3장 : 액티티비 연동과 데이터 관리

옥 상 훈
Blog: http://okgosu.tistory.com
Web Site: http://okgosu.net
twitter @okgosu

3장:액티티비 연동과 데이터 관리

:액티비티, 인텐트, 데이터 관리 방법

by OkGosu.Net

3.1. 인텐트

- 인텐트 개요
- 인텐트 작동원리
- 인텐트 호출

인텐트 개요

- 인텐트
  - 원하는 작업을 안드로이드 시스템에 지시하는 방법
  - 인텐트는 다음과 같은 대상에게 전달되어 다양한 작업을 수행
    - 화면에 보이는 액티비티
    - 백그라운드 서비스
    - 폰시스템의 상태를 알려주는 브로드캐스트 리시버

- 인텐트 구성
  - 액션 : 작업 종류 지시
    - 예) ACTION_VIEW = 해당 자원을 화면에 표시
      ACTION_EDIT = 해당 자원을 변경
  - 데이터 (URI) : 자원의 위치
    - 예) content://contacts/people/1 또는 특정 액티비티명
  - 카테고리 : 인텐트 제어

액션

- 의미
  - 수행되어야 하는 액션을 지칭하는 문자열
  - 브로드캐스트 인텐트의 경우에는, 얻게 되고 리포트 될 액션
- 액션의 종류는 상수로 정의

상수	타겟 컴포넌트	액션
ACTION_CALL	액티비티	전화를 걸어라.
ACTION_EDIT	액티비티	사용자에게 편집할 데이터를 표시하라.
ACTION_MAIN	액티비티	데이터 입력과 반환 결과 없이 태스크의 최초의 액티비티로써 액티비티를 실행하라.
ACTION_SYNC	액티비티	모바일 디바이스의 데이터와 서버의 데이터를 동기화하라.
ACTION_BATTERY_LOW	브로드캐스트 리시버	배터리가 부족하다는 경고.
ACTION_HEADSET_PLUG	브로드캐스트 리시버	헤드셋이 디바이스에 연결 또는 분리되었다는 것.
ACTION_SCREEN_ON	브로드캐스트 리시버	스크린이 켜졌다는 것.
ACTION_TIMEZONE_CHANGED	브로드캐스트 리시버	타임존 설정이 바뀌었다는 것.

카테고리

- 액션을 추가 분류하기 위한 정보
  - 인텐트를 제어해야 하는 컴포넌트의 종류에 대한 추가적인 정보를 포함하는 문자열
- 인텐트 클래스에 카테고리는 상수로 정의

상 수	의미하는 바
CATEGORY_BROWSABLE	타겟 액티비티는 링크에 의해 참조되는 데이터(예를 들어 이미지나 이메일 메시지)를 보여주기 위해 브라우저에 의해 안전하게 호출될 수 있다.
CATEGORY_GADGET	액티비티는 가젯들을 보유할 수 있는 다른 액티비티 내에 임베딩될수 있다.
CATEGORY_HOME	액티비티는 홈 화면을 보여준다. 디바이스가 켜질 때나 HOME 키가 눌렸을 때, 사용자가 보게되는 첫 번째 화면.
CATEGORY_LAUNCHER	액티비티는 하나의 태스크에서 최초의 액티비티가 될 수 있으며, 최상위 계층의 애플리케이션 런처에 리스트된다.
CATEGORY_PREFERENCE	타겟 액티비티는 설정 패널이다.

