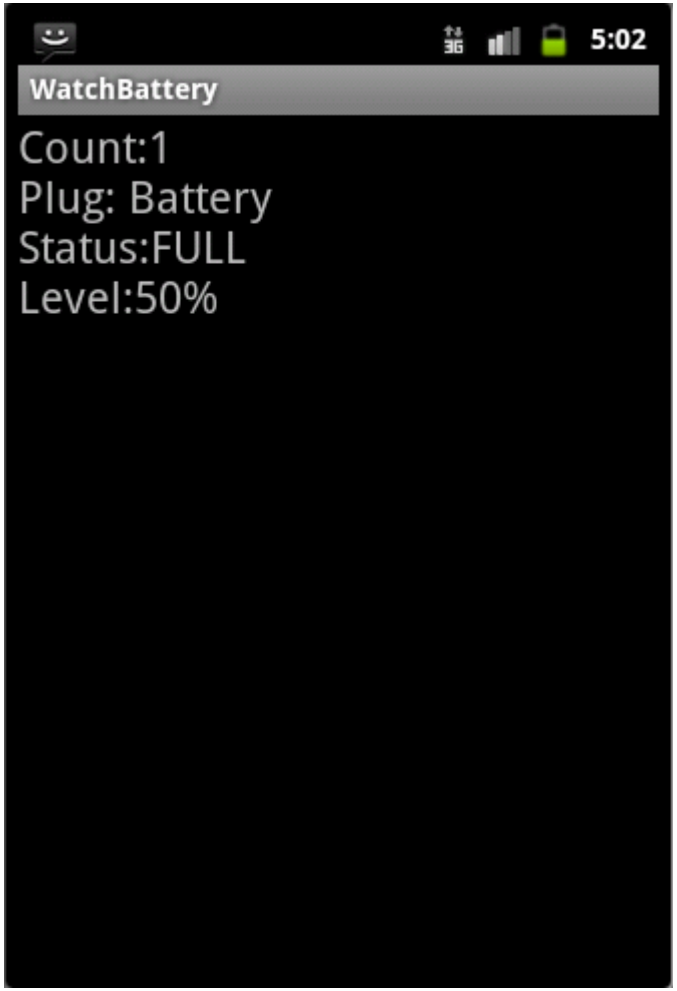
ShowBR (Broadcast Recever)

```
// Manifest
 <receiver android:name=".ShowBR"   android:enabled="true">
    <intent-filter>
        <action android:name="com.androidapp.broadcast.CALLBOARD" />
    </intent-filter>
 </receiver>
 <activity android:name=".SubActivity"    android:label="@string/app_name">
    <intent-filter>
         <action android:name="com.android.broadcast.SHOWBR" />
         <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
 </activity>
```

Broadcast Receiver 사용 방법

- BR은 보통 매니패스트에 등록해 두며 이 경우 BR은 설치 직후부터 항상 방송을 수신 대기하고 있는 것이며, **액티비티가 실행중인 동안만 방송을 수신하려면 코드에서 BR을 일시적으로 등록**해 놓을 수 있음 (매니페이스에 등록할 필요 없이 메서드로 등록 및 해제 함)
  - ✓ Intent registerReceiver (BroadcastReceiver r, IntentFilter f)
  - ✓ void unregisterReceiver (BroadcastReceiver receiver)
  - registerReceiver로 BR객체와 IntentFilter를 전달함
  - BR 등록은 onResume에 하고, 해제는 onPause에서 함

WatchBattery



- Battery의 상태를 파악하여 표시함
- Activity가 종료되면서 동작 종료됨
- telnet을 이용하여 power에 대한 상태를 조절함

telnet 접속 & Battery 관련 명령 사용 방법

command 창에서
> telnet localhost 5554
접속 후
> power capacity n // 배터리 레벨 변경 0~100 사이 값
> power ac on/off // 외부 전원 연결 / 해제
> power status 상태 // 배터리 상태 변경 (charging, discharging, not-charging, full, unknown 등)
> power health // 배터리 성능 조사
> power present true/false // 배터리 탈부착
> power display // 배터리 현재 상태 조사 실행

// telnet 프로그램 다운로드
http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

WatchBattery (Broadcast Recever)

```
public void onResume() {
	super.onResume();
	IntentFilter filter = new IntentFilter();
	filter.addAction(Intent.ACTION_BATTERY_CHANGED);
	filter.addAction(Intent.ACTION_BATTERY_LOW);
	filter.addAction(Intent.ACTION_BATTERY_OKAY);
	filter.addAction(Intent.ACTION_POWER_CONNECTED);
	filter.addAction(Intent.ACTION_POWER_DISCONNECTED);
	registerReceiver(mBRBattery, filter);
}
public void onPause() {
	super.onPause();
	unregisterReceiver(mBRBattery);
}
```

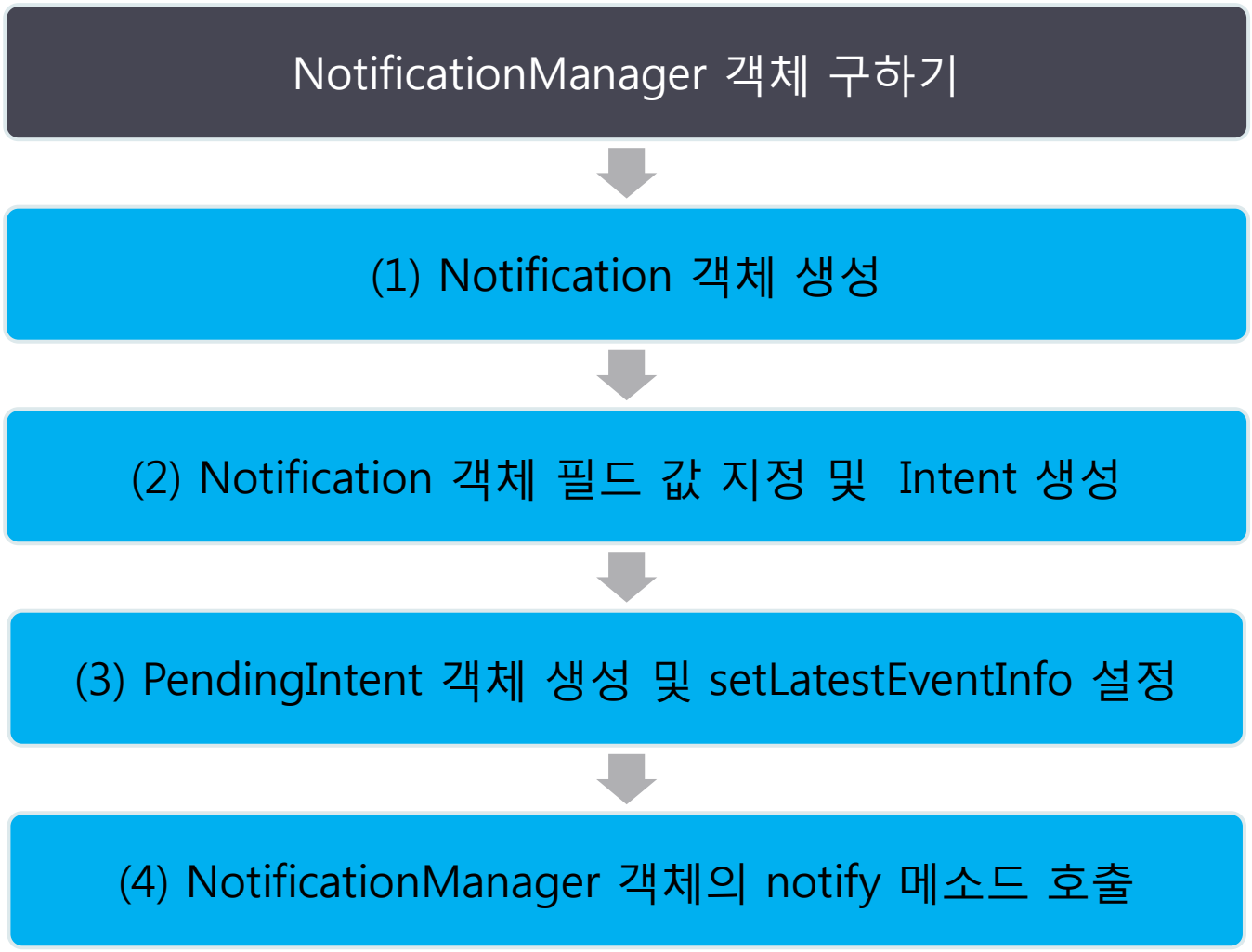
WatchBattery (Broadcast Recever)

```
BroadcastReceiver mBRBattery = new BroadcastReceiver( ) {
    //  배터리 상태를 파악하고 Toast로 표시하는 코드
    public void onReceive(Context context, Intent intent) {
    …
    }
};
```

