4장 : 데이터와 서버통신

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

데이터 관리

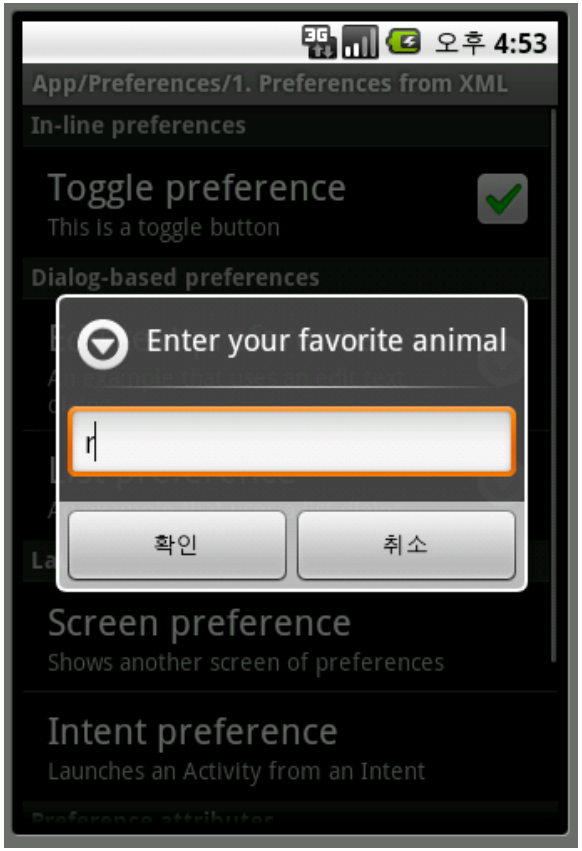
환경 설정

- 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>

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

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

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

서버통신

- 쓰레드
- 소켓 통신
- 서비스 객체
- AIDL

스마트폰의 애플리케이션의 작동

- 홈(Home)화면
  - 사용자가 전원을 켰을 때, 가장 먼저 보는 화면
  - 시간, 배경, 애플리케이션 목록 등을 보여줌
- 포어그라운드 애플리케이션
  - 사용자가 어플리케이션을 실행하면, 그 프로그램이 포어그라운드가 됨
  - 하나의 포어그라운드 애플리케이션은 화면 전체를 차지함
- 액티비티 매니저
  - 모든 프로그램과 화면은 시스템의 액티비티 매니저에 의해 애플리케이션 스택에 기록
  - 이전(back) 버튼을 누르면 이전 화면으로 이동
  - 가로, 세로 방향이 바뀌면, pause-stop-destroy-create로 액티비티 상태가 변화

쓰레드 기본

구현 & 실행

extends Thread

```
class MyThread extends Thread {
      public void run() {
         // 쓰레드 구현
      }
}
```

```
MyThread th = new MyThread();
th.start();
```

implements Runnable

```
class MyRunnable implements Runnable {
      public void run() {
         // 쓰레드 구현
      }
}
```

```
Runnable r = new MyRunnable();
Thread t = new Thread(r);
t.start();
```

쓰레드 기본

- 하나의 쓰레드에 대해 start()는 1회 호출가능
- **start()와 run()에 대한 차이와 쓰레드가 실행되는 과정**
-
- run()을 호출하는 것은 생성된 쓰레드를 실행시키는 것이 아니라 단순히 클래스에 속한 메서드 하나를 호출하는 것이다. 반면에 start()는 새로운 쓰레드가 작업을 실행하는데 필요한 호출스택(call stack)을 생성한 다음에 run()을 호출해서, 생성된 호출스택에 run()이 첫 번째로 저장되게 한다.
-
- 1. main메서드에서 쓰레드의 start메서드를 호출한다.
- 2. start메서드는 쓰레드가 작업을 수행하는데 사용될 새로운 호출스택을 생성한다.
- 3. 생성된 호출스택에 run메서드를 호출해서 쓰레드가 작업을 수행하도록 한다.
- 4. 이제는 호출스택이 2개이기 때문에 스케줄러가 정한 순서에 의해서 번갈아 가면서 실행된다.
-
- 한 쓰레드에서 예외가 발생해서 종료되어도 다른 쓰레드의 실행에는 영향을 미치지 않는다.