인텐트 작동원리

- 인텐트 라우팅
  - 명시적 라우팅
    - 인텐트를 처리할 대상을 명시적으로 지정
  - 묵시적 라우팅
    - 처리할 대상을 명시적으로 지정하지 않음
    - 다수의 컴포넌트에 전달될 수도 있음

인텐트 호출

- 액티비티 실행
  - 인텐트 작성
    - 예) new Intent(this, HelpActivity.class)
  - 인텐트 호출
    - startActivity()
    - startActivityForResult()
    - sendBroadcast()
    - sendOrderedBroadcast()

인텐트의 활용

- 인텐트 활용 예
  - 하나의 화면(액티비티)에서 다른 화면(액티비티)로 전환할 때
  - 전화를 걸 때
  - SMS, 이메일을 전송할 때
  - 지도를 보거나 검색할 때
  - 사진, 음악, 비디오를 재생할 때
  - 스케쥴을 설정할 때
  - 폰 부팅 또는 배터리가 방전시 필요한 작업을 할 때

인텐트 사용 방법

- 인텐트에 대한 노출 설정 (인텐트 필터)
  - 안드로이드 매니페스트 XML 파일에 액티비티나 서비스에 인텐트를 수신하는 조건

  - 예) AndroidExam5_1 액티비티의 매티페스트 XML 정의
    - <activity android:name=". AndroidExam5_1" android:label="예제 5-1">
    - <intent-filter>
    - <action android:name="exam5-1" />
    - <category android:name="android.intent.category.DEFAULT" />
    - </intent-filter>
    - </activity>

인텐트 사용 방법

- 인텐트 호출 방법
  - Intent 객체와 매개변수의 조합을 사용하여 인텐트 객체를 생성한 다음 인텐트를 전달하는 startActivity() 함수를 호출

- 종류
  - 명시적 인텐트: 액티비티 클래스명으로 호출
    - Intent intent = new Intent(getApplicationContext(), AndroidExam5_1.class);
    - startActivity(intent);
  - 암시적 인텐트: 인텐트 필터의 조건으로 호출 (예: 액션명)
    - Intent intent = new Intent("exam5-1");
    - startActivity(intent);

인텐트 내장 액션의 활용

- 액션과 함께 지정하는 요소
  - 처리할 리소스가 있는 URI
  - 리소스의 MIME 타입 지정
  - 부가 데이터

인텐트 내장 액션의 활용

INTENT 액션 상수		INPUT/OUTPUT
액티비티 작동	ACTION_MAIN	없음/없음
	ACTION_CHOOSER	없음/EXTRA_INTENT의 프로토콜에 따름
	ACTION_PICK_ACTIVITY	EXTRA_INTENT / 선택된 액티비티 클래스
데이터 입출력	ACTION_VIEW	URI/없음
	ACTION_PICK	URI/없음
	ACTION_INSERT	URI / URI
	ACTION_EDIT	URI/없음
	ACTION_DELETE	URI / 없음
	ACTION_GET_CONTENT	MIME 타입/URI
전화	ACTION_DIAL	URI / 없음
	ACTION_CALL	URI / 없음
	ACTION_ANSWER	없음/없음
메시징	ACTION_SEND	MIME, EXTRA_TEXT, EXTRA_STREAM / 없음
	ACTION_SENDTO	URI / 없음
	ACTION_ATTACH_DATA	URI/없음
기타	ACTION_RUN	? / 없음
	ACTION_SYNC	? / ?
	ACTION_SEARCH	getStringExtra(SearchManager.QUERY) / 없음
	ACTION_WEB_SEARCH	getStringExtra(SearchManager.QUERY) / 없음
	ACTION_FACTORY_TEST	없음/없음

인텐트 내장 액션의 활용

```
Intent intent;
// 전화 걸기 직전
intent = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:010-1234-5678"));
// 전화 걸기
intent = new Intent(Intent.ACTION_CALL, Uri.parse("tel:010-1234-5678"));

// 연락처보기 (통화 기록과 함께)
intent = new Intent(Intent.ACTION_VIEW, Uri.parse("content://contacts/people/"));
// 연락처만 따로 보기
intent = new Intent(Intent.ACTION_PICK, Uri.parse("content://contacts/people"));

// 사진앨범 (내장메모리)
intent = new Intent(Intent.ACTION_VIEW, Uri.parse("content://media/internal/images/media"));
// 사진앨범 (외장메모리)
intent = new Intent(Intent.ACTION_VIEW, Uri.parse("content://media/external/images/media"));

// 웹사이트 보기
intent = new Intent(Intent.ACTION_VIEW,   Uri.parse("http://okgosu.net"));
// 지도 보기
intent = new Intent(Intent.ACTION_VIEW,   Uri.parse("geo:132,37"));
```

