09장. 자료저장 및 관리

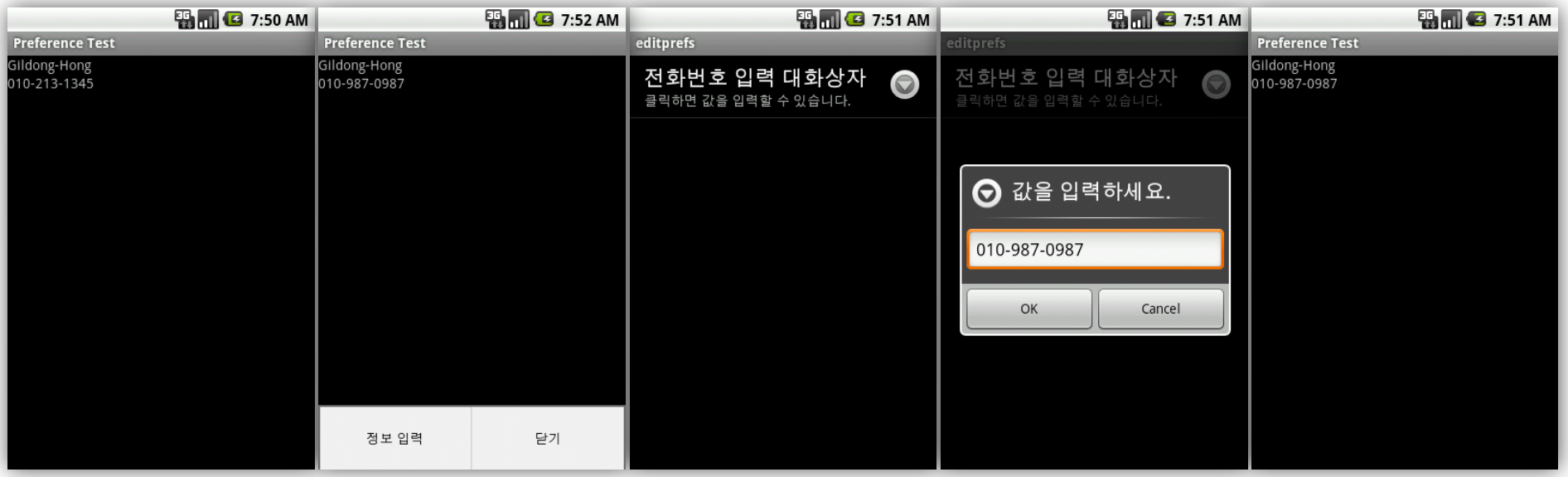
01. Preference 다루기
02. 파일과 폴더 다루기
03. SQLite 데이터베이스 다루기
04. 콘텐츠 제공자를 통한 자료 공유하기

컴퓨터공학과 변영철 교수

자료 저장/조회 방법

- Preferences
- 파일 시스템
- 내장 SQLite 데이터베이스
- 콘텐츠 제공자
  - 자신이 가지고 있는 데이터(콘텐츠)를 타 응용에게 제공하는 응용

Preference 특징



Preference 특징

- Preference 객체는 안드로이드가 제공하는 경량 자료 저장/조회 메커니즘을 제공함
  - 응용프로그램 상태
  - 간단한 사용자 정보
  - 구성 옵션 같은 자료를 저장하기에 적합
- 키/값 쌍으로 저장
  - 부울 값, 부동소수점 값, 정수 값, 긴 정수(long) 값, 문자열 값
- 하나의 Activity 내에서, 혹은 동일한 응용 내의 여러 Activity들 간에 자료 공유 가능
- 외부 응용과는 공유 불가

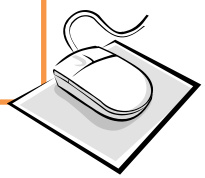
Preference 메소드

- 쓰기
  - putBoolean, putFloat, putInt, putLong, putString
- 읽기
  - getBoolean, getFloat, getInt, getLong, getString
  - getAll : 모든 키/값들을 담은 map 객체 반환
- contains
  - 특정 항목이 존재하는지 확인
- clear
  - 모든 항목 제거
- remove
  - 해당 키를 갖는 항목만 제거
- commit
  - 변경한(쓰는, put) 내용 반영

Preference 3가지 유형

- getPreferences 호출을 통한 Preference 객체 생성
  - 해당 Activity 내에서만 액세스 가능
  - /data/data/<package_name>/shared_prefs/액티비티이름.xml
- getSharedPreferences 호출을 통한 Preference 객체 생성
  - 동일 응용 내의 다른 Activity에서도 사용 가능
  - /data/data/<package_name>/shared_prefs/지정이름.xml
- getDefaultSharedPreferences 호출을 통한 생성
  - 다른 Activity에서도 사용 가능
  - /data/data /<package_name>/shared_prefs/패키지명_preferences.xml