쓰레드

- 개요
  - 오래 걸리는 작업을 쓰레드로 안돌리면 현재 액티비티 뿐만 아니라 다른 애플리케이션도 작업 종료를 기다려야함
  - 쓰레드를 써야할 경우
    - 예) 파일 작업, 네트워크 조회, DB 트랜잭션
    - 사용자 조작에 대해서 5초 이내에 반응하지 않을때
  - Handler를 사용
    - 쓰레드(Child Thread)의 처리 결과를 화면(Main Thread)에서 처리할 수 있도록 해줌

서비스의 개요

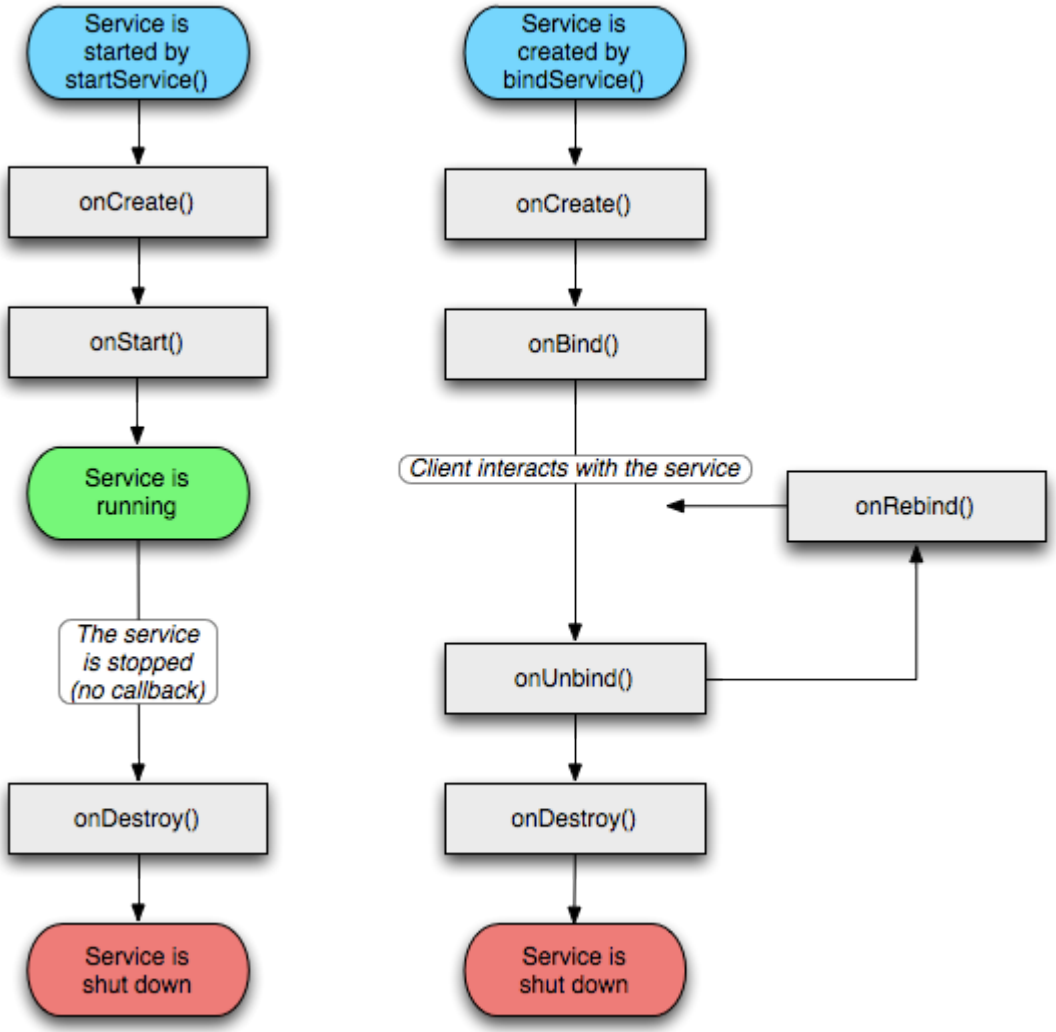
- 특징
  - 백그라운드에서 실행
  - 비활성 액티비티보다 높은 우선순위