인텐트 내장 액션의 활용

```
// 음악재생
intent = new Intent (Intent.ACTION_VIEW);
Uri uri = Uri.parse("file:///sdcard/1.mp3");
intent.setDataAndType (uri, "audio/mp3");
// 음악선택
intent = new Intent(Intent.ACTION_GET_CONTENT);
intent.setType("audio/*");
//startActivityForResult(Intent.createChooser(intent, "Select Audio Source..."), 0);

// SMS
intent = new Intent (Intent.ACTION_SENDTO, Uri.parse ( "smsto: 0800000123"));
intent.putExtra ( "sms_body", "The SMS text");
// MMS
intent = new Intent(Intent.ACTION_SEND);
Uri uri = Uri.parse ("content://media/external/images/media/1");
intent.putExtra ("address", "you@android.com");
intent.putExtra ("subject", "요청한 사진입니다.");
intent.putExtra ("sms_body", "그럼 이만... 감사합니다.");
intent.putExtra (Intent.EXTRA_STREAM, uri);
intent.setType ( "image/*");
startActivity(intent);
```

인텐트를 통한 컨텐트 프로바이더 활용

- 데이터를 입력/수정/삭제/조회를 위해 지정된 액션과 컨텐트 프로바이더에 정의된 URI를 사용함
  - 인텐트를 이용하여 지정된 액션과 URI를 처리하는 액티비티로 이동
    - 예)
      - String uriString = "content://okgosu.net.provider.MyProd/myprods";
      - Intent intent = new Intent(Intent.ACTION_VIEWL, Uri.parse(uriString));
      - startActivity(intent);
  - getContentResolver()를 통해서 단순히 데이터만 입력/수정/삭제/조회
    - 예)
      - Cursor cursor = getContentResolver().query(Uri.*parse*(uriString), selectionArgs, **null**, **null**, **null**);
      - startManagingCursor(cursor);

3.2.액티비티 작동원리

- 액티비티 상태
- 라이프사이클
- 액티비티 상태 활용 (실습)

액티비티 상태와 라이프사이클

- 액티비티 라이프 사이클
  - 화면에 보일 때부터 사라질 때까지 효율적으로 관리

- 액티비티의 효율적인 관리를 위해서
  - 액티비티는 상태를 가짐
  - 액티비티 상태변경시 콜백함수가 호출

액티비티 상태

- 실행여부와 화면상태 기준에 따라

	화면노출	사용자인터렉션	비고
활성	보임	가능	
일시정지	일부만 보임	불가능	전화, 알림창이 화면 일부를 가린 상태
정지	안보임	Notification으로만 가능	다른 액티비티가 화면을 완전히 가린 상태
종료	안보임	불가능	액티비티가 종료된 상태

액티비티 상태에 따른 콜백함수

- 액티비티 화면 시작
  - onCreate() -> onStart() -> onResume()
- 실행과 종료
  - onCreate()
    - 액티비티가 생성될 때
  - onDestroy()
    - 액티비티가 종료될 때
- 화면 등장 직전
  - onStart()
    - 액티비티가 화면에 보이기 직전에