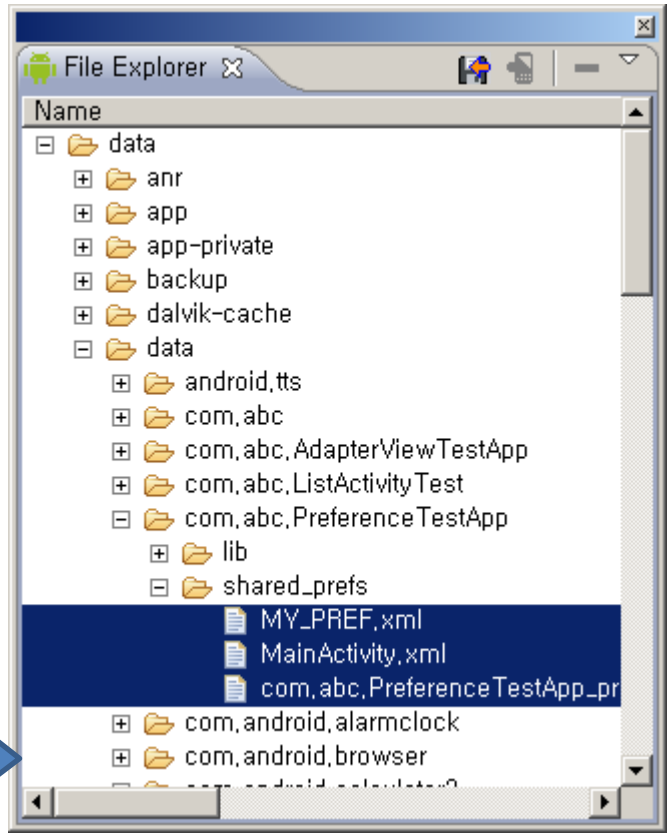
PreferenceTest



Preference별 파일명

/data/data/패키지명/shared_prefs/

- 액티비티명.xml
- 사용자가 지정한 이름.xml
- 패키지명.xml



DDMS

Preference가 저장된 예

- 키 – 값

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<map>
    <string name="NAME">Gildong-Hong</string>
    <int name="Age" value="24" />
    <float name="Weight" value="60.5" />
    <long name="TEMP" value="92233720368547758O7" />
    <boolean name="MALE" value="true" />
</map>
```

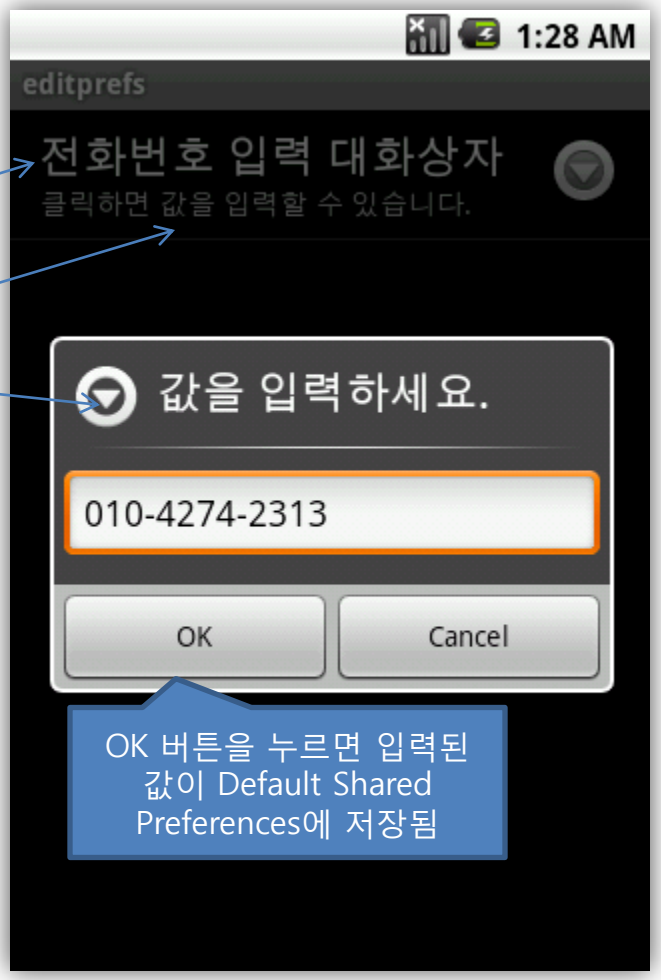
PreferenceActivity

/xml/pref_activity_info.xml

```
<PreferenceScreen
    xmlns:android="http://schemas.android.com/apk/res/android">

    <EditTextPreference
        android:key="TEL"
        android:title="전화번호 입력 대화상자"
        android:summary="클릭하면 값을 입력할 수 있습니다."
        android:dialogTitle="값을 입력하세요."/>

</PreferenceScreen>
```

MyPrefActivity.java

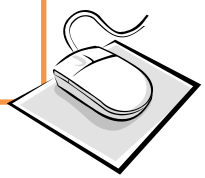
```
public class MyPrefActivity extends PreferenceActivity
{
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

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



PreferenceActivity

```
@Override
public void onResume() {
    super.onResume();

    SharedPreferences default_p =
        PreferenceManager.getDefaultSharedPreferences(this);
    String tmp = default_p.getString("TEL", "N/A");

    TextView tv2=(TextView)findViewById(R.id.textview2);
    tv2.setText(tmp);
}
```

PreferenceTest



파일과 디렉터리

- 각 안드로이드 응용은 리눅스 운영체제의 고유 사용자 계정으로 실행되며, 각자 자신만의 디렉터리와 파일들을 사용
- 텍스트 파일, 이진 파일, XML 파일 등 다양한 종류의 파일들을 처리하는 데 유용 : java.io 패키지에 있음
- 기본 파일 디렉터리
  - /data/data/패키지명/files/
- 그 외 디렉터리(캐쉬 파일 디렉터리의 경우)
  - /data/data/패키지명/cache/

관련 주요 메소드

메서드	용도
Context.openFileInput()	하위 디렉터리 /files에 있는 응용프로그램 파일을 읽기 모드로 연다.
Context.openFileOutput()	하위 디렉터리 /files에 있는 응용프로그램 파일을 쓰기 모드로 열거나 생성한다.
Context.deleteFile()	하위 디렉터리 /files에 있는 응용프로그램 파일을 삭제한다.
Context.fileList()	하위 디렉터리 /files의 모든 파일의 목록을 얻는다.
Context.getFilesDir()	하위 디렉터리 /files에 대한 객체를 얻는다.
Context.getCacheDir()	하위 디렉터리 /cache에 대한 객체를 얻는다.
Context.getDir()	주어진 이름의 응용프로그램 하위 디렉터리를 얻거나 생성한다.

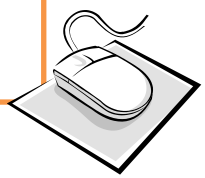
파일 생성 및 자료 저장

```
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    String file_name = "file1.xxx";

    // 파일 생성 후 문자열 저장
    FileOutputStream fo_stream;

    try {
        fo_stream = openFileOutput (file_name, MODE_PRIVATE);
        fo_stream.write(CONTENTS1.getBytes());
        fo_stream.close();

        Log.i(DEBUG_TAG, "Wrote to File: " + file_name);
    }
    catch (Exception e) {
        Log.i(DEBUG_TAG, "Exception: " + e.getMessage());
    }
}
```

파일 생성 : FileOutputStream 객체의 openFileOutput 메소드 호출

자료 저장 : write, close 메소드 호출

FileTest



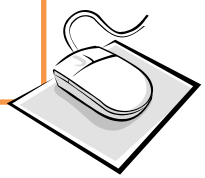
파일 읽기

```
FileInputStream fi_stream;

try {
      fi_stream = openFileInput (file_name);

      StringBuffer stringButter = new StringBuffer();
      int length = 70;
      byte[] bytearray = new byte[length];

      // read 70 bytes at a time
      while ((fi_stream.read(bytearray, 0, length)) != -1) {
            String str = new String(bytearray);
            stringButter.append(str + "₩n");
            if(fi_stream.available() < 70) {
                  //initialization
                  Arrays.fill(bytearray, 0, length, (byte) ' ');
            }
      }

      fi_stream.close();

      //로그에 출력함
      Log.i(DEBUG_TAG, "File Contents:₩n" + stringButter.toString());
}  catch (Exception e) {
      Log.i(DEBUG_TAG, "Exception: " + e.getMessage());
}
```

파일 읽기 :
openFileInput 객체의 read 메소드 호출로 가능

기본 디렉터리(/files 하위 디렉터리)에 있는 파일을 다루는 정도라면 Context.openFileOutput 메소드와 Context.openFileInput 메소드로 충분함.

FileTest



보다 일반적인 방법

- 디렉터리를 직접 지정하고자 할 경우 표준 java.io.File 클래스 이용

```
File abs_dir = getFilesDir();
String file_name = "myFile.dat";
String strFileContents = "파일에 저장할 내용";

File newFile = new File(abs_dir, file_name);

newFile.createNewFile();
FileOutputStream fo = new FileOutputStream(newFile.getAbsolutePath());

fo.write(strFileContents.getBytes());
fo.close();
```

생성자 함수 파라미터로 절대 경로 전달

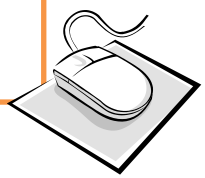
- File 객체를 이용하면 원하는 디렉터리 안에 파일이나 하위 디렉터리들을 생성할 수 있음(가령 캐시 파일 관리)

보다 일반적인 방법

```
String abs_path = getFilesDir() + "//" + file_name;

FileInputStream fi_stream;
try {
        fi_stream = new FileInputStream(abs_path);
        StringBuffer sBuffer = new StringBuffer();
        int length = 70;
        byte[] bf = new byte[length];

        // read 70 bytes at a time
        while ((fi_stream.read(bf, 0, length)) != -1) {
                String str = new String(bf);
                sBuffer.append(str + "₩n");
                if(fi_stream.available() < 70) {
                        //initialization
                        Arrays.fill(bf, 0, length, (byte) ' ');
                }
        }

        fi_stream.close();

        Log.i(DEBUG_TAG, "File Contents:₩n" + sBuffer.toString());
        Log.i(DEBUG_TAG, "₩nEOF");
}  catch (Exception e) {
        Log.i(DEBUG_TAG, "Exception: "+ e.getMessage());
}
```