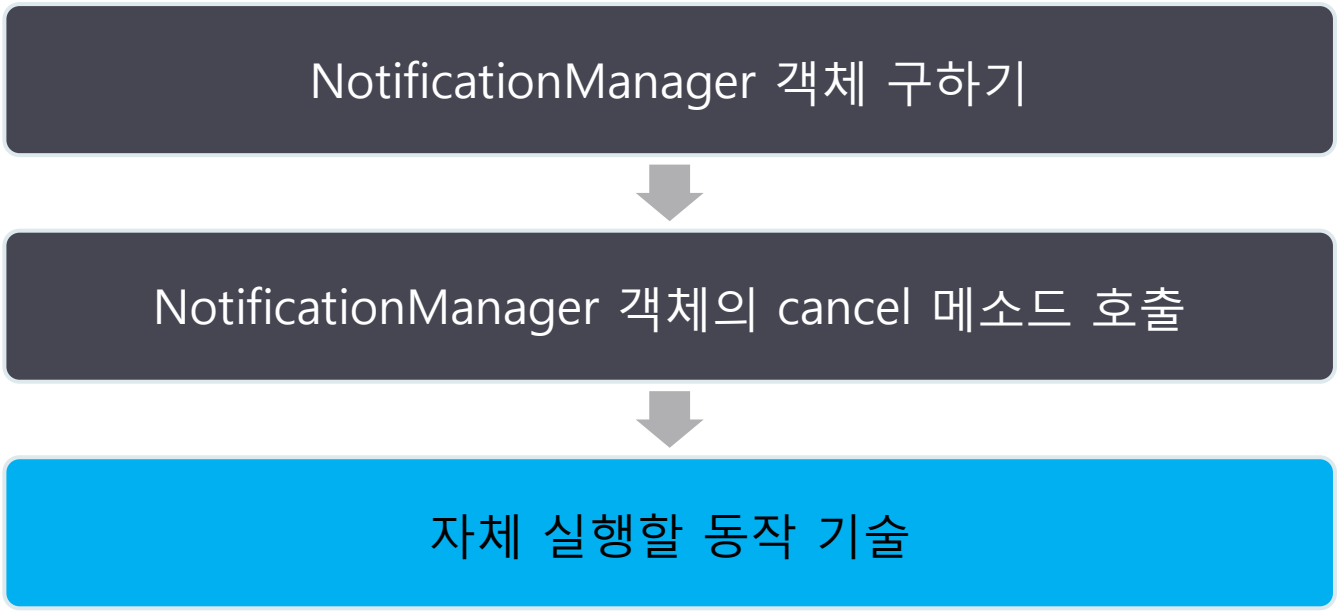
Notification

Notification

준비사항 : 상태줄에 표시할 이미지



Notification 호출로 연결된 구성요소에서의 처리



Notification의 이해

- 백그라운드 프로세스가 사용자와 통신할 수 있는 방법
- 통지는 최초 잠깐만 보이지만 토스트와 달리 사용자가 확인하기 전에는 아이콘이 계속 표시됨
- 소리, 진동, 불빛 같은 방법으로 사용자에게 신호를 보낼 수 있음
- 상태란: 아래로 드래그하여 확장하거나 Home 화면의 메뉴에서 Notification 항목을 선택하면 통지에 대한 상세한 정보가 출력됨
- 통지 출력을 위해서는 통지 관리자(NotificationManager)와 통지 객체(Notification)를 사용해야 함

Notification 생성자

Notification(int icon, CharSequence tickerText, long when)

- icon : 상태란에 표시될 작은 그림
- tickerText : 통지 영역에 아이콘이 처음 나타날 때 잠시 출력될 짧은 문자열
- when : 통지가 발생한 시간 지정. (System.currentTimeMillis 메서드로 구한 현재 시간을 지정하는 것이 일반적)
- 생성자로 객체를 생성한 후, 실행 중에도 icon, tickerText, when 등을 변경할 수 있음

Notification의 추가 field

필드	설명
number	통지 아이콘에 겹쳐서 출력될 숫자 지정. (예) 새로운 메시지 도착 통지라면 메시지의 개수를 같이 표시할 수 있음 (1 이상의 값만 설정 가능)
sound	통지와 함께 출력할 소리를 Uri 객체로 지정
vibrate	진동 방식 지정 (진동 시간, 멈춤 시간을 배열로 전달)
ledARGB	불빛의 색상 지정 (장비에 장착된 LED의 능력에 따라 표현 가능한 색상은 조금 달라질 수 있음)
ledOnMs, ledOffMs	LED를 켤 시간과 끌 시간을 1/1000초 단위로 지정함. LED의 점멸 주기를 결정하며, 정확하지는 않지만 장비는 가급적 근접한 시간을 지킴
default	디폴트로 취할 통지 전달 방식 지정
flags	통지의 동작 방식 지정

Notification의 default/flag 필드에 사용할 수 있는 플래그

플래그	설명
DEFAULT_SOUND	소리를 발생시킴
DEFAULT_VIBRATE	진동을 발생시킴
DEFAULT_LIGHTS	불빛을 깜박거림
DEFAULT_ALL	위 세가지 동작을 모두 수행함

플래그	설명
FLAG_AUTO_CANCEL	사용자가 아이콘을 탭하면 통지 취소
FLAG_INSISTENT	취소하거나 상태란을 확장하기 전까지 소리 계속 발생
FLAG_NO_CLEAR	사용자가 clear all 을 선택할 때 취소
FLAG_ONGOING_EVENT	계속 진행중인 이벤트를 참조함
FLAG_ONLY_AFTER_ONCE	이전에 취소된 통지라도 매번 소리, 진동 발생
FALG_SHOW_LIGHTS	LED 불빛 출력

setLatestEventInfo() 메서드

void setLatestEventInfo (Context context, CharSequence contentTitle, CharSequence contentText, PendingIntent contentIntent)

- 속성 설정 후, 확장 상태란에 표시될 정보와 사용자가 통지 객체를 선택했을 때의 반응을 지정함
- 상태란을 확장해야만 보이며, 생략할 수 없음
- 상태란에 아이콘만 배치하는 방법은 지원하지 않으며, 확장 상태란과 선택시의 동작까지도 반드시 지정해야 함
- content : 통지를 발생시킨 주체
- contentTitle : 제목
- contentText : 메시지 내용 문자열
- contentIntent : 통지 뷰를 탭 했을 때 호출할 인덴트 지정

PendingIntent 클래스

- Intent를 Wrapping함
- 보통의 Intent와의 차이점 : 다른 응용 프로그램으로 전달하여 실행 권한을 주는 Intent라는 것
- 시스템이 관리, Intent를 만든 응용 프로그램이 종료되어도 유효함
- 생성자가 정의되어 있지 않아 객체 생성 메서드를 사용해 객체 생성

  PendingIntent getActivity(Context context, int requestCode,
  Intent intent, int flags)

  PendingIntent getBroadcast(Context context, int requestCode,
  Intent intent, int flags)

  PendingIntent getService(Context context, int requestCode,
  Intent intent, int flags)
- intent : 사용자가 통지 객체를 탭 했을 때의 동작을 지정하며, 주로 Activity를 띄우는데 이 경우 intent에 FLAG_ACTIVITY_NEW_TASK 플래그를 지정해야 함, Service는 사용자의 반응을 처리할 별도의 Activity를 준비해 두어야 함

통지 관리자(NotificationManager)로 등록

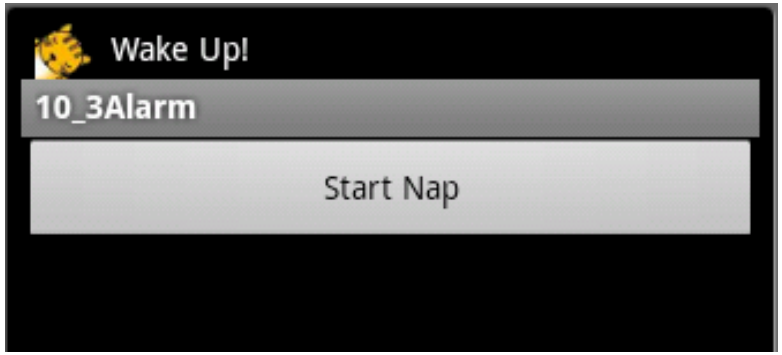
- 통지 관리자는 시스템이 제공하는 서비스
- 객체를 직접 생성할 필요 없이 getSystemService(NOTIFICATION_SERVICE)를 이용하여 구함
- 통지 관리자의 메서드
  - void notify (int id, Notification notification) : 통지
  - void cancel (int id) : 취소
  - void cancelAll ()

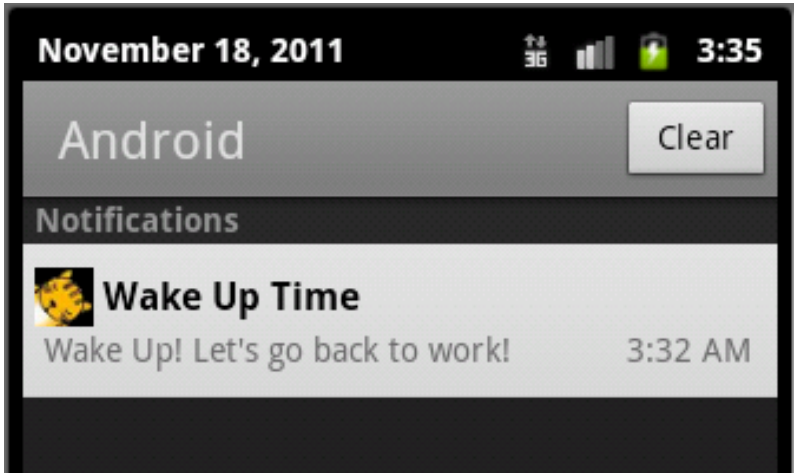
  id : 통지 객체의 고유 식별 번호 (한 프로그램이 여러 개의 통지를 동시에 보낼 수 있기 때문에 중복 되지 않도록 ID 부여), id는 아이콘을 변경하거나 취소할 때 사용됨

  notification : 미리 준비해 둔 통지 객체

통지 출력 (Project : 10_3Alarm)

- AlarmActivity.java : 알람 통지를 수행하는 Activity
- NapEnd.java : 알람 통지를 취소하는 Activity
- nap.png
- main.xml
- napend.xml