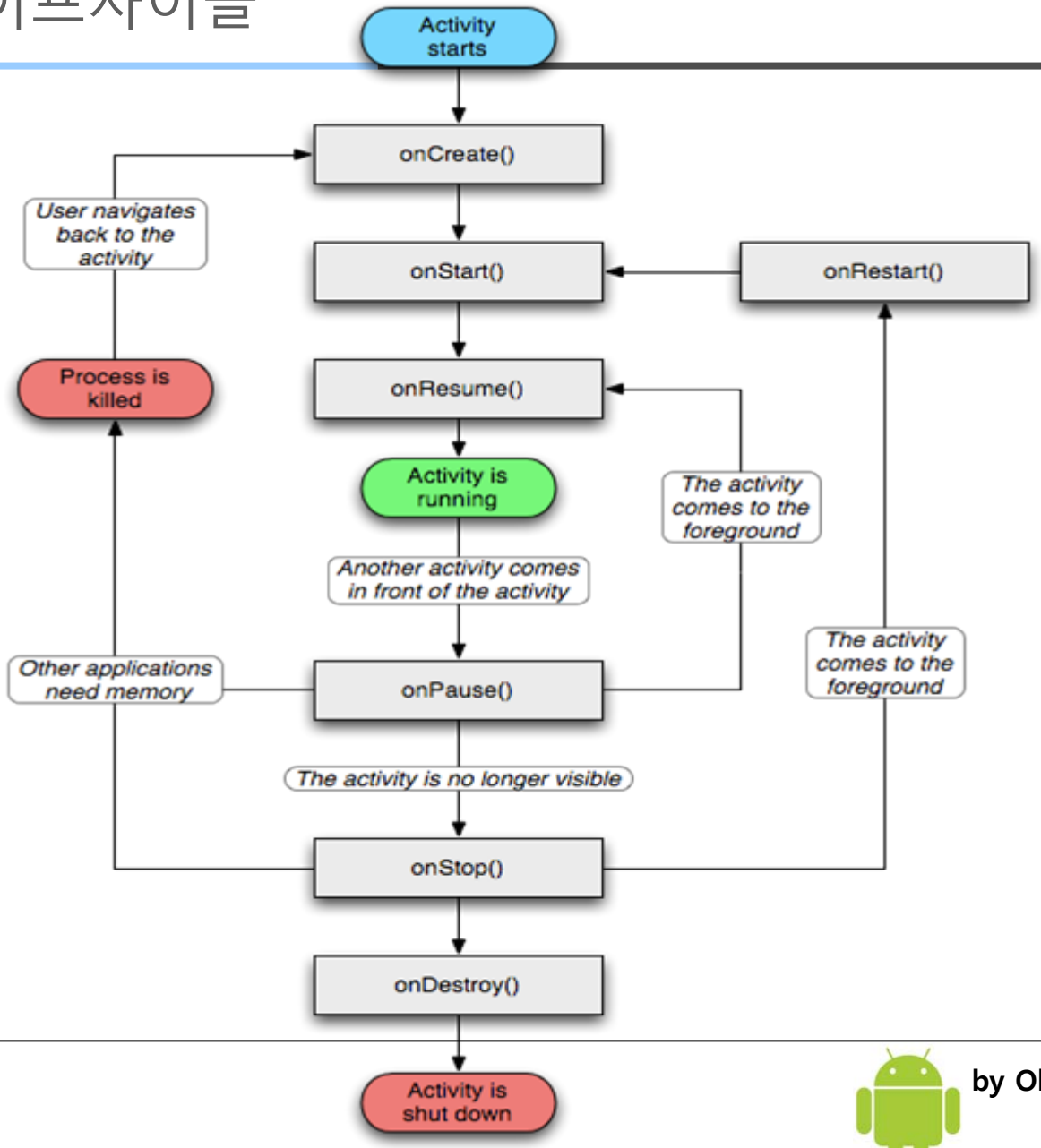
액티비티 상태에 따른 콜백함수

- 정지
  - onStop()
    - 액티비티가 정지 될 때
    - 강제 종료 가능
  - onRestart()
    - 액티비티가 정지되었다가 다시 실행될 때

- 화면 포커스 변화
  - onPause()
    - 화면 뒤로 물러날 때
    - 강제 종료 가능
    - 예) 백그라운드로 가거나 액티비티가 닫힐 때 작업내용 저장하기