절대 경로를 지정하여 파일을 읽는 방법

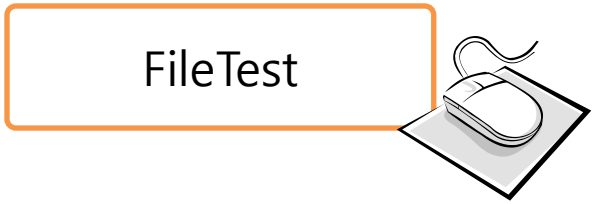
FileTest



캐시 파일 생성 및 자료 저장

- 응용 프로그램의 성능과 네트워크 접근 속도를 높이기 위해 일부 자료를 임시로 보관하고자 할 경우 흔히 캐시(cache) 파일을 사용
- 관례적으로 data/data/패키지명/cache/ 폴더에 저장

```
File abs_dir = getCacheDir();
String file_name = "my.cache";

File file = new File (abs_dir, file_name);
try {
    file.createNewFile();

    FileOutputStream fi_stream = new FileOutputStream(file.getAbsolutePath());
    fi_stream.write(CONTENTS1.getBytes());
    fi_stream.close();
}
catch (Exception e) {
    Log.i(DEBUG_TAG, "Exception: " + e.getMessage());
}
```



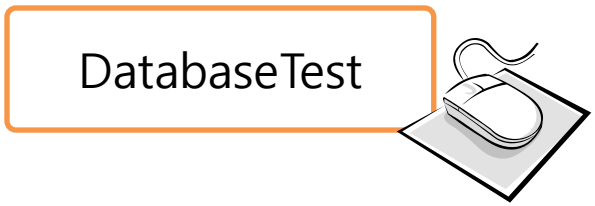
SQLite 데이터베이스

- 자료를 안정적이고 조직적으로 저장하고자 할 때 사용하는 방법
- 가벼운 파일 기반 데이터베이스
- android.database.sqlite 패키지에 SQLite 데이터베이스 관리에 유용한 여러 클래스들이 있음
- ADB 쉘에서 sqlite3 도구를 이용하여 데이터베이스를 직접 조작하는 것도 가능
- 데이터베이스(SQLite이든 아니든) 파일은 다음 디렉터리에 저장
  - /data/data/패키지명/databases/

DB 생성

- 가장 간단한 DB 생성 방법 : 응용의 Context 객체의 openOrCreateDatabase 메소드 호출

```
private static final String DB_FILE = "test.db";

@Override
public void onCreate(Bundle savedInstanceState) {
	super.onCreate(savedInstanceState);
	setContentView(R.layout.main);

	// DB 파일이 이미 존재하면 제거
	if (Arrays.binarySearch (databaseList(), DB_FILE) >= 0) { //test.db
		deleteDatabase(DB_FILE);
	}

	// 데이터베이스 생성
	SQLiteDatabase mDatabase;
	mDatabase =
		openOrCreateDatabase(DB_FILE, SQLiteDatabase.CREATE_IF_NECESSARY, null);
}
```



DB 설정 및 정보 표시

- 데이터베이스의 주요 구성 속성 : 버전, 로캘, 스레드에 안전한 잠금 활성화 여부 등

```
mDatabase.setLocale (Locale.getDefault());

// 임계구역에서 잠기는 것을 허용함으로써 데이터베이스가 thread-safe하게 됨
mDatabase.setLockingEnabled (true);

// DB 버전 설정
mDatabase.setVersion(1);

// 데이터베이스 정보 표시
Log.i(DEBUG_TAG, "Created database: " + mDatabase.getPath());
Log.i(DEBUG_TAG, "Database Version: " + mDatabase.getVersion());
Log.i(DEBUG_TAG, "Database Page Size: " + mDatabase.getPageSize());
Log.i(DEBUG_TAG, "Database Max Size: " + mDatabase.getMaximumSize());

Log.i(DEBUG_TAG, "Database Open?  " + mDatabase.isOpen());
Log.i(DEBUG_TAG, "Database read-only?  " + mDatabase.isReadOnly());
Log.i(DEBUG_TAG, "Database Locked by current thread? "
      + mDatabase.isDbLockedByCurrentThread());
```

스키마와 테이블 생성

- 테이블을 생성하는 SQL의 예

```
CREATE TABLE tbl_authors (id INTEGER PRIMARY KEY AUTOINCREMENT ,
    firstname TEXT,
    lastname TEXT);
```

- SQLiteDatabase의 execSQL 메소드 이용한 테이블 생성 예

```
String CREATE_AUTHOR_TABLE =
    "CREATE TABLE tbl_authors ( id INTEGER PRIMARY KEY AUTOINCREMENT ,
    firstname TEXT,
    lastname TEXT);";

mDatabase.execSQL(CREATE_AUTHOR_TABLE);
```

스키마와 테이블 생성

- SQLiteStatement의 execute 메소드를 이용한 생성 예

```
String CREATE_BOOK_TABLE =
    "CREATE TABLE tbl_books (
    id INTEGER PRIMARY KEY AUTOINCREMENT ,
    title TEXT,
    dateadded DATE,
    authorid INTEGER NOT NULL CONSTRAINT authorid REFERENCES
        tbl_authors(id) ON DELETE CASCADE);";

SQLiteStatement sqlSelect = Database.compileStatement(CREATE_BOOK_TABLE);
sqlSelect.execute();
```

트리거를 이용한 제약

- 트리거 : '방아쇠'
- DB에 추가, 삭제, 갱신 등이 실행되었을 때 자동으로 수행되는 DB 객체
- 책에 반드시 유효한 저자가 있도록 하는 트리거를 생성하는 SQL 명령문

```
private static final String CREATE_TRIGGER_ADD =
    "CREATE TRIGGER fk_insert_book BEFORE INSERT ON tbl_books
    FOR EACH ROW BEGIN  SELECT RAISE (ROLLBACK, 'insert on table
    ₩"tbl_books₩" violates foreign key constraint ₩"fk_authorid₩"')
    WHERE  (SELECT id FROM tbl_authors WHERE id = NEW.authorid)
    IS NULL; END;";

mDatabase.execSQL(CREATE_TRIGGER_ADD);
```

레코드 삽입

- 테이블에 새 레코드를 추가하려면 insert 메소드를 이용함
- 이 메소드는 삽입할 열의 이름과 값의 쌍을 담은 ContentValues 객체를 요구
- 다음은 저자 테이블 tbl_authors에 J.K. Rowling 레코드를 추가하는 예

```
ContentValues values = new ContentValues();
values.put("firstname", "J.K."); //필드명과 값
values.put("lastname", "Rowling");
long newAuthorID = mDatabase.insert("tbl_authors", null, values);
```

- 반환값으로 추가된 레코드의 ID를 돌려주며, 이는 이후 이 저자에 대한 책 레코드를 생성할 때 쓰임
- insertOrThrow라는 메소드를 이용하면 insert와 같은 일을 하되, 삽입 실패 시 SQLException 예외 발생