- 용도
  - 사용자 입력에 직접 의존하지 않는 동작을 규칙적이며 연속적으로 수행할 때 사용
    - 예)
      - 파일 다운로드, RSS 리더, 미디어 플레이어
    - 안드로이드 시스템 서비스
      - Location Manger, Media Controller, Alarm Manager 등

서비스의 개요

- 서비스의 시작과 종료
  - 다른서비스, 엑티비티, 브로드캐스트 수신자를 포함한 다른 애플리케이션으로부터 가능
  - 예)
    - startService(new Intent(this, MyService.class);
    - stopService(new Intent(this, service.getClass());

서비스의 라이프사이클


Service is started by startService()
onCreate()
onStart()
Service is running
The service is stopped (no callback)
onDestroy()
Service is shut down
Service is created by bindService()
onCreate()
onBind()
Client interacts with the service
onRebind()
onUnbind()
onDestroy()
Service is shut down

서비스의 구현

- 서비스 클래스 작성
  - Service를 extends
  - onCreate(), onStart() 메소드 오버라이딩
  - Service를 Manifest에 등록
    - <service android:enabled="true" android:name=".MyService"></service>

- 서비스 사용
  - 서비스를 사용할 액티비티에서 startService() 호출
  - 한번 생성된 서비스를 다시 startService()로 실행할 때는 onCreate()는 타지 않고 onStart() 실행됨
  - 서비스가 여러 번 시작되었다하더라도 종료는 stopService() 한 번 호출로 종료

바인딩 서비스

- 개요
  - 일종의 프로세스간 통신
  - Binder는 리눅서 커널에 있는 Binder를 통해서 관리
  - 액티비티가 서비스의 특정 메소드를 호출하기 위함

- 특징
  - 서비스에 바인딩된 액티비티는 실행중에 서비스의 메소드 호출 가능
  - 서비스가 가지고 있는 모든 public 메서드와 프로퍼티는 onServiceConnected핸들러를 통해서 얻어진 serviceBinder객체를 통해서 이용 가능

AIDL : Android Interface Definition Language

- 개요
  - 서비스와 애플리케이션 콤포넌트간에 프로세스 간의 통신(IPC)을 지원하기 위한 안드로이드 인터페이스 정의 언어
  - Java에서 프로세스간 통신을 위한 IDL과 유사하며, AIDL은 안드로이드 AIDL 툴에서 자바 코드로 생성됨
  - Client와 Server사이의 값을 전달하기 위해 proxy클래스를 사용한다.
- AIDL이 지원하는 Data Type
  - Primitive Java Data Type
  - String, List, Map, CharSequence
  - 다른 AIDL-generated interface
  - Parcelable protocol 을 구현한 클래스 (커스텀 클래스 객체 전달)

AIDL : Android Interface Definition Language

- AIDL 파일
  - 프로세스간 호출가능한 서비스를 위해서 정의
  - 서비스가 구현할 인터페이스에 포함될 메서드와 필드를 정의
  - 예) IEarthquakeService.aidl
    - package com.paad.earthquake;
    - import com.paad.earthquake.Quake;
    - interface IEarthquakeService {
    - List<Quake> getEarthquakes();
    - void refreshEarthquakes();
    - }
  - ADT플러그인은 .aidl파일이 저장되면 "자바 interface파일 코드"가 자동 생성

AIDL : Android Interface Definition Language

- 액티비티에서 IPC 서비스 이용
  - IPC서비스를 바인드
  - ServiceConnection 클래스를 포함해서 onServiceConnected 메소드를 재정해서 사용

수고하셨습니다 ^^/