통지 출력 (Project : 10_3Alarm)

- 소리와 함께 진동을 설정할 때
- noti.defaults |= (Notification.DEFAULT_SOUND | Notificationn.DEFAULT_VIBRATE);
- custom vibrate - DEFAULT_VIBRATE인 경우 무시됨
- noti.vibrate = new long[] {1000,1000, 500, 500, 200, 200, 200, 200, 200, 200};
- Manifest에 permission필요

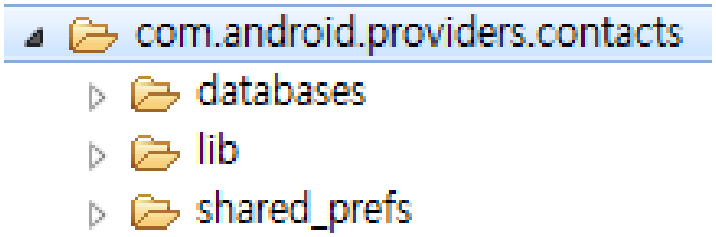
```
<uses-permission adnroid:name="android.permission.VIBRATE" />
```

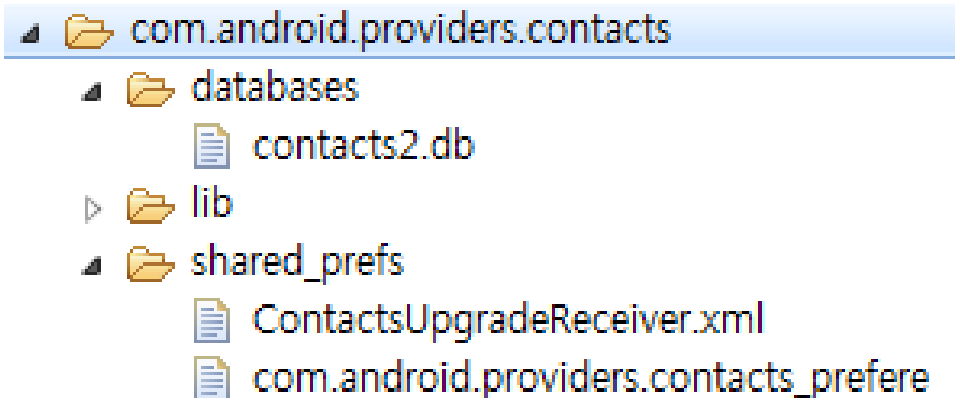
Data 관리

Data 관리

1. 환경설정 Data 관리 – preperence
   /data/data/<package>/shared_prefs
   위의 디렉터리에 환경설정 Data가 저장됨
2. 일반 Data 관리 – File -> java.io
   /data/data/<package>/files
3. Database – SQLite(관계형 DB 엔진, 무료 – iphone에서도 사용)
   /data/data/<package>/databases

일반 Data : 형식이 정해져 있지 않은 Data





Preference

- 응용 프로그램의 설정 정보를 영구적으로 저장하는 도구
- 윈도우에서의 레지스트리나 리눅스의 환경 셋팅 파일에 대응되는 개념
- XML 포맷의 텍스트 파일에 정보를 저장함
  : XML 형식 (Key(name) + 내용)
- SharedPreferences 클래스 사용
- 응용 프로그램내의 모든 액티비티가 공유함
- 외부(다른 Application)에서는 사용할 수 없음
- 내용으로 사용 가능한 Type이 제한되어 있음
  사용 가능 Type : boolean, int, float, long, String

SharedPreferences 객체 생성 메소드 2가지

<<Application 내 모든 Activity가 공유>>

SharedPreferences get**Shared**Preferences(String name, int mode)

name : Preferences를 저장한 XML 파일의 이름

mode : 파일의 공유 모드로 '0'이면 읽기 쓰기가 가능,

MODE_WORLD_READABLE : 읽기공유 (외부에서도 읽기 가능)

MODE_WORLD_WRITEABLE : 쓰기공유 (외부에서도 쓰기 가능)

MODE_PRIVATE : 혼자만 사용하는 배타적 모드로 파일 생성

MODE_APPEND: 파일 존재할 경우, 추가 모드로 열기

<<하나의 Activity에서만 사용>>

SharedPreferences getPreferences(int mode)

파일 이름이 없음, 액티비티의 이름과 같은 XML 파일이 생성됨

public interface SharedPreferences의 주요 메서드 (읽기용)

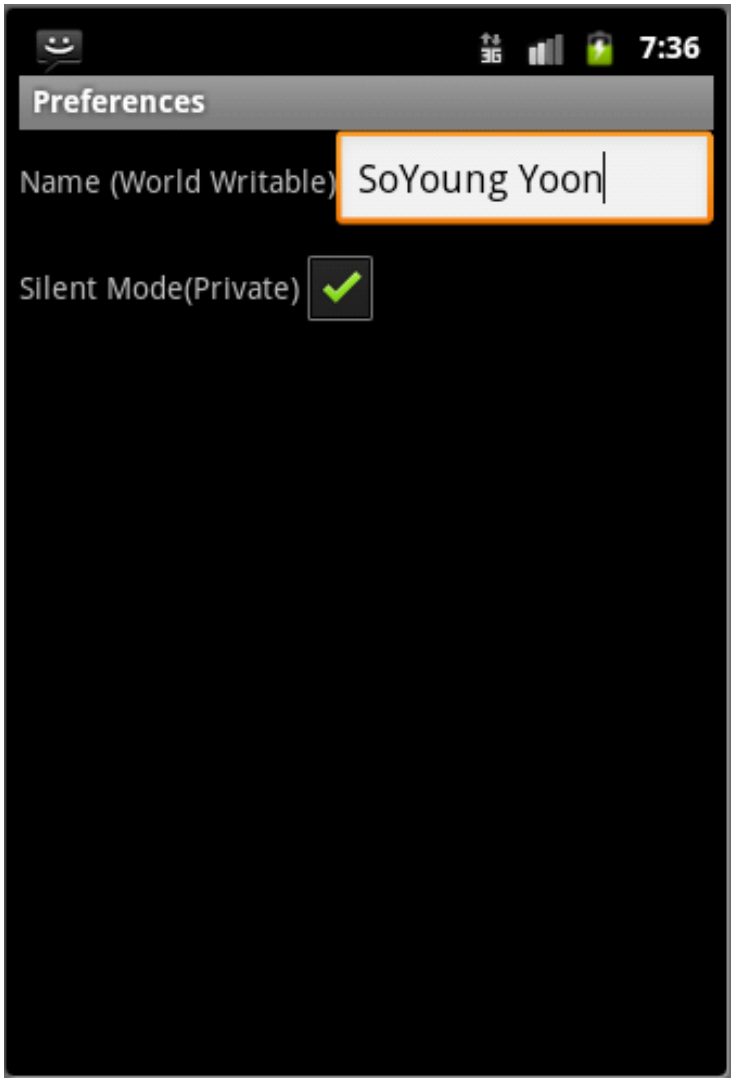
abstract Map<String, ?>	getAll () Retrieve all values from the preferences.
abstract boolean	getBoolean (String key, boolean defValue) Retrieve a boolean value from the preferences.
abstract float	getFloat (String key, float defValue) Retrieve a float value from the preferences.
abstract int	getInt (String key, int defValue) Retrieve an int value from the preferences.
abstract long	getLong (String key, long defValue) Retrieve a long value from the preferences.
abstract String	getString (String key, String defValue) Retrieve a String value from the preferences.

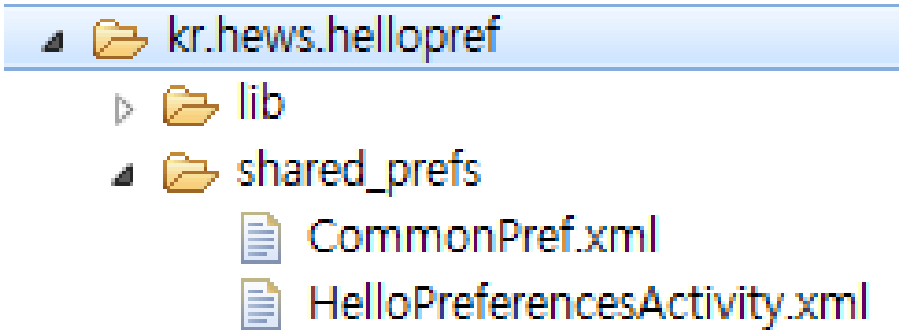
SharedPreferences.Editor 의 주요 메서드 (쓰기용)

SharedPreferences.Editor putInt(String key, int value);
SharedPreferences.Editor putBoolean(String key, boolean value);
SharedPreferences.Editor putString(String key, String value);
SharedPreferences.Editor remove(String key);
boolean commit();
SharedPreferences.Editor clear();

- putXX 메서드를 이용해 값을 저장한 후,
  반드시 **commit 메서드를 호출해야 실제 파일에 저장됨**

11_1HelloPreferences(Preference Test)





- 실행 후 변경한 값이 항상 저장됨
- 정적으로 저장되는 것 (xml 파일)
- 이름저장 : CommonPref.xml
- 모드저장 : HelloPreferencesActivity.xml