레코드 갱신

- 기존 레코드를 갱신(수정)할 때는 update 메소드 이용
- update 메소드 인자
  - 레코드들을 갱신할 테이블
  - 갱신할 필드 명과 값을 담은 ContentValues 객체
  - 추가적인 WHERE 절을 담은 문자열 : '?' 문자들은 그 다음의 인수로 주어진 배열의 값들로 치환
  - WHERE 절의 각 '?' 문자를 대신할 값들의 배열
- 책 ID와 책 제목을 받아 tbl_books에서 해당 레코드의 책 제목 변경하는 예

```
public void updateBookTitle(Integer bookId, String newtitle) {
    ContentValues values = new ContentValues();
    values.put("title", newtitle);
    mDatabase.update("tbl_books", values, "id=?", new String[]{bookId.toString()});
}
```

레코드 삭제

- 레코드를 삭제할 때에는 delete 메소드 호출
- delete 메소드 인자
  - 레코드들을 삭제할 테이블
  - 추가적인 WHERE 절을 담은 문자열. 이 WHERE 절 안의 '?' 문자들은 다음 인수로 주어진 배열 값으로 치환
  - WHERE 절의 각 '?' 문자를 대신할 값들의 배열
- 셋째 인수(WHERE 절)로 null을 넣으면 테이블의 모든 레코드가 삭제됨
- tbl_authors 테이블의 모든 레코드를 삭제하는 예

```
mDatabase.delete("tbl_authors", null, null);
```

레코드 삭제

- tbl_books 테이블에서 특정 책 ID를 갖는 레코드 삭제

```
public void deleteBook(Integer bookId) {
    mDatabase.delete("tbl_books", "id=?", new String[] bookId.toString() });
}
```

- tbl_books 테이블에서 특정 저자가 쓴 모든 책 레코드를 삭제하는 메소드 작성 예

```
public void deleteBooksByAuthor(Integer authorID) {
    int numBooksDeleted = mDatabase.delete("tbl_books", "authorid=?",
        new String[] { authorID.toString() });
}
```

트랜잭션으로 수행하기

- 트랜잭션 : 모두 함께 수행되어야 하는 여러 연산 묶음
- 여러 데이터베이스 연산들이 반드시 모두 성공해야 하는 경우가 있으며, 이 경우 연산들 중 하나라도 실패하면 연산들 모두를 실행 이전으로 되돌림(rollback)

```
mDatabase.beginTransaction();
try {
    // 여기서 레코드 삽입, 갱신, 삭제 등등 하나의 단위로서
    // 수행되어야 할 데이터베이스 연산들을 수행함.

    // 트랜잭션 성공 및 commit
    mDatabase.setTransactionSuccessful();
}
catch (Exception e) {
    // 트랜잭션이 실패하였으며, 여기에서 적절한 조치(rollback 등)를 취함.
}
finally {
    mDatabase.endTransaction();
}
```

질의와 커서

- 원하는 자료를 다양한 방식으로 조회할 수 있음
  - 문자열 형태의 SQL 질의문을 직접 수행
  - 여러 SQL 질의문 구축 클래스들을 이용해서 질의 수행 객체를 구성
  - 데이터베이스를 컨테이너 뷰 등의 특정 사용자 인터페이스 위젯에 묶을 수도 있음
- SQL 질의문을 수행하여 만들어진 결과 집합에 접근할 때에는 커서(Cursor) 객체 이용

```
// select * from tbl_books
Cursor c = mDatabase.query("tbl_books",null,null,null,null,null,null);
// ... 여기서 커서 c를 사용하여 레코드들을 액세스함

c.close(); // 바로 닫아준다.
```

질의와 커서

- Cursor 객체들은 반드시 응용프로그램 수명주기의 일부로서 관리해야 함
  - 응용프로그램이 정지 또는 종료되면 deactivate 메소드를 호출해서 커서를 비활성화시켜야 함
  - 응용프로그램이 재시작되면 requery 메소드를 호출하여 결과 집합을 다시 얻어야 함
  - 커서를 다 썼다면 close를 호출하여 커서가 가지고 있던 자원들을 해제해야 함
- startManagingCursor 메소드
  - 일일이 처리하기가 귀찮을 경우 Cursor 객체의 관리를 현재 Activity 객체에 맡길 수도 있음
  - 이 함수를 호출하면 해당 Activity가 커서를 자동으로 관리

질의와 커서

- startManagingCursor 메소드를 호출하는 예

```
// select * from tbl_books
Cursor c = mDatabase.query("tbl_books",null,null,null,null,null,null);
startManagingCursor(c);
```

- Cursor 객체를 받아서 레코드 수를 출력하고 열들의 정보를 출력하는 예

```
Log.i(DEBUG_TAG,
      "Count: " + c.getCount() + " Columns: " + c.getColumnCount());

// 열 이름들을 출력한다.
String rowHeaders = "|| ";
for (int i = 0; i < c.getColumnCount(); i++) {
      rowHeaders = rowHeaders.concat(c.getColumnName(i) + " || ");
}

Log.i(DEBUG_TAG, "COLUMNS " + rowHeaders);
```

질의 수행 방법

- query 메소드 인자들
  - [String] : 질의할 테이블 이름
  - [String Array] : 결과 집합에 포함시킬 열 이름들의 목록. null을 지정하면 모든 열이 포함됨
  - [String] WHERE 절(치환될 선택 인수들을 '?'로 표시) : null을 지정하면 모든 레코드가 선택됨
  - [String Array] : WHERE 절의 '?'들에 치환될 선택 인수 값들
  - [String] GROUP BY 절 : null 지정 가능
  - [String] HAVING 절 : GROUP BY 절로서 null을 지정 가능
  - [String] ORDER BY 절 : null 지정 가능
  - [String] LIMIT 절 : 레코드 수를 제한할 필요 없으면 null 지정

질의 수행의 예

- SELECT * tbl_books WHERE id=9;

```
Cursor c = mDatabase.query("tbl_books", null, "id=?", new String[]{"9"},
null, null, null);
```

- SELECT title, id FROM tbl_books ORDER BY title ASC;

```
String strSortOrder = "title ASC";
Cursor c = mDatabase.query("tbl_books", asColumnsToReturn, null,
null, null, null, strSortOrder);
```

SQLiteQueryBuilder

- 여러 개의 테이블들에 대한 질의와 같이 질의가 복잡하면 SQLiteQueryBuilder를 이용함
- 복잡한 질의를 체계적으로 구축할 수 있음
- 두 테이블을 결합하여 책 정보와 저자 정보가 함께 있는 레코드들을 얻기 위한 질의를 구축하고 실행하는 예

```
SELECT tbl_books.title, tbl_books.id, tbl_authors.firstname, tbl_authors.lastname,
tbl_books.authorid FROM tbl_books INNER JOIN tbl_authors on
tbl_books.authorid=tbl_authors.id ORDER BY title ASC;
```

```
SQLiteQueryBuilder queryBuilder = new SQLiteQueryBuilder();
queryBuilder.setTables("tbl_books, tbl_authors");
queryBuilder.appendWhere("tbl_books.authorid=tbl_authors.id");
String asColumnsToReturn[] = {
      "tbl_books.title", "tbl_books.id", "tbl_authors.firstname",
      "tbl_authors.lastname", "tbl_books.authorid" };
String strSortOrder = "title ASC";
Cursor c = queryBuilder.query(mDatabase, asColumnsToReturn, null, null, null,
      null, strSortOrder);
```