  - onResume()
    - 처음 또는 정지상태에서 다시 실행될 때
    - 예) 기존 작업 파일 불러와서 보여주기

액티비티 라이프사이클


Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
Other applications need memory
onPause()
The activity comes to the foreground
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

액티비티 상태 처리

- 액티비티 라이프 사이클
  - 화면에 보일 때부터 사라질 때까지 효율적으로 관리

- 액티비티의 효율적인 관리를 위해서
  - 액티비티는 4가지 상태를 가짐
  - 액티비티 상태변경시 콜백함수가 호출

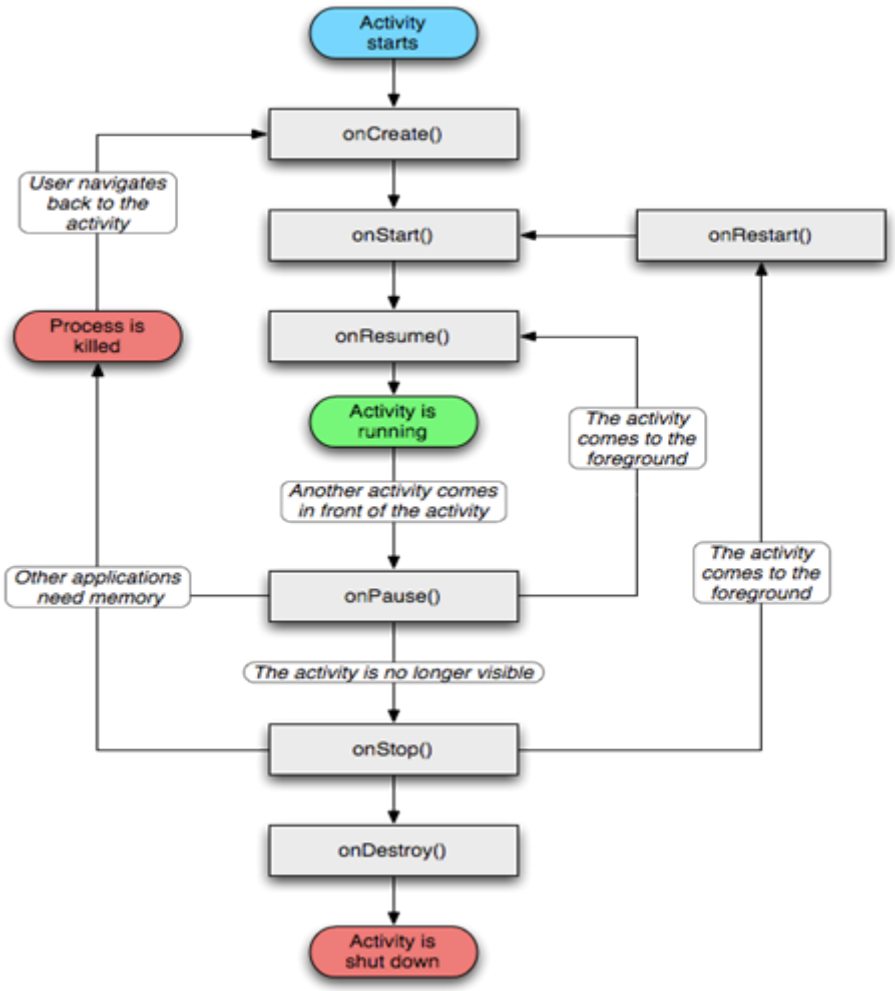
	화면노출	사용자인터렉션	비고
활성	보임	가능	
일시정지	일부만 보임	불가능	전화, 알림창이 화면 일부를 가린 상태
정지	안보임	Notification으로만 가능	다른 액티비티가 화면을 완전히 가린 상태
종료	안보임	불가능	액티비티가 종료된 상태