11_1HelloPreferences(Preference Test))

```
//  읽어 오기   (onResume()에 입력)
public static final String PACKAGE_NAME = "kr.hews.hellopref";
public static final String COMMON_PREF = "CommonPref";
public static final String NAME = "Name";
private static final String SILENT_MODE = "SilentMode";
// 외부에서 액세스 할 수 있는 Preferences에서 값 검색
      SharedPreferences common = getSharedPreferences(
      COMMON_PREF, MODE_WORLD_READABLE |
          MODE_WORLD_WRITEABLE);
      EditText name = (EditText)findViewById(R.id.name);
      name.setText(common.getString(NAME, ""));
// Activity용의 Preferences에서 값 검색
     SharedPreferences activity = getPreferences(MODE_PRIVATE);
     CheckBox s = (CheckBox)findViewById(R.id.silent_mode);
     s.setChecked(activity.getBoolean(SILENT_MODE, false));
```

11_1HelloPreferences(Preference Test)

```
//  저장 하기   (onPause()에 입력)
// 외부에서 액세스 할 수 있는 Preference에 값을 설정
SharedPreferences common = getSharedPreferences(
COMMON_PREF, MODE_WORLD_READABLE |
    MODE_WORLD_WRITEABLE);
EditText name = (EditText)findViewById(R.id.name);
Editor editor = common.edit();
editor.putString(NAME, name.getText().toString());    editor.commit();
// Activity용의 Preferences에 값을 설정
SharedPreferences activity = getPreferences(MODE_PRIVATE);
CheckBox s = (CheckBox)findViewById(R.id.silent_mode);
editor = activity.edit();
editor.putBoolean(SILENT_MODE, s.isChecked());    editor.commit();
```

파일의 공유

- 외부 읽기가 가능하도록 하려면, 파일 생성시 MODE_WORLD_READABLE 플래그를 사용해야 함
- 외부 쓰기는 MODE_WORLD_WRITABLE 플래그를 사용해야 함
- 외부의 파일을 읽기 위해서는 해당 파일을 생성한 프로그램의 컨텐스트를 구해야 함 (파일 열기에서 경로를 지정할 수 없기 때문임)
- 컨텍스트를 구하는 메소드

Context createPackageContext (String packageName, int flags)

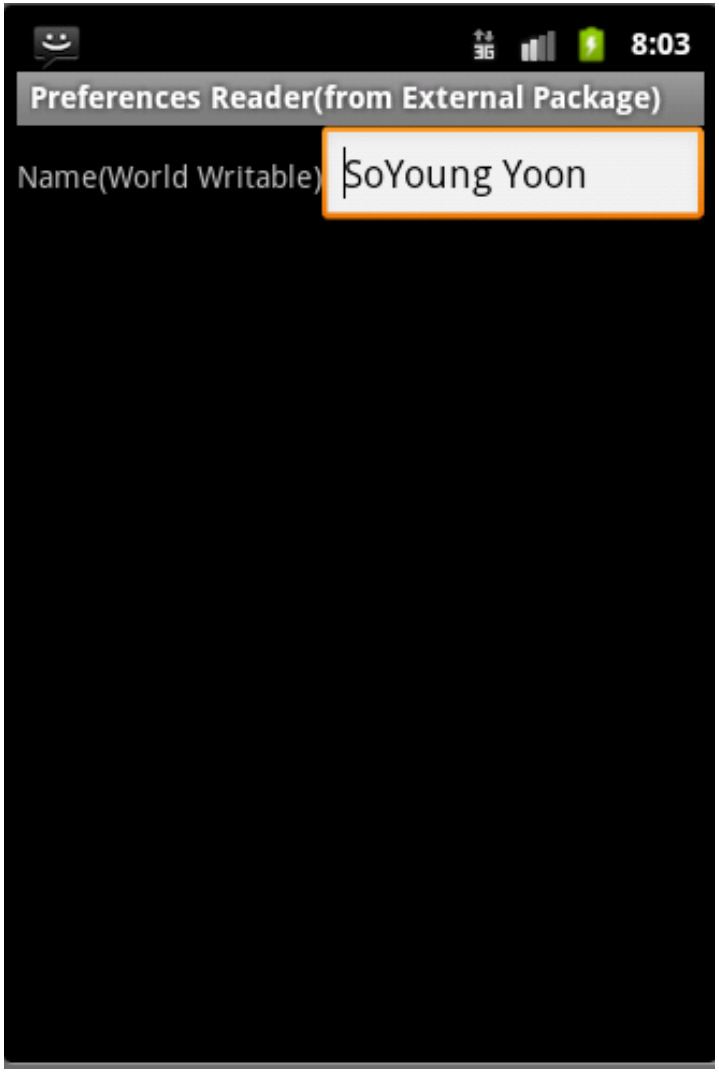
packageName : 패키지 경로 지정

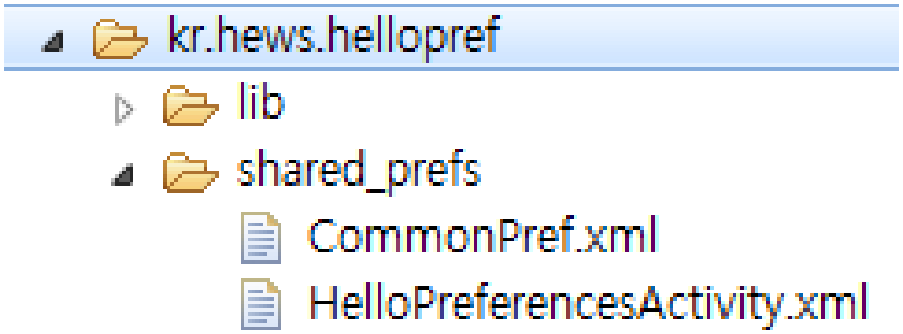
flags : 모드 지정 (CONTEXT_IGNORE_SECURITY 플래그로 지정함)

CONTEXT_IGNORE_SECURITY = 0의 값

- 사용하기 전에 해당 패키지가 설치 되어 있어야 함

다른 패키지에 있는 Preference 파일 읽어 표시





- 위에 있는 패키지의 파일을 읽어 이름 표시
- kr.hews.hellopref
- CommonPref.xml
- Name

11_2HelloPrefReader (Pref 공유)

```
//  읽어 오기  (onResume()에 입력)
public static final String PACKAGE_NAME = "kr.hews.hellopref";
public static final String COMMON_PREF = "CommonPref";
public static final String NAME = "Name";
try {
   Context other = createPackageContext(PACKAGE_NAME,0);
   // 환경 설정에서 값을 검색
   SharedPreferences common = other.getSharedPreferences(
       COMMON_PREF, MODE_WORLD_READABLE |
       MODE_WORLD_WRITEABLE);
   EditText name = (EditText)findViewById(R.id.name);
   name.setText(common.getString(NAME, ""));
} catch (NameNotFoundException e) { }
```

11_2HelloPrefReader (Pref 공유)

```
//  저장 하기   (onPause()에 입력)
try {
    Context other = createPackageContext(PACKAGE_NAME,0);
    SharedPreferences common =
    other.getSharedPreferences(
        COMMON_PREF, MODE_WORLD_READABLE |
          MODE_WORLD_WRITEABLE);
    EditText name = (EditText)findViewById(R.id.name);
    Editor editor = common.edit();
    editor.putString(NAME, name.getText().toString());
    editor.commit();
} catch (NameNotFoundException e) {        }
```

11_3 SimplePrefsTest

public abstract class

PreferenceActivity

extends ListActivity

java.lang.Object
  ↳ android.content.Context
    ↳ android.content.ContextWrapper
      ↳ android.view.ContextThemeWrapper
        ↳ android.app.Activity
          ↳ android.app.ListActivity
            ↳ android.preference.PreferenceActivity

PrefernceActivity

- 자동화된 옵션 입력 및 관리방법

1. Activity가 아닌 PreferenceActivity로부터 상속을 받음
2. 입력 받고자 하는 옵션의 종류를 XML 문서에 기록에 놓음
   XML파일에서 레이아웃 루트는 PreferenceScreen 이어야 함
3. onCreate()에서
   addPreferencesFromResource()로 레이아웃 지정

위의 같이 처리하면 Preference 관리가 자동으로 됨 (파일 저장)