SQL 질의 바로 실행하기

- "ow"라는 부분 문자열이 있는 저자들(Hallows, Rowling 등)과 저자들의 책 제목을 모두 얻는 SQL UNION 질의

```
SELECT title AS Name,
'tbl_books' AS OriginalTable
FROM tbl_books
WHERE Name LIKE '%ow%'
UNION
SELECT (firstname||' '|| lastname) AS Name,
'tbl_authors' AS OriginalTable
FROM tbl_authors
WHERE Name LIKE '%ow%'
ORDER BY Name ASC;
```

```
String sqlUnionExample = "SELECT title AS Name, 'tbl_books' AS
OriginalTable from tbl_books WHERE Name LIKE ? UNION SELECT (firstname||' '||
lastname) AS Name, 'tbl_authors' AS OriginalTable from tbl_authors WHERE Name
LIKE ? ORDER BY Name ASC;";

Cursor c = mDatabase.rawQuery(sqlUnionExample, new String[]{ "%ow%", "%ow%"});
```

SQLite 데이터베이스 종료와 삭제

- 데이터베이스를 사용한 후에는 반드시 닫아야 함

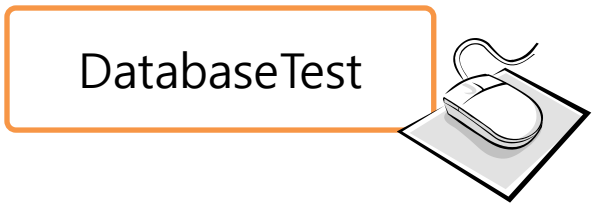
```
mDatabase.close();
```

- 필요 없는 테이블도 삭제할 수 있음 (두 테이블과 하나의 트리거를 삭제하는 예)

```
mDatabase.execSQL("DROP TABLE tbl_books;");
mDatabase.execSQL("DROP TABLE tbl_authors;");
mDatabase.execSQL("DROP TRIGGER IF EXISTS fk_insert_book;");
```

- 데이터베이스 삭제는 Context의 deleteDatabase 메소드를 통하여 수행

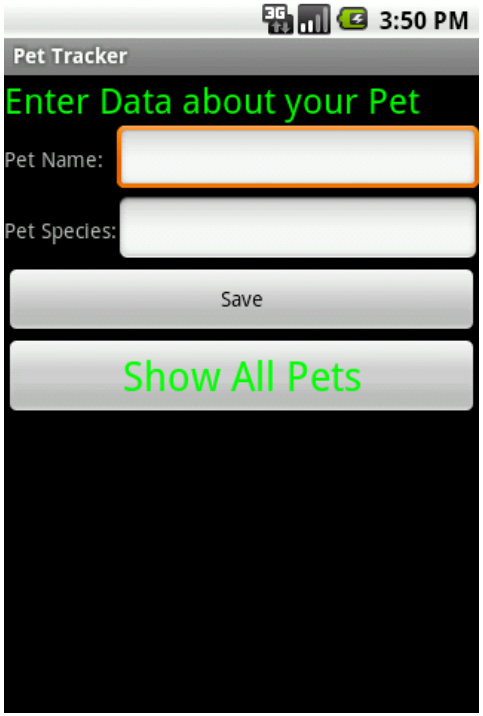
```
deleteDatabase("my_sqlite_database.db");
```

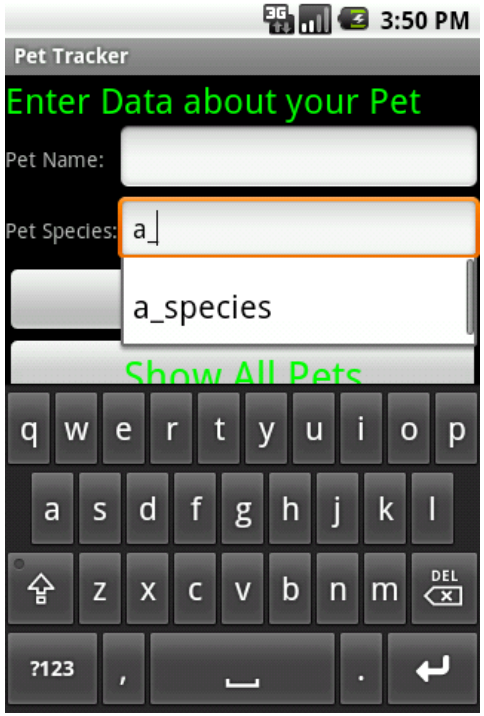


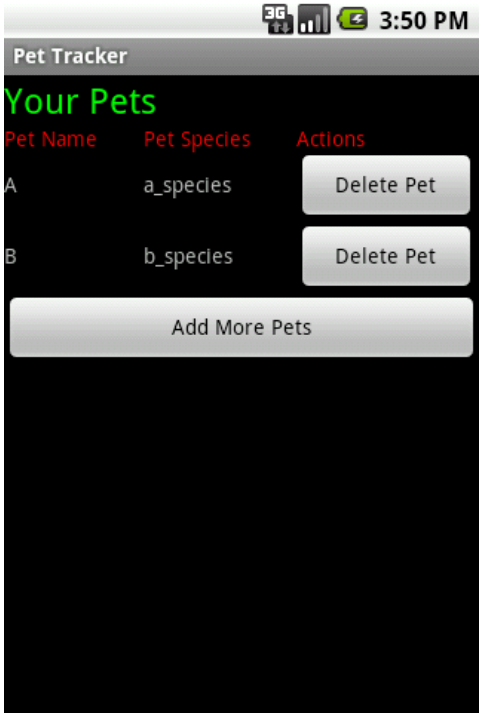
영속적 데이터베이스 설계

- 데이터베이스 필드 이름관리
  - SQL 질의문에서 오타 등의 문제점을 피할 수 있도록 테이블 필드 이름들을 코드에서 좀 더 체계적으로 조직화
  - 데이터베이스 스키마를 하나의 클래스로 캡슐화
  - BaseColumns 인터페이스로 구현하면 필드를 데이터베이스 대응 사용자 인터페이스 위젯에 매핑가능
- SQLiteOpenHelper 클래스 확장
  - onCreate, onUpgrade, onOpen 가상함수를 재정의해야 함
  - 읽기/쓰기 전용 데이터베이스 객체 접근 가능

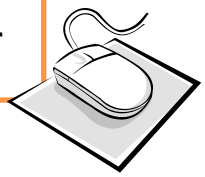
자료와 UI 연결







SuperPetTracker



자료와 UI 연결

- DB와 UI 연결 코드를 직접 손으로 작성
  - 모든 작업이 주 스레드에서 일어나기 때문에 레코드가 많이 추가될수록 응용프로그램의 반응 시간이 길어짐
  - 데이터베이스 질의 결과를 개별 사용자 인터페이스 위젯들에 연결하는 데 상당히 많은 양의 커스텀 코드가 필요
  - 자료를 다른 위젯으로 표시하고자 한다면 코드를 크게 수정
  - 데이터베이스에 대한 질의가 빈번하며, 이 역시 성능에 나쁜 영향

자료와 UI 연결

- 아답터를 이용한 데이터 바인딩
  - 아답터 : 데이터 소스(data source)에서 자료를 가져온 후 지정한 레이아웃으로 자식 뷰들을 만들고 뷰 컨테이너를 채우는 객체
  - 자료를 UI 위젯들과 바인딩하여 자료 표시나 갱신이 자동으로 일어나게 함으로써 앞의 문제점 극복
  - 아답터를 이용하여 커서를 TableLayout 대신 ListView 안의 각 TextView 자식 위젯에 바인딩