액티비티 상태 전이에 따른 콜백함수

- 액티비티 화면 시작
  - onCreate() -> onStart() -> onResume()
- 실행과 종료
  - onCreate() : 액티비티가 생성될 때
  - onDestroy() : 액티비티가 종료될 때
- 화면 등장 직전
  - onStart() : 액티비티가 화면에 보이기 직전에
- 정지
  - onStop(): 액티비티가 정지 될 때 (강제 종료 가능)
  - onRestart() : 액티비티가 정지되었다가 다시 실행될 때

액티비티 상태 전이에 따른 콜백함수

- 화면 포커스 변화
  - onPause()
    - 화면 뒤로 물러날 때
    - 강제 종료 가능
    - 예) 백그라운드로 가거나 액티비티가 닫힐 때 작업 내용 저장하기
  - onResume()
    - 처음 또는 정지상태에서 다시 실행될 때
    - 예) 기존 작업 파일 불러와서 보여주기



화면 회전 처리

- 화면이 회전하면 액티비티가 onCreate부터 다시 시작함
  - onPause에서 처리
    - 데이터는 Preference, File 또는 DB에 저장
    - 인텐트 호출로 다른 액티비티로 전환시 사용 가능
  - onSaveInstanceState 활용 (Bundle 메소드)
    - onStop전에 호출되어 Bundle객체에 상태를 저장함
    - onCreate(Bundle) 또는 onRestoreInstanceState(Bundle)에서 Bundle객체로 상태 복원
  - onPause와 onSaveInstanceState 차이점
    - onPause는 항상 호출된다.
    - onSaveInstanceState는 1) 화면이 회전하거나 2) 리소스 문제로 액티비티가 킬되어 조만간 그 상태를 복원할 때 호출

화면 회전 처리

- onSaveInstanceState 활용예

```
private String savedMsg=null;
private EditText ed;
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ed = new EditText(this);
        setContentView(ed);
        savedMsg = savedInstanceState ==null ? "null" :
savedInstanceState.getString("savedMsg");
        ed.setText(savedMsg);
}

@Override
protected void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
        outState.putString("savedMsg", ed.getText().toString());
}
```

기타 화면 회전 처리 방법

- onRetainNonConfigurationInstance
  - configuration이 바뀌면서 액티비티가 종료될 때 호출
  - Object객체에 상태를 저장함
  - onStop() and onDestroy()사이에 호출됨
  - onDestroy() 이후 새로운 환경에 대한 새 인스턴스 생성
  - getLastNonConfigurationInstance() 함수로 저장한 객체 복원

- onConfigurationChanged
  - 액티비티가 실행도중 환경설정이 바뀌었을 때
  - 환경설정내용 : mcc, mnc, locale, orientation, uiMode 등
    - http://d.android.com/reference/android/R.attr.html#configChanges
  - onConfigurationChanged가 호출되면 리소스객체를 업데이트하여 새로운 환경에 맞는 리소스를 적용한다.

- 화면 회전 방지
  - Manifest 파일에 android:screenOrientation="portrait"이라고 설정하면 화면이 회전하지 않음, android:screenOrientation="sensor"라고 하면 센서를 통한 화면 회전 감지