11_3SimplePrefsTest // /res/xml/prefactivity.xml

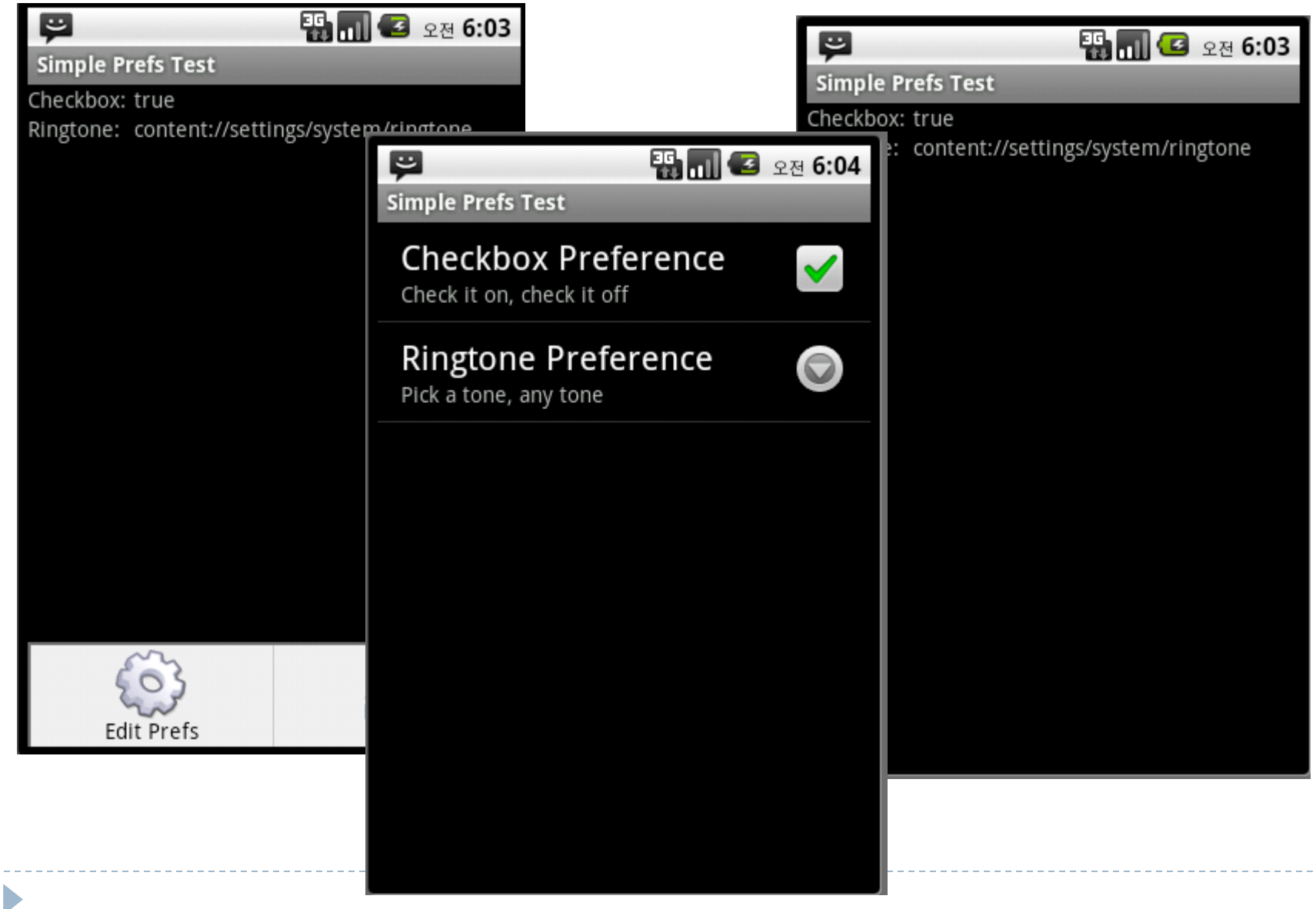
```
<PreferenceScreen
        xmlns:android="http://schemas.android.com/apk/res/android">
    <CheckBoxPreference
    android:key="@string/checkbox"
    android:title="Checkbox Preference"
    android:summary="Check it on, check it off" />
    <RingtonePreference
    android:key="@string/ringtone"
    android:title="Ringtone Preference"
    android:showDefault="true"
    android:showSilent="true"
    android:summary="Pick a tone, any tone" />
</PreferenceScreen>
```

11_3SimplePrefsTest

```
public class EditPreferences extends PreferenceActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    addPreferencesFromResource(R.xml.preferences);
    }
}
```

11_3SimplePrefsTes



Local File

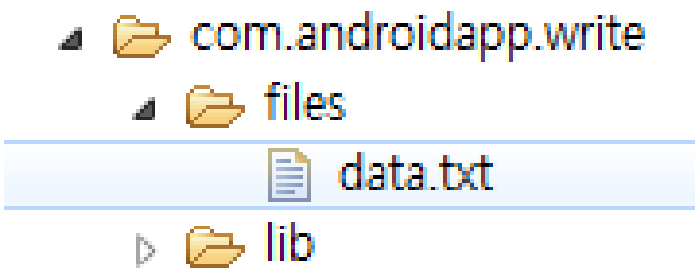
Local File

1. FileWrite
   - ✓ Context 클래스의 openFileOutput 메소드를 사용하여 반환 값으로 FileOutputStream을 얻을 수 있음
   - ✓ 디렉토리 명을 포함하지 않는 파일명 전달
   - ✓ 디렉토리는 시스템이 어플리케이션별로 작성
   - ✓ /data/data/<패키지명>/files 디렉토리가 지정됨
   - ✓ 다른 애플리케이션의 로컬 파일을 직접 액세스 불가능
2. FileRead
   - ✓ 개발 시 작성한 파일을 패키지에 포함시키기
   - ✓ /res/raw에 파일을 둠
   - ✓ 그 파일은 Resource 클래스의 openRawResource 메소드에 리소스 ID를 지정하여 호출한 InputStream을 사용
   - ✓ 검색 전용, 재기록 불가능

Local File - FileWriteTest

```
TextView t = (TextView)findViewById(R.id.writetext);
BufferedWriter br = null;
try {
        br = new BufferedWriter(
                new OutputStreamWriter(openFileOutput("data.txt",
                                                      MODE_WORLD_WRITEABLE)));
        br.append("Android");    br.append("Application");
        t.setText("파일이 정상적으로 생성");
} catch (IOException e) {
        Log.e("IO", "File Output Error");
        t.setText("파일 생성 Error");
} finally {
        try {
                if (br != null) br.close();
        } catch (IOException e) {    }
}
```

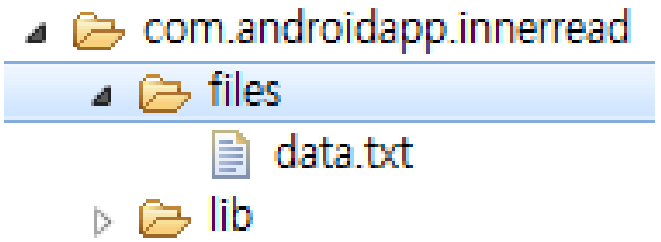
FileOutputStream



Local File – FileInnerReadTest

```
TextView t = (TextView)findViewById(R.id.writetext);
BufferedWriter br = null;
 try {
        br = new BufferedReader(
                    new InputStreamReader(openFileInput("data.txt")));
        String msg = br.readLine();
        while (msg != null) {
        t.append(msg + "₩n");
        msg = br.readLine();
        }
} catch (IOException e) {
        Log.e("IO", "File Output Error");
        t.setText("파일 생성 Error");
} finally {
        try {
              if (br != null) br.close();
        } catch (IOException e) {    }
}
```

InputStream

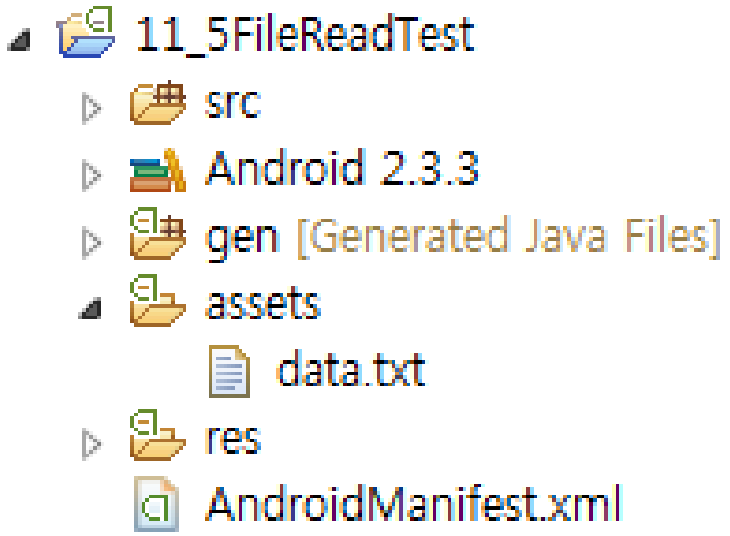


등록 되 있어야 함

Local File - FileReadTest

```
try {
      br = new BufferedReader(
                  new InputStreamReader(
                        getAssets().open("data.txt"),"euc-kr" ));
      String msg = br.readLine();
      while (msg != null) {
         t.append(msg + "₩n");
         msg = br.readLine();
      }
} catch (IOException e) {
         Log.e("IO", "File Input Error");
} finally {
     try {
         br.close();
     } catch (IOException e) {    }
}
```

InputStream



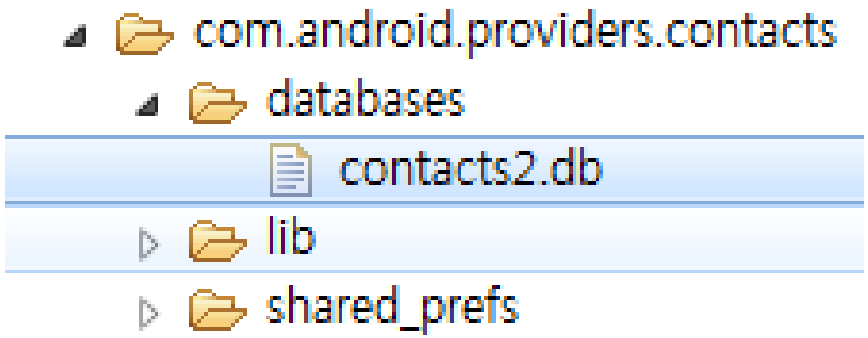
SQLite

SQLite의 이해

- 데이터의 액세스나 테이블의 정의에는 일반적인 질의어를 사용할 수 있음 (SQL 사용)
- Client/Server 형이 아닌 라이브러리 형 (User/Password 없음)
- 애플리케이션 별로 디렉토리가 지정됨
  /data/data/<패키지명>/database
- 다른 애플리케이션에서는 직접 엑세스 할 수 없음
- 다른 애플리케이션에서 데이터 공유를 원할 경우 Content Provider의 인터페이스를 이용해야 함
- 웹을 통한 외부의 DB에는 접속할 수 없음
- 테이블에는 관습적으로 autoincrement를 지정한 _id 라는 이름의 column을 작성함
- UTF-8 포맷으로 데이터를 저장함