폰에 있는 사진 활용하기

- 사진을 촬영하거나 이미지 내려 받기
  - http://www.perlgurl.org/archives/2004/03/rabbits_rabbits.html
  - DDMS를 이용한 /sdcard/DCIM/Camera 폴더에 드래그-앤-드롭
- MediaPetTracker 실행해보기
  - 오류가 발생할 경우 카메라 프로그램 실행하여 이미지 확인 후 다시 실행
- MediaStore 콘텐츠 제공자 질의와 URI
  - DB 질의와 같이 콘텐츠 제공자에게 질의하는 형태로 접근
  - 콘텐츠 제공자에 따라 미리 정해진 URI로 질의
  - DB와 마찬가지로 커서를 이용하여 접근
  - 외부 매체 저장장치(SD 카드)에 접근하기 위한 URI

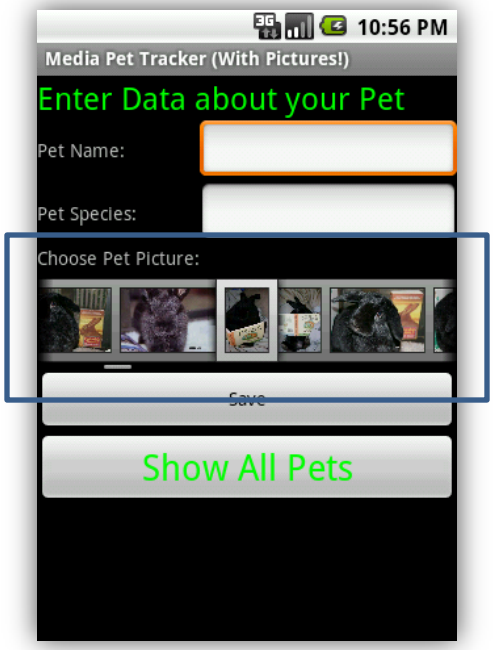
```
Uri mMedia = Media.EXTERNAL_CONTENT_URI;
```

폰에 있는 사진 활용하기

- 콘텐츠 제공자로부터 자료 가져오기(managedQuery)

```
String[] projection = new String[] { Media._ID, Media.TITLE };
Uri mMedia = Images.Media.EXTERNAL_CONTENT_URI;
Cursor mCursorImages = managedQuery(mMedia, projection, null, null,
   Media.DATE_TAKEN + "ASC"); // ORDER BY 절
```

  - 이제 커서 객체를 이용하여 매체에 저장되어 있는 모든 이미지 접근 가능
- 커서를 Gallery 위젯에 연결하기
  - 앞서 생성한 커서를 이용하여 아답터(ImageUriAdapter)를 만들고 이를 Gallery 위젯에 설정(setAdapter)
  - ImageUriAdapter는 커서가 가리키는 결과를 GalleryRecord 객체 배열 생성
  - 각 GalleryRecord 객체는 기준 URI와 이미지 ID를 담음



폰에 있는 사진 활용하기

- Save 기능 구현
  - 입력한 정보, 선택한 사진의 URI 및 ID 정보 등을 레코드 객체로 생성한 후 데이터베이스에 삽입(insert 명령)
- Show All Pets 기능 구현
  - DisplayActivity
    - displayPets
      - MyBaseAdapter
        - » getView

내장 콘텐츠 제공자

제공자	콘텐츠
MediaStore	기기와 외부 저장 매체의 오디오, 이미지, 동영상 자료
CallLog	걸고 받은 통화 기록
Browser	브라우저 히스토리와 북마크
Contacts	주소록의 연락처 자료
UserDictionary	예측식 텍스트 입력에 쓰이는 사용자 정의 단어 사전(안드로이드 1.5 이상에서만)

MediaStore 콘텐츠 제공자

- 안드로이드 기기와 외부 저장 매체에 있는 오디오, 이미지, 동영상을 제공
- android.provider.MediaStore 내부 클래스들

```
import android.provider.MediaStore.*;
```

클래스	용도
Video.Media	기기의 동영상 파일들을 관리
Images.Media	기기의 이미지 파일들을 관리
Images.ThumNails	이미지 섬네일들을 얻음
Audio.Media	기기의 오디오 파일들을 관리
Audio.Albums	기기의 앨범별 오디오 파일들을 관리
Audio.Artists	기기의 아티스트별 오디오 파일들을 관리
Audio.Genres	기기의 장르별 오디오 파일들을 관리
Audio.Playlists	기기의 재생 목록별 오디오 파일들을 관리

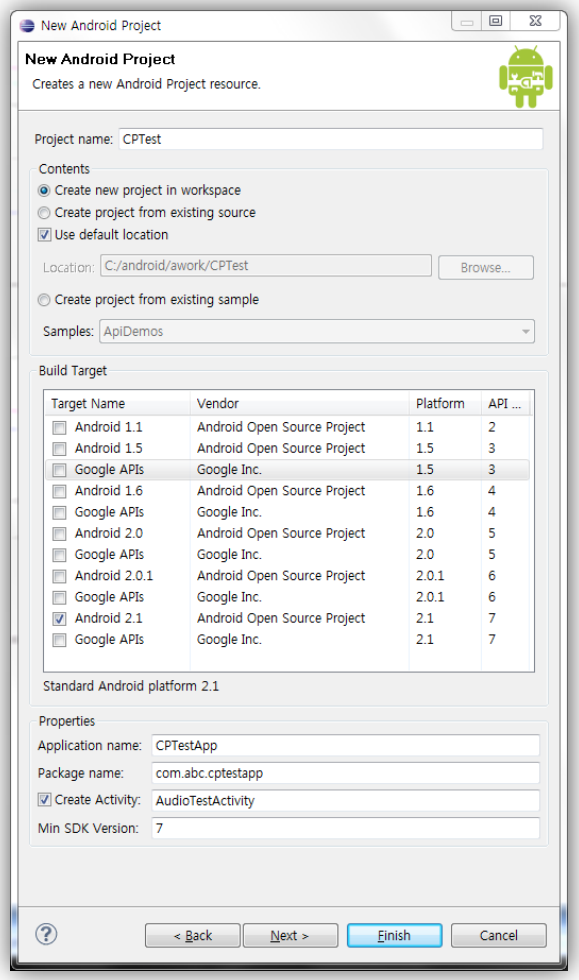
- 콘텐츠 접근 URI : Images.Media.EXTERNAL_CONTENT_URI

MediaStore 콘텐츠 제공자

- 이미지를 갤러리 위젯에 채우는 예제

```
import android.provider.MediaStore.*;

private void fillGallery() {
    // 반환할 컬럼(field)을 기술 : 실제로는 TITLE을 사용하지 않음.
    String[] fields = new String[] { Images.Media._ID, Images.Media.TITLE };

    // SD 카드 콘텐츠 접근 URI
    Uri content_uri = Images.Media.EXTERNAL_CONTENT_URI;

    // 콘텐츠 제공자 쿼리를 수행하여 SD 카드에 있는 모든 이미지(_ID와 TITLE 정보)를 얻음
    images_cursor = managedQuery(content_uri, fields, null, null,
        Images.Media.DATE_TAKEN + " ASC"); // Order-by 절

    // 커서를 이용하여 아답터 생성
    MyBaseAdapter myAdapter = new MyBaseAdapter(this, images_cursor, content_uri);

    // 갤러리 위젯에 아답터 설정 -> 이미지 자동 표시
    final Gallery pictureGal = (Gallery) findViewById(R.id.GalleryOfPics);
    pictureGal.setAdapter(myAdapter);
}
```