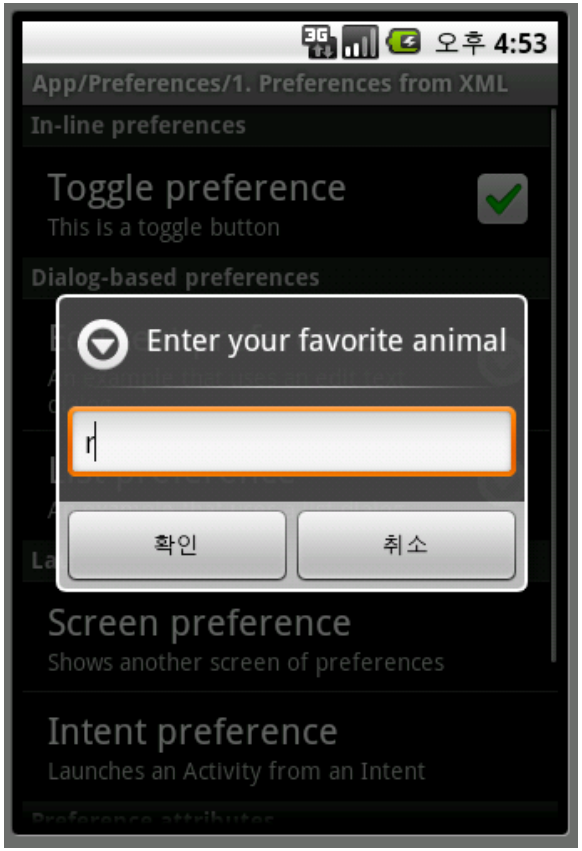
3.3. 환경 설정

- 환경 설정 개요
- UI구성과 환경설정 적용

환경설정 개요

- 환경설정 레벨
  - Activity 레벨
    - Activity의 getPreferences(mode)
  - Context 레벨
    - Context의 getSharedPreferences(name, mode)
  - 패키지 레벨
    - PreferencesManager의 getDefaultSharedPreferences(context)

- 환경설정모드
  - Activity.MODE_PRIVATE : 기본값
  - Activity.MODE_APPEND : 추가 가능
  - Activity.MODE_WORLD_READABLE : 다른 앱도 읽을 수 있게
  - Activity.MODE_WORLD_WRITEABLE : 다른 앱도 쓸 수 있게



환경설정 개요

- 환경 설정 저장
  - PreferenceActivity에서는 자동 저장
  - Activity는 Editor를 이용
    - SharedPreferences p=getSharedPreferences("mypref", Activity.MODE_PRIVATE);
    - SharedPreferences.Editor editor = p.edit();
    - editor.putString("myname", "okgosu");
    - editor.putInt("mynum", 1234);
    - editor.commit();

- 환경설정값 읽기
  - PreferenceActivity
    - PreferencesManager의 getDefaultSharedPreferences(context)
  - Activity
    - SharedPreferences p=getSharedPreferences("**mypref** ", Activity.MODE_PRIVATE);
    - Log.d("okgosu", p.getString("myname", "String"));
    - Log.d("okgosu", String.valueOf(p.getInt("mynum", 0)));

- 환경설정 저장 장소 확인
  - PreferenceActivity
    - DDMS/data/data/팩키지/shared_prefs/패키지이름_preferences.xml (PreferencesManager사용시)
  - Activity
    - DDMS/data/data/팩키지/shared_prefs/**mypref**.xml

XML을 이용한 환경설정 방법

- XML 설정파일을 res/xml에 저장

```
<PreferenceScreen
xmlns:android="http://schemas.android.com/apk/res/android">
    <PreferenceCategory android:title="@string/inline_preferences">
        <CheckBoxPreference
                android:key="checkbox_preference"
                android:title="@string/title_toggle_preference"
                android:summary="@string/summary_toggle_preference" />
    </PreferenceCategory>
</PreferenceScreen>
```

  - 주요 XML 설정파일 요소
    - 루트태그: PreferenceScreen
    - 설정항목 그룹 구분자 : PreferenceCategory
    - 체크박스 : CheckBoxPreference
    - 텍스트 : EditTextPreference
    - 리스트 : ListPreference
    - 벨소리 : RingTonePreference

XML을 이용한 환경설정 방법

- PreferenceActivity 상속
- onCreate() 오버라이딩
  - addPreferencesFromResource(R.xml.preference) 호출
- AndroidManifest.xml에 EditPreferences 액티비티 추가
  - <activity
    - android:name=".EditPreferences"
    - android:label="@string/app_name">
  - </activity>

3.4. 파일 관리

- 개요
  - 애플리케이션 패키지에 함께 들어간 파일 사용
  - 애플리케이션 실행 도중에 파일 생성해 사용
- 파일 사용
  - 바이너리파일 (읽기/쓰기 가능)
    - openFileInput, openFileOutput 사용
  - res/raw 폴더 (읽기만 가능)
    - 안드로이드에서 따로 처리하지 않고 애플리케이션과 배포
    - Resource를 통해 openRawResource()로 파일 오픈