SQLite의 shell

shell → sqlite3 : SQLite 관리용 tool (GUI 툴 없음)




```
C:₩Users₩julie yoon>adb -s emulator-5554 shell
# sqlite3 /data/data/com.android.providers.contacts/database/contacts2.db
sqlite3 /data/data/com.android.providers.contacts/database/contacts2.db
SQLite version 3.6.22
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> _
```

SQLite 의 주요 클래스

- SQLiteOpenHelper 클래스
  - ✓ 추상 클래스(메서드 재정의하여 인스턴스 작성)
  - ✓ Database open/close, 생성, 업그레이드
  - ✓ getReadableDatabase, getWriteableDatabase의 결과로 SQLiteDatabase Instance가 반환됨
- SQLiteDatabase class
  - ✓ Table에 대한 조회/삽입/삭제./수정
  - ✓ 조회 : query, 삽입 : insert, 삭제 : delete, 수정 : update 메소드가 있음

SQLiteOpenHelper

public abstract class

SQLiteOpenHelper

extends Object

Public Methods	
synchronized void	close () Close any open database object.
synchronized SQLiteDatabase	getReadableDatabase () Create and/or open a database.
synchronized SQLiteDatabase	getWritableDatabase () Create and/or open a database that will be used for reading and writing.
abstract void	onCreate (SQLiteDatabase db) Called when the database is created for the first time.
void	onOpen (SQLiteDatabase db) Called when the database has been opened.
abstract void	onUpgrade (SQLiteDatabase db, int oldVersion, int newVersion) Called when the database needs to be upgraded.

SQLiteOpenHelper 클래스의 생성자

SQLiteOpenHelper(Context context, String name,
SQLiteDatabase.CursorFactory factory, int version)

- ✓ context : DB를 생성하는 컨텍스트 (보통 메인 액티비티)
- ✓ name: DB의 파일 이름
- ✓ version : DB의 파일 버전
- ✓ factory : 커스텀 커서를 사용하고자 할 때 지정하는 것으로 표준을 사용할 경우 null로 지정

SQLiteOpenHelper의 주요 메서드

- **onCreate** : DB가 처음 만들어질 때 호출됨, 테이블을 만들고 초기 레코드를 삽입함
- **onUpgrade** : DB를 업그레이드(버전이 변경될 때)할 때 호출됨, 기존 테이블을 삭제하고 새로 만들거나 ALTER TABLE로 스키마를 수정함
- **onOpen** : DB를 열 때 호출됨
- **getReadableDatabase** : 읽기 위해 DB를 열기 함, DB가 존재하지 않으면 onCreate가 호출되며 버전이 바뀌었으면 onUpgrade가 호출됨
- **getWriteableDatabase** :읽고 쓰기 위해 DB를 열기 함, 권한이 없거나 디스크가 가득 차면 실패함
- **close** : DB를 닫기 함

SQLiteOpenHelper를 사용하지 않고 직접 DB 생성

- SQLiteDatabase **Context.openOrCreateDatabase** (String name, int mode, SQLiteDataBase.CursorFactory factory)를 사용하여 직접 DB 파일을 생성
- 리턴되는 db의 execSQL 메서드를 통해 CREATE TABLE 실행
- 위의 경우 직접 버전이 바뀐 경우 등을 판단해야 하므로 도우미(SQLiteOpenHelper)를 사용하는 것이 편리함

SQLiteDatabase

public class
SQLiteDatabase

extends SQLiteClosable

java.lang.Object
  ↳ android.database.sqlite.SQLiteClosable
    ↳ android.database.sqlite.SQLiteDatabase

long	insert (String table, String nullColumnHack, ContentValues values) Convenience method for inserting a row into the database.
int	update (String table, ContentValues values, String whereClause, String[] whereArgs) Convenience method for updating rows in the database.
Cursor	query (String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy, String limit) Query the given table, returning a `Cursor` over the result set.
Cursor	query (String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy) Query the given table, returning a `Cursor` over the result set.
Cursor	query (boolean distinct, String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy, String limit) Query the given URL, returning a `Cursor` over the result set.
void	execSQL (String sql) Execute a single SQL statement that is not a query.
void	execSQL (String sql, Object[] bindArgs) Execute a single SQL statement that is not a query.

ContentValues

- ContentValues : 테이블의 레코드를 표현하는 클래스
- 레코드에 포함된 필드의 이름과 값의 Map
- 레코드는 테이블마다 형태가 다르기 때문에, 빈 객체를 만든 후 put 메서드를 호출하여 필드와 값의 쌍을 여러 개 저장함
  - ✓ void put (String key, Integer value)
  - ✓ void put (String key, String value)
  - ✓ void put (String key, Boolean value)

SQLiteDatabase

- 레코드 삽입
  - ✓ ContentValues를 준비한 뒤에 insert 메서드 호출 Long **SQLiteDatabase.insert(String table, String nullColumnHack, ContentValues values)**
  - ✓ SQL 명령 직접 실행 (SELECT 명령을 제외한 명령 실행) void execSQL(String sql)
- 레코드 삭제/ 갱신
  - ✓ **int delete(String table, String whereClause, String[] whereArgs)**
  - ✓ **int update(String table, ContentValues values, String whereClause, String[] whereArgs)**

SQLiteDatabase

- 레코드 검색
  - ✓ **Cursor query(boolean distinct, String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy, String limit)**
  - ✓ 위의 방법은 각각 WHERE, ORDER BY, GROUP BY, HAVING, TOP, DISTINCT 등을 지정하여 복잡함
  - ✓ 사용하지 않는 부분은 null로 지정
  - ✓ rawQuery 메서드를 이용해 SELECT문을 바로 실행
  - ✓ 검색의 결과가 직접 전달 되지 않으며, 위치를 가리키는 커서(Cursor)가 전달됨

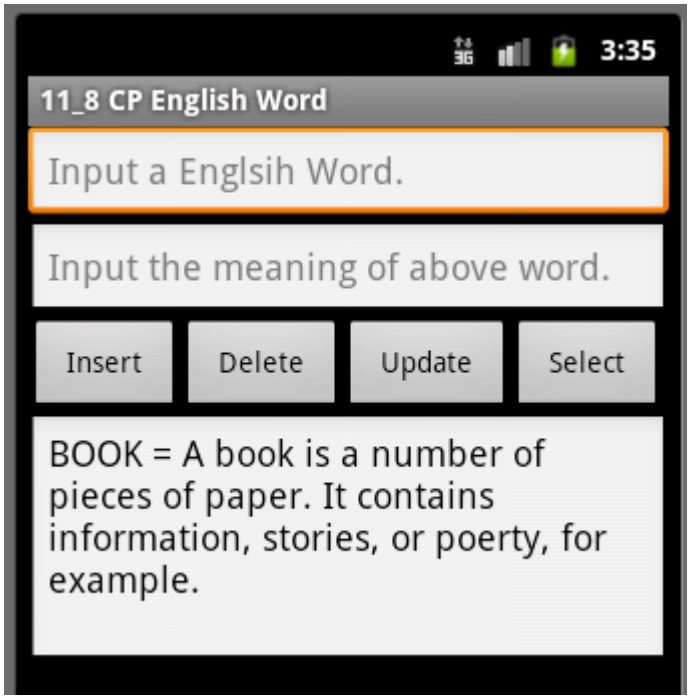
Cursor 클래스의 주요 메서드

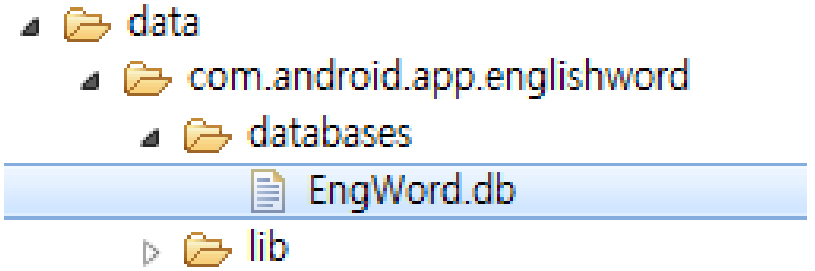
- close : 결과셋 닫기
- getColumnCount : 컬럼의 개수 구하기
- getColumnIndex : 이름으로부터 컬럼 번호 구하기
- getColumnName : 번호로부터 컬럼 이름 구하기
- getCount : 결과셋의 레코드 개수
- getInt : 컬럼값을 정수로 구함, 인수로 컬럼 번호 전달
- getDouble : 컬럼 값을 실수로 구함, 인수로 컬럼 번호 전달
- getString : 컬럼 값을 문자열로 구함, 인수로 컬럼 번호 전달
- moveToFirst(첫 레코드), moveToLast(마지막) : 결과셋이 비어있으며 false 리턴
- moveToNext(다음), movetoPrevious(이전) : 마지막/첫 레코드일 때 false 리턴
- moveToPosition : 임의의 위치로 이동

Cursor 클래스

- SQLite는 타입 점검을 느슨하게 수행하므로 반드시 컬럼의 타입과 같은 타입으로 읽어야 하는 것은 아님
- 정수 컬럼을 getString으로 읽을 수 있음. 만일 정수 컬럼을 읽어 문자열 형태로 사용하고 싶다면, getString으로 읽어 사용해도 됨 (캐스팅 처리가 됨)
- getInt, getDouble, getString 등의 Index는 번호로 주거나 getColumnIndex 메서드를 사용해 조사하여 넘겨줌 : 매번 조사하지 말고 미리 조사하여 두고 사용할 것
- 커서 및 DB는 사용 후 반드시 닫아야 함