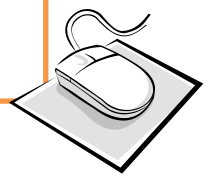
MediaStore 콘텐츠 제공자




```
import android.provider.MediaStore.*;
import android.util.Log;

public class AudioTestActivity extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        String[] requestedColumns = { Audio.Media.TITLE, Audio.Media.DURATION };

        Cursor cur = managedQuery(Audio.Media.EXTERNAL_CONTENT_URI,
            requestedColumns, null, null, null);

        Log.d("MyLog", "Audio files: " + cur.getCount());
        Log.d("MyLog", "Columns: " + cur.getColumnCount());

        //커서에게 제목(TITLE)과 재생시간(DURATION) 필드의 위치를 물음
        int pos1 = cur.getColumnIndex(Audio.Media.TITLE); // 제목의 컬럼 인덱스
        int pos2 = cur.getColumnIndex(Audio.Media.DURATION); // 재생시간의 컬럼 인덱스

        cur.moveToFirst();
        while (!cur.isAfterLast()) { // 마지막의 다음(끝)이 아니라면
            Log.d("MyLog", "Title" + cur.getString(pos1));
            Log.d("MyLog", "Length: " + cur.getInt(pos2) / 1000 + " seconds");
            cur.moveToNext();
        }
    }
}
```

SD 카드의 모든 오디오 파일의 제목과 재생 시간을 알아내는 방법

CPTest
AudioTestActivity



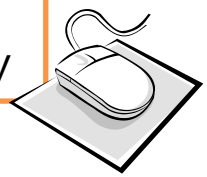
CallLog 콘텐츠 제공자

- 안드로이드는 통화가 일어날 때마다 해당 전화번호와 날짜, 통화 시간을 저장
- CallLog 콘텐츠 제공자는 통화 기록을 제공
- 걸거나 받은 전화번호나 통화 실패 기록을 조회할 수 있음
- 고객 관계 관리(CRM) 응용에 유용
- 권한 설정(AndroidManifest.xml)을 해야 함

```
<uses-permission xmlns:android="http://schemas.android.com/apk/res/android"
    android:name="android.permission.READ_CONTACTS">
</uses-permission>
```

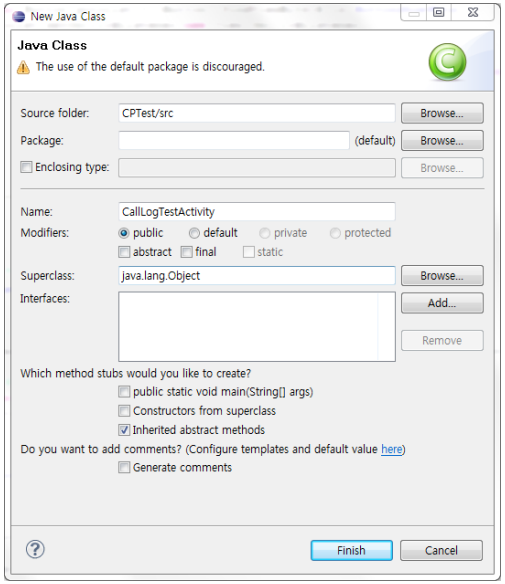
- 연락처 접근하기 위한 권한과 동일 : CallLog 콘텐츠 제공자 자료는 연락처 콘텐츠 제공자 자료와 유사

CPTest
CallLogTestActivity



CallLog 콘텐츠 제공자

클래스 생성 : File > New > Class
onCreate 자동 생성 : Source > Override/Implement Methods...



AndroidManifest.xml

```
<uses-permission
xmlns:android="http://schemas.android.com/apk/res/android"
android:name="android.permission.READ_CONTACTS">
</uses-permission>
```

```
import android.provider.*;
import android.util.Log;

public class CallLogTestActivity extends Activity
{
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        String[] requestedColumns = {
            CallLog.Calls.CACHED_NUMBER_LABEL,
            CallLog.Calls.DURATION
        };

        try {
            Cursor calls = managedQuery(CallLog.Calls.CONTENT_URI, requestedColumns,
            CallLog.Calls.CACHED_NUMBER_LABEL + " = ?", new String[] {"Gildong"} , null);
            Log.d("MyLog", "Call count: " + calls.getCount());

            int position = calls.getColumnIndex(CallLog.Calls.DURATION);

            int totalDuration = 0;
            calls.moveToFirst();
            while (!calls.isAfterLast()) {
                Log.d("MyLog", "Duration: " + calls.getInt(position));
                totalDuration += calls.getInt(position);
                calls.moveToNext();
            }

            Log.d("MyLog", "Total time : " + totalDuration);
        }
        catch (Exception e) {
            Log.d("MyLog", "Exception!" + e.toString());
        }
    }
}
```

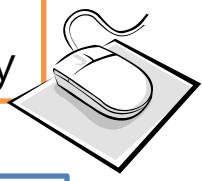
Gildong과의 총 통화 시간 구하기

Brower 콘텐츠 제공자

```
public class BrowserTestActivity extends Activity
{
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        String[] requestedColumns = {
            Browser.BookmarkColumns.TITLE,
            Browser.BookmarkColumns.VISITS
        };

        try {
            Cursor cursor = managedQuery(Browser.BOOKMARKS_URI,
                requestedColumns,
                Browser.BookmarkColumns.BOOKMARK + "=1", null,
                Browser.BookmarkColumns.VISITS + " DESC limit 5");

            Log.d("MyLog", "Bookmarks count: " + cursor.getCount());

            int titleIdx = cursor.getColumnIndex(Browser.BookmarkColumns.TITLE);
            int visitsIdx = cursor.getColumnIndex(Browser.BookmarkColumns.VISITS);
            cursor.moveToFirst();

            while (!cursor.isAfterLast()) {
                Log.d("MyLog", cursor.getString(titleIdx) + " visited " +
                   cursor.getInt(visitsIdx) + " times");

                cursor.moveToNext();
            }
        }
        catch (Exception e) {
            Log.d("MyLog", e.toString());
        }
    }
}
```

- 사용자의 웹 항해 내력(히스토리)과 북마크 정보를 제공
- 북마크된 사이트들 중 자주 방문한 다섯 곳을 얻는 예

CPTest
BrowserTestActivity

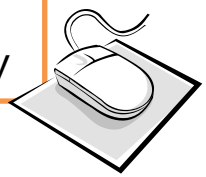


```
<uses-permission xmlns:android="http://schemas.android.com/apk/res/android"
android:name="com.android.browser.permission.READ_HISTORY_BOOKMARKS">
</uses-permission>
<uses-permission xmlns:android="http://schemas.android.com/apk/res/android"
android:name="com.android.browser.permission.WRITE_HISTORY_BOOKMARKS">
</uses-permission>
```

Contacts 콘텐츠 제공자

- 연락처 자료를 제공
- 이 콘텐츠 제공자를 사용하기 위하여 매니페스트 파일에 READ_CONTACTS 권한에 대한 <usespermission> 요소를 추가해야 함
- 연락처에서 이름과 전화번호를 알아내는 예 (다음 장)