3.5. 로컬 데이터베이스

- SQLite 개요
- SQLite API
- DB 작업

SQLite 개요

- 임베딩 DB
  - 저메모리
  - 빠른 처리속도

- 특징
  - 오픈소스
  - 표준 SQL 인터페이스 사용
  - 매니페스트 타입 사용
    - 컬럼 데이터타입에 해당하지 않는 타입도 저장 가능

SQLite API

- SQLiteOpenHelper 사용해 DB 연결
  - getWritableDatabase();
  - getReadableDatabase();

- 테이블 작업
  - 삭제
    - db.execSQL("DROP TABLE ...");
  - 생성
    - db.execSQL("CREATE TABLE tbl (id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT, val REAL);");

SQLite API

- 데이터 추가
  - db.execSQL에서 INSERT, UPDATE, DELETE SQL실행
    - 예) db.execSQL("INSERT INTO tbl (name, val) VALUES ('okgosu', 8)");
  - SQLiteDatabase의 insert(), update(), delete() 사용
    - ContentValues cv = new ContentValues();
    - cv.put(tbl.Name, "okgosu");
    - db.insert("tbl", getNullColumnHack(), cv);

SQLite API

- 데이터 조회
  - 실행방법
    - rawQuery() : SELECT문 직접 실행
    - query(): 메소드 인자로 각 부분의 값을 넘겨 실행
      - 테이블명, 컬럼 이름 배열, where 구문, where 인자값, group by, order by, having
    - SQLiteQueryBuilder 클래스
      - 컨텐트 프로바이더에 적용 가능
  - 조회 결과는 Cursor 이용
    - Cursor에서는 여러 건의 결과를 하나씩 받아오면서 처리 가능

SQLiteQueryBuilder 객체

- 특징
  - 컨텐트 프로바이더에 적용
  - UNION 이나 서브쿼리 같은 복잡한 쿼리 실행가능

- 사용 방법
  - SQLiteQueryBuilder 인스턴스 생성
  - 쿼리에 사용할 테이블 설정
  - SQL 구성 요소 설정 (컬럼명, WHERE절 등)
  - 쿼리 구문 실행

Cursor 객체

- 주요 메소드
  - 건수 : getCount()
  - 커서 이동: moveToFirst(), moveToNext(), isAfterLast()
  - 컬럼 이름: getColumnNames()
  - 기타
    - requery() : 쿼리 재실행
    - close() : 커서 자원 해제

SQLite 관리

- adb 명령어
  - cmd 에서 adb
  - shell로 들어가
    - sqlite3 옵션
    - 데이터베이스 파일 경로 지정
      - /data/data/pakcage명/database/db명ㅇㅇㅇ
- 파이어폭스 확장 플러그인
  - SQLite Manager 사용

3.6. 컨텐트 프로바이더

- 컨텐트 프로바이더 개요
- 컨텐트 프로바이더 주요 API

컨텐트 프로바이더 개요

- 애플리케이션의 데이터를 다른 애플리케이션에서 사용할 수 있도록 오픈API처럼 사용하는 것
  - URI 를 이용 데이터 조회, 입력, 수정, 저장 가능
    - 예) content://contacts/people

- URI 구성
  - 스키마, 데이터 네임스페이스, 인스턴스 ID
    - content:// : 스키마
    - contacts : 네임스페이스

컨텐트 프로바이더 주요 API

- ContentProvider 클래스 상속
  - onCreate()
  - query()
  - insert()
  - update()
  - delete()
  - getType()
- URI 정의
  - public static 상수값으로 URI 정의
- 속성정의
  - 접근할 데이터의 속성을 final 변수에 정의
- 매니페스트 설정
  - <provider>엘리먼트 추가

수고하셨습니다 ^^/