11_8CPEnglishWord





- database : EngWord.db, table : dic
- field : _id, word, meaning
- Insert : 중복체크 없이 삽입
- Delete : 입력된 단어에 해당하는 레코드 삭제
- Update : 입력된 단어에 해당하는 의미 갱신
- Select : 단어가 입력 안된 경우 모든 레코드 검색, 단어가 입력된 경우 해당 단어만 검색

11_8CPEnglishWord

EnglishWord

EngWordDbAdapter mAdapter
onCreate()
Button.OnClickListener
mClickListener {
}

EngWordDbAdapter
open() close() insertWord() deleteWord() updateWord() selectWord() class WordDBHelper extends SQLiteOpenHeper {…}

Content Provider

URI (Uniform Resource Identifier)

- URI : 정보의 고유한 명칭
- URL보다 상위 개념으로 누가 어떤 정보를 제공하는지, 어떤 정보를 원하는지에 대한 정보가 URI에 작성됨
- RFC 2396에 URI를 작성하는 방식이 명시되어 있음

  content://authority/path/id

  - ✓ content:// 해당 문자열이 URI임을 나타내는 접두 무조건 붙여야 함
  - ✓ authority 정보 제공자의 명칭이되 중복되면 안 되므로 패키지명 사용을 권장함
  - ✓ path 정보의 종류를 지정하는 가상의 경로 (한 제공자가 여러 개의 정보를 제공할 수 있으므로 경로로 구분함)
  - ✓ id 구체적으로 원하는 정보 (전체 정보를 다 읽을 때는 생략 가능)

URI (Uniform Resource Identifier)

- CP는 단수와 복수에 대해서 두 가지 형태의 URI를 각각 정의해야 함
  - ✓ 복수 URI – path까지 포함되어 있는 것
  - ✓ 단수 URI - 복수 URI 뒤에 id가 더 추가되는 형식
- 예)
  - ✓ content://com.stockmarket/stock
  - ✓ content://com.stockmarket/stock/posko
- URI 객체 생성

  URI 문자열을 만든 뒤, Uri.parse()를 이용해 URI 객체 생성

  static Uri parse (String uriString)

  (parse 메서드는 오류 발생을 하지 않으므로 주의 할 것)

URI (Uniform Resource Identifier)

- URI의 path 정보를 문자열 목록으로 조사함
  List<String> getPathSegments()
  - ✓ 위 목록의 0번 요소 : path
  - ✓ 위 목록의 1번 요소 : id

  "문자열이므로 속도가 느리며, 개수가 많아지면 분석이 까다로움"

- 문자열 안의 요구 정보를 분석하여 정수 코드로 변환하는 UriMatcher라는 유틸리티 클래스를 사용함
  (URI를 분석해 놓고 요구 코드를 정의함)
  - ✓ void addURI(String authority, String path, int code)
  - ✓ int match(Uri uri)

URI (Uniform Resource Identifier)

- addURI : authrity, path의 쌍으로 정수 코드와 대응시켜 맵을 등록 함, path에서 *은 임의의 문자열과 대응되며 #은 숫자 하나와 대응됨
- match : uri를 분석하여 등록된 정수 코드를 반환함.
  만약 uri에 해당하는 코드가 발견되지 않으면 -1을 리턴함
- CP는 클라이언트의 URI를 일일이 분석할 필요 없이 UriMacher가 분석해 놓은 정수 코드로 요청을 파악하여 정보를 리턴하면 됨

ContentProvider

public abstract class
ContentProvider

extends Object
implements ComponentCallbacks

java.lang.Object
  ↳ android.content.ContentProvider

▶ Known Direct Subclasses
MockContentProvider, SearchRecentSuggestionsProvider

The primary methods that need to be implemented are:

- `onCreate()` which is called to initialize the provider
- `query(Uri, String[], String, String[], String)` which returns data to the caller
- `insert(Uri, ContentValues)` which inserts new data into the content provider
- `update(Uri, ContentValues, String, String[])` which updates existing data in the content provider
- `delete(Uri, String, String[])` which deletes data from the content provider
- `getType(Uri)` which returns the MIME type of data in the content provider

자료 공유

- 자료를 공유하기 위한 CP 작성
  1. ContentProvider 클래스를 상속받아 클래스 작성
  2. 정보를 관리 및 제공하는 메서드들을 재정의
  - ✓ CP가 로드 될 때 onCreate 메서드가 호출되며, 이곳에 제공할 데이터를 준비함
  - ✓ 정보가 DB에 들어 있을 필요는 없으며 내부 배열이나 네트워크의 정보도 가능하지만 대개의 경우 DB에 들어 있음 (DB에 들어있는 정보의 경우 onCreate에서 DB open)
  - ✓ CP가 query, update, delete, insert 등의 메서드를 모두 재정의 해야 하는 것은 아님. 읽기 전용의 경우는 query 메서드만 제공하고 나머지는 구현하지 않아도 됨, 삽입 전용 insert만 구현
  3. CP를 매니페스트에 등록 (CP의 이름과 제공자명을 밝힘)

자료 공유

- 데이터의 MIME 타입 조사

  String getType(Uri uri)

  ✓ 정보의 개수에 따라 MIME 타입의 형식이 다름

  ✓ 단수 : vnd.회사명.cursor.item/타입

  ✓ 복수 : vnd.회사명.cursor.dir/타입

- 실제 데이터를 제공 및 관리하는 query, insert, delete 등의 메서드들을 제공하는 정보에 따라 재정의해야 함

URI (Uniform Resource Identifier)

- URI의 path 정보를 문자열 목록으로 조사함
  List<String> getPathSegments()
  - ✓ 위 목록의 0번 요소 : path get(0)
  - ✓ 위 목록의 1번 요소 : id get(1)
- 예) Uri contentUri =
  Uri.parse("content://com.stockmarket/stock/posco");

```
List<String> uriSegment=contentUri.getPathSegments();
uriSegment.get(0);  // stock
uriSegment.get(1);  // posco
// Uri에 따라 id는 없을 수도 있으므로 어떤 종류 정보(id 제공 or Not)를 제공하는지 밝히고 사용자는 URI를 통해 필요 정보 요청 (id를 가지고 있는지 아닌지를 구분해서 사용하는 것이 필요함)
```

Macher의 작성

```
// UriMacher 정의
static final Uri CONTENT_URI =
    Uri.parse("content://com.android.app.englishword.EnglishW
    ord/word");
static final int ALLWORD = 1;
static final int ONEWORD = 2;
static final UriMatcher Matcher;
static {
Matcher = new UriMatcher(UriMatcher.NO_MATCH);
Matcher.addURI("com.android.app.englishword.EnglishWord",
    "word", ALLWORD);
Matcher.addURI("com.android.app.englishword.EnglishWord",
    "word/*", ONEWORD);
}
```

Matcher의 사용 -1 (getType)

```
// getType 작성시 Matcher 사용
public String getType(Uri uri) {
    if (Matcher.match(uri) == ALLWORD) {
        return "vnd.EnglishWord.Data.cursor.item/words";
    }
    if (Matcher.match(uri) == ONEWORD) {
        return "vnd.EnglishWord.Data.cursor.dir/word";
    }
    return null;
}
```

Matcher의 사용 -2 (query)

```
// 전체에 대한 쿼리 명령
sql = "SELECT eng, han FROM dic";
// 단어 선택 where절 추가
if (Matcher.match(uri) == ONEWORD) {
    sql += " where eng = '" + uri.getPathSegments().get(1) + "'";
}
```

1) 전체를 요구하면 dic 테이블 전체를 덤프해서 리턴
2) 한 단어만 요구한 것이면 where절을 작성하여 요구한 단어만 조사

예) 모든 레코드 리턴 Uri
content://com.android.app.englishword.EnglishWord/word

예) boy 레코드 리턴 Uri
content://com.android.app.englishword.EnglishWord/word/boy

Matcher의 사용 (insert, delete, update)

- insert : 단수, 복수의 구분이 필요 없음, 삽입 성공 후 notyfyChange 메서드 호출하여 변경 사실 통보
- delete, update : 단수, 복수에 대한 구분 필요 - 단수의 경우 where절에 조건 추가가 필요함 - 조건은 URI를 통해 전달되거나 selection인수로 전달 될 수 있으므로 조건을 AND로 연결함 (URI로는 하나 또는 전체라는 것만 지정할 수 있기 때문에 selection인수 사용)

Matcher의 사용 -3 (delete / update)

```
public int delete(Uri uri, String selection, String[] selectionArgs) {
    int count = 0;
    switch (Matcher.match(uri)) {
        case ALLWORD:
            count = mDB.delete("dic", selection, selectionArgs); break;
        case ONEWORD:
            String where;
            where = "eng = '" + uri.getPathSegments().get(1) + "'";
            if (TextUtils.isEmpty(selection) == false) {
                where += " AND" + selection;
            }
            count = mDB.delete("dic", where, selectionArgs); break;
    }
    // .....  나머지 코드
}
```