CPTest
ContactsTestActivity

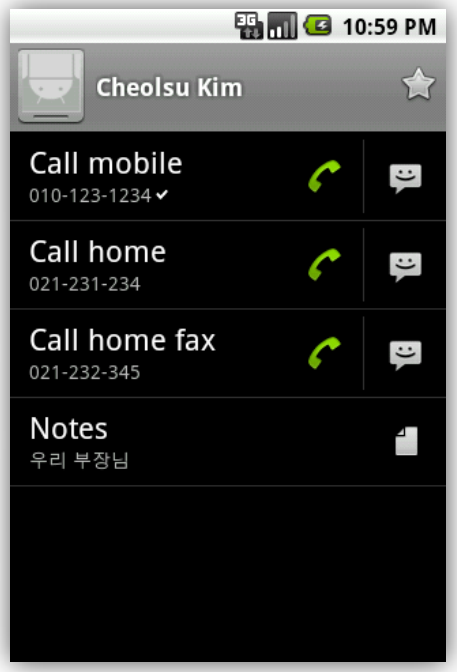


Contacts 콘텐츠 제공자

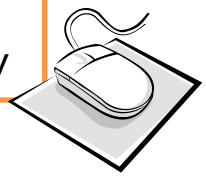
```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    //ContentResolver cr = getContentResolver();
    //Cursor cur = cr.query(ContactsContract.Contacts.CONTENT_URI, null, null, null, null);

    Cursor cur = managedQuery(ContactsContract.Contacts.CONTENT_URI, null, null, null, null);

    if (cur.getCount() > 0) {
        while (cur.moveToNext()) {
            //_ID, DISPLAY_NAME, 여러 전화번호
            String id = cur.getString(cur.getColumnIndex(ContactsContract.Contacts._ID));
            String name = cur.getString(cur.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME));

            int index = cur.getColumnIndex(ContactsContract.Contacts.HAS_PHONE_NUMBER);
            String phone = cur.getString(index);

            //전화번호가 있으면
            if (Integer.parseInt(phone) > 0) {
                Cursor pCur = managedQuery(ContactsContract.CommonDataKinds.Phone.CONTENT_URI, null,
                ContactsContract.CommonDataKinds.Phone.CONTACT_ID +" = ?", new String[]{id}, null);

                while (pCur.moveToNext()) {
                    String number = pCur.getString(pCur.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER));
                    Log.d("MyLog", "Name : " + name + "₩tNumber : " + number);
                }
                //pCur.close();
            }
        }
    }
    cur.close();
}
```

내장 콘텐츠 제공자 자료 수정

- 콘텐트 제공자에 자료 추가, 생성, 삭제도 가능
- 이 경우 READ_CONTACTS가 아니라 WRITE_CONTACTS 등과 같은 쓰기 권한이 있어야 함
- 연락처 추가 (Contacts 콘텐츠 제공자)



CPTest
UpdateTestActivity



```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    //우선 빈 레코드 하나를 해당 콘텐츠 접근 URI에 삽입함. 그러면 새로 생성된 row를 접근할 수 있는 URI가 반환됨
    ContentValues values = new ContentValues();
    Uri rawContactUri = getContentResolver().insert(ContactsContract.RawContacts.CONTENT_URI, values);

    values.clear();
    //ID 입력
    long rawContactId = ContentUris.parseId(rawContactUri);
    values.put(ContactsContract.Data.RAW_CONTACT_ID, rawContactId);
    //타입 입력
    values.put(ContactsContract.Data.MIMETYPE,ContactsContract.CommonDataKinds.StructuredName.CONTENT_ITEM_TYPE);
    //표시되는 이름 입력
    values.put(ContactsContract.CommonDataKinds.StructuredName.DISPLAY_NAME, "Cheolsu Kim");
    //저장함
    getContentResolver().insert(ContactsContract.Data.CONTENT_URI, values);

    Uri mobileUri = null;
    Uri homeUri = null;
    Uri emailUri = null;
    Uri workUri = null;
    Uri faxUri = null;
    Uri noteUri = null;

    //모바일 전화번호를 저장할 URI 알아냄
    mobileUri = Uri.withAppendedPath(rawContactUri, ContactsContract.Contacts.Data.CONTENT_DIRECTORY);
    values.clear();
    values.put(ContactsContract.CommonDataKinds.Phone.TYPE, ContactsContract.CommonDataKinds.Phone.TYPE_MOBILE);
    values.put(ContactsContract.CommonDataKinds.Phone.IS_SUPER_PRIMARY, 1);
    values.put(ContactsContract.Data.MIMETYPE, ContactsContract.CommonDataKinds.Phone.CONTENT_ITEM_TYPE);
    values.put(ContactsContract.CommonDataKinds.Phone.NUMBER, "010-123-1234");
    //거기에 전화번호를 저장함
    getContentResolver().insert(mobileUri, values);

    //집 전화번호를 저장할 URI 알아냄
    homeUri = Uri.withAppendedPath(rawContactUri, ContactsContract.Contacts.Data.CONTENT_DIRECTORY);
    values.clear();
    values.put(ContactsContract.CommonDataKinds.Phone.TYPE,
    ContactsContract.CommonDataKinds.Phone.TYPE_HOME);
    values.put(ContactsContract.Data.MIMETYPE,
    ContactsContract.CommonDataKinds.Phone.CONTENT_ITEM_TYPE);
    values.put(ContactsContract.CommonDataKinds.Phone.NUMBER, "02-123-1234");
    //거기에 전화번호를 저장함
    getContentResolver().insert(homeUri, values);
```

04. 콘텐츠 제공자를 통한 자료 공유(15)

```
//직장 전화번호를 저장할 URI 알아냄
workUri = Uri.withAppendedPath(rawContactUri,
ContactsContract.Contacts.Data.CONTENT_DIRECTORY);
values.clear();
values.put(ContactsContract.CommonDataKinds.Phone.TYPE,
ContactsContract.CommonDataKinds.Phone.TYPE_WORK);
values.put(ContactsContract.Data.MIMETYPE,
ContactsContract.CommonDataKinds.Phone.CONTENT_ITEM_TYPE);
values.put(ContactsContract.CommonDataKinds.Phone.NUMBER, "02-123-1234");
//거기에 전화번호를 저장함
getContentResolver().insert(workUri, values);

faxUri = Uri.withAppendedPath(rawContactUri,
ContactsContract.Contacts.Data.CONTENT_DIRECTORY);
values.clear();
values.put(ContactsContract.CommonDataKinds.Phone.TYPE,
ContactsContract.CommonDataKinds.Phone.TYPE_FAX_HOME);
values.put(ContactsContract.Data.MIMETYPE,
ContactsContract.CommonDataKinds.Phone.CONTENT_ITEM_TYPE);
values.put(ContactsContract.CommonDataKinds.Phone.NUMBER, "02-123-2345");
getContentResolver().insert(faxUri, values);

emailUri = Uri.withAppendedPath(rawContactUri,
ContactsContract.Contacts.Data.CONTENT_DIRECTORY);
values.clear();
values.put(ContactsContract.CommonDataKinds.Email.TYPE,
ContactsContract.CommonDataKinds.Email.TYPE_OTHER);
values.put(ContactsContract.Data.MIMETYPE,
ContactsContract.CommonDataKinds.Email.CONTENT_ITEM_TYPE);
values.put(ContactsContract.CommonDataKinds.Email.DISPLAY_NAME, "SUNNY");
getContentResolver().insert(emailUri, values);

noteUri = Uri.withAppendedPath(rawContactUri,
ContactsContract.Contacts.Data.CONTENT_DIRECTORY);
values.clear();
values.put(ContactsContract.Data.MIMETYPE,
ContactsContract.CommonDataKinds.Note.CONTENT_ITEM_TYPE);
values.put(ContactsContract.CommonDataKinds.Note.NOTE, "우리 부장님");
getContentResolver().insert(noteUri, values);
```