매니페스트 등록

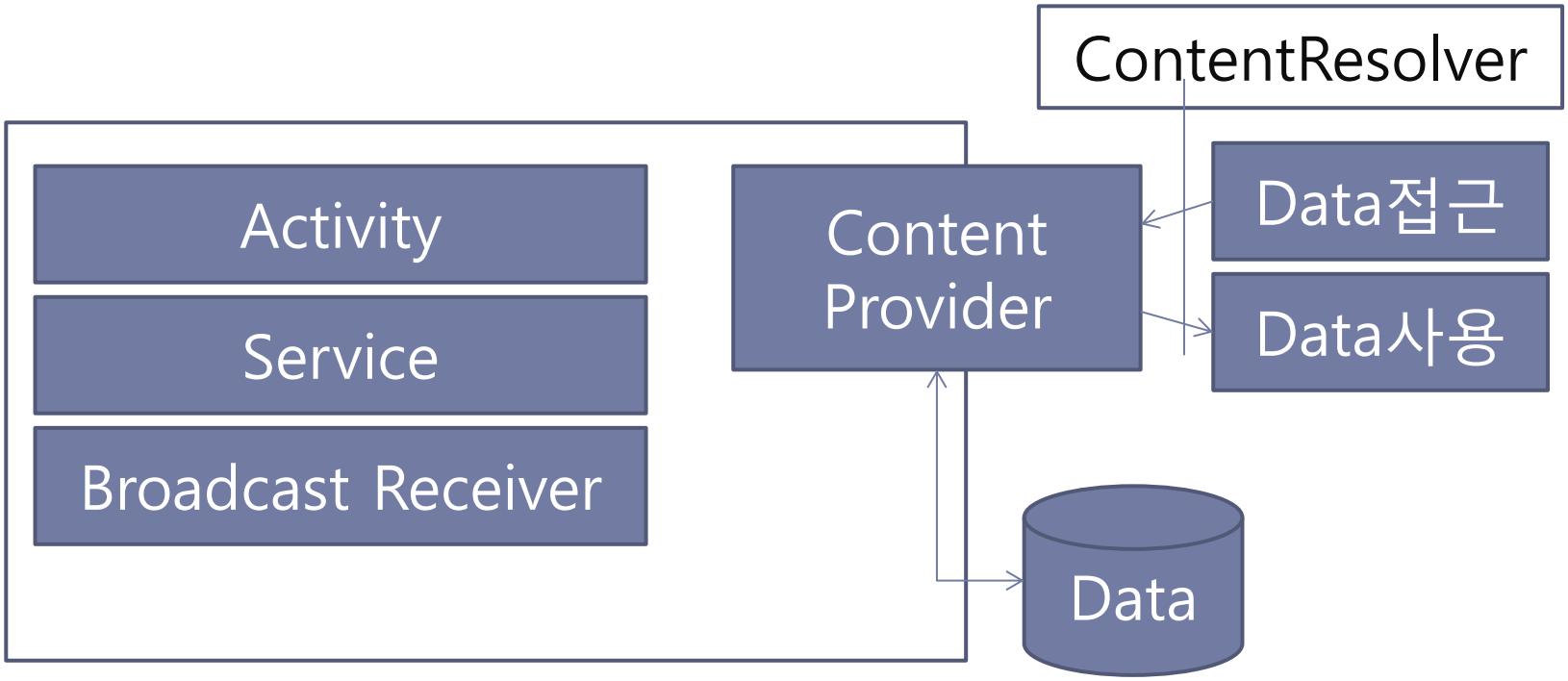
```
<provider android:name="EWProvider"
    android:authorities="com.android.app.englishword.English
    Word" />
```

시스템은 정보를 요청하는 쪽의 URI와 매니패스트의 URI를 비교해 보고 제공자가 일치하는 CP를 호출함. 매니패스트에 자신이 제공하는 데이터가 무엇인지를 시스템 전역에 알리는 것임

Content Provider

Data 공유 기능

다른 Application 에서 Data를 접근할 수 있는 Interface 제공



ContentResolver

public abstract class
ContentResolver

extends Object

java.lang.Object
↳ android.content.ContentResolver

▶ Known Direct Subclasses
MockContentResolver

final String	getType (Uri url) Return the MIME type of the given content URL.
final Uri	insert (Uri url, ContentValues values) Inserts a row into a table at the given URL.

final Cursor	query (Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder) Query the given URI, returning a `Cursor` over the result set.

ContentResolver 클래스

- 콘텐트를 읽기 위해 ContentResolver 객체를 구해야 함
- 리졸브는 쿼리를 통해 CP와 간접적으로 통신하는 표준 인터페이스
- 응용 프로그램마다 하나씩 ContentResolver를 가지고 있으므로 메서드를 이용해 구하여 사용함
- ContentResolver ContextWrapper.getContentResolver()
- 어떤 CP와 통신할 것인가는 각 메서드로 전달되는 URI에 의해 결정
- 시스템은 URI로부터 콘텐트 제공자를 찾아 CP에 로드하고 CP의 메서드를 호출함 (authority를 비교함)
- 응용 프로그램이 직접 CP를 생성할 수 없음 // 응용 프로그램은 원하는 데이터의 URI만 전달함
- 리졸브의 query, insert, delete, update 등의 메서드를 통해 CP의 메서드를 호출하고 CP의 데이터를 자신의 것처럼 관리
- 리졸브는 URI로부터 CP를 찾고, CP의 메서드를 호출하여 결과셋을 중계하는 역할을 함

CallWordCP

```
static final String WORDURI =
      "content://com.android.app.englishword.EnglishWord/word";
ContentResolver cr = getContentResolver();
Cursor cursor = cr.query(Uri.parse(WORDURI), null, null, null, null);
```

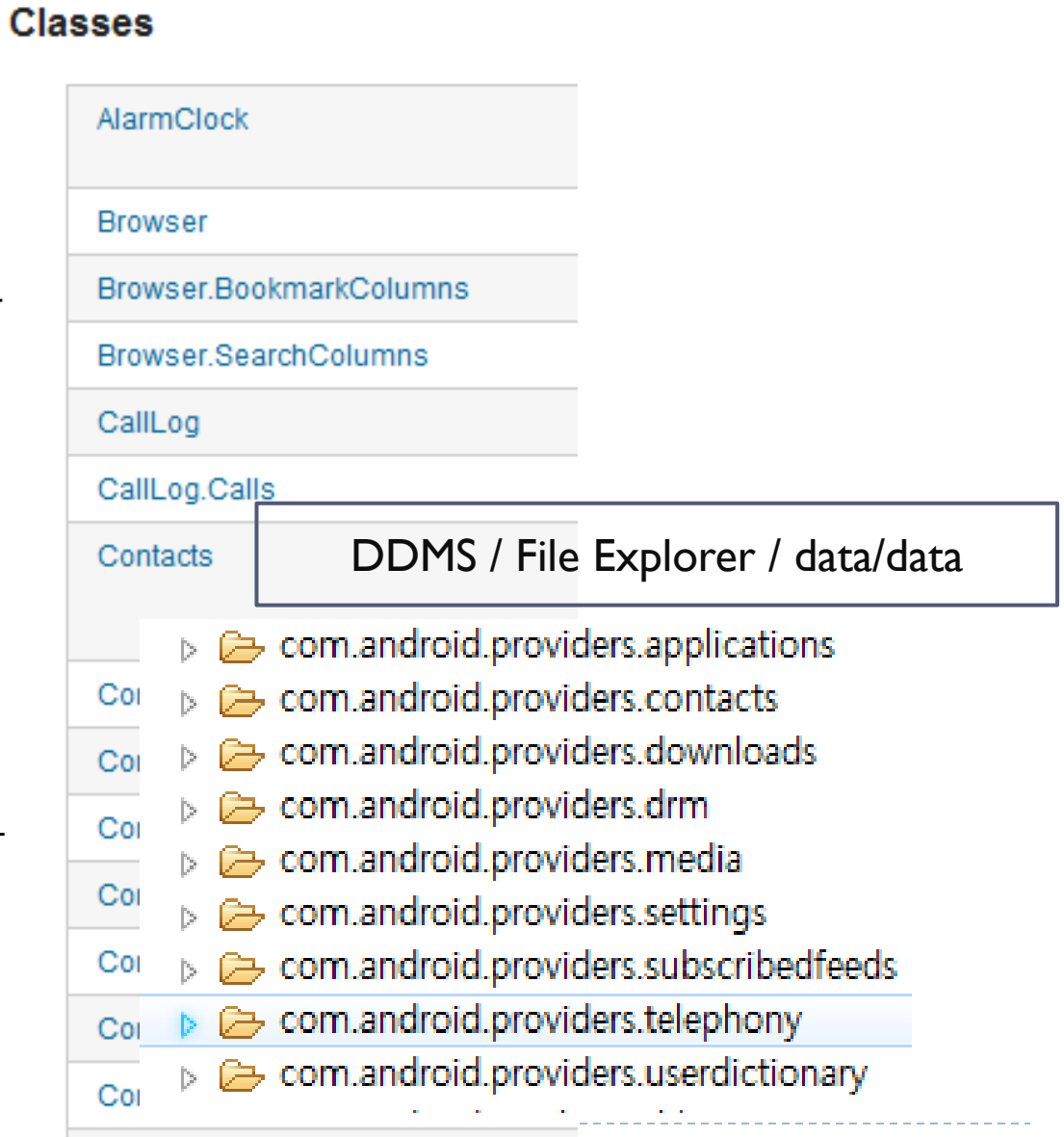
매니페스트 등록

외부의 저장소에 저장하기 위해 필요한 권한

```
<uses-permission
    android:name="android.permission.WRITE_EXTERNAL_STORAGE"
    />
```

providers

Browser – 인터넷 즐겨찾기 등
Contacts - 연락처
CallLog – 전화 통화
Media – mp3
UserDictionary – 사용자 사전
위의 내용은
공유가 open되어 있는 상태로
ContactResolver를 이용해
데이터 정보를 사용할 수 있음



Permission

CallLog에 접근하기 위해 필요한 permission

<uses-permission
android:name= *"android.permission.READ_CONTACTS" />*