● 콘텐츠 프로바이더(ContentPRovider): 앱 간의 데이터 공유를 목적으로 사용하는 컴포넌트. 데이터를 외부 앱에 공유하거나, 반대로 외부 앱이 가지고 있는 데이터를 획득할 목적으로 사용된다. 스마트폰에 기본으로 설치된 앱들이 중요한 데이터를 많이 가지고 있기 때문에 이 기본 앱(주소록, 갤러리 등)과의 데이터 공유가 중요하기에 콘텐츠 프로바이더는 중요하다. 콘텐츠 프로바이더를 이용하면 파일, 데이터베이스, Preference처럼 외부 앱의 내장 메모리 공간에 저장된 데이터에 접근할 수 있다. 데이터를 가지고 있는 앱이 자신의 데이터를 일정 정도 외부 앱에 오픈할 의사가 있다면 개발자가 해당 앱에 콘텐츠 프로바이더를 만들어야 한다. 그리고 외부 앱은 해당 앱에서 만들어준 콘텐츠 프로바이더의 함수를 이용하여 데이터를 이용하는 구조이다. 이렇게 콘테츠 프로바이더의 함수를 이용하는 구조이므로 개발자 알고리즘으로 얼마든지 오픈되는 데이터를 제어할 수 있다. 보안상 문제가 되는 데이터를 제외하고 오픈하는 등의 작업이 가능하다.

▶ 콘텐츠 프로바이더 작성법: 콘텐츠 프로바이더 내부에서 접근하는 데이터는 파일, 데이터베이스, Preference 혹은 메모리의 데이터일 수도 있다. 콘텐츠 프로바이더는 ContentProvider 클래스를 상속받아 작성하며, 여섯 개의 함수 모두 재정의해야 한다. 콘텐츠 프로바이더의 생명주기 함수는 onCreate() 함수 하나만 있으며 최초에 한 번만 호출된다. 그리고 나머지 query(), insert(), update(), delete() 함수는 외부 앱에서 필요할 때 호출된다. 이곳에서 적절한 데이터 획득, 저장, 수정, 삭제 작업을 하면 된다.

```
public class MyContentProvider extends ContentProvider {
    public MyContentProvider() {
    }

    @Override
    public int delete(Uri uri, String selection, String[] selectionArgs) {
    return 0;
    }

    @Override
    public String getType(Uri uri) {
        throw new UnsupportedOperationException("Not yet implemented");
    }

    @Overrie
    public Uri insert(Uri uri, ContentValues values) {
        return null;
    }

    @Override
    public boolean onCreate() {
        return false;
    }

    @Override
    public Cursor query(Uri uri, String[] projection, String selection,
                        String[] selectionArgs, String sortOrder) {
        retrun null;
    }

    @Override
    public int update(Uri uri, ContentValues values, String selection, String[] selectic
        return 0;
    }
}
```

만약 외부 앱의 요청으로 인해 자신의 데이터가 추가되거나 삭제되는 걸 막고 싶다면, 즉, 그저 데이터를 외부 앱에 전달하는 용도로만 쓰고 싶다면, insert(), update(), delete() 함수 부분을 비워두면 된다. 참고로 구글 기본 앱의 콘텐츠 프로바이더들은 대부분 insert(), update(), delete() 함수에도 일정 정도의 작업을 해준다.

콘텐츠 프로바이더도 안드로이드 컴포넌트이므로 AndroidManifest.xml에 등록해서 사용해야 한다. 액티비티나 서비스, 그리고 브로드캐스트 리시버는 AndroidManifest.xml 파일에 각각 <activity>, <service>, <receiver> 태그로 등록할 때 생략할 수 없는 속성이 name뿐이지만, 콘텐츠 프로바이더는 <provider> 태그로 등록하는데 name 이외에 authorities 속성을 꼭 정의해 주어야 한다. authorities 속성값은 개발자 임의의 문자열이지만 유일해야 한다. 다른 앱의 콘텐츠 프로바이더와 같은 값이 대입되면 앱 자체가 사용자 스마트폰에 설치되지 않는다. authorities 값은 스마트폰 전체에서 콘텐츠 프로바이더를 구분하기 위한 식별자가 되는 것이다.

```
<provider
    android:name=".MyContentProvider"
    android:authorities="com.example.test.Provider"
    android:enabled="true"
    android:exported="true">
</provider>
```

▶ 콘텐츠 프로바이더 이용: 콘텐츠 프로바이더는 안드로이드 컴포넌트 중에 유일하게 인텐트로 실행하지 않는 컴포넌트인데, 이는 콘텐츠 프로바이더의 독특한 생명주기 때문이다. 액티비티, 서비스, 브로드캐스트 리시버는 부팅 완료 시점부터 어느 앱에서 어떤 이름의 컴포넌트를 가지고 있다는 정보가 시스템에 등록되지만, 실제 객체가 생성되는 시점은 어디선가 그 컴포넌트를 이용하기 위해 인텐트를 발생시키는 순간이다. 하지만 콘텐츠 프로바이더는 시스템에서 인지하는 순간 이용하지 않더라도 미리 생성해 놓는다. 따라서 스마트폰이 부팅되면 여러 앱의 모든 콘텐츠 프로바이더가 생성된다.

외부 앱의 콘텐츠 프로바이더를 이용하는 앱에서는 이렇게 시스템에서 미리 생성해놓은 콘텐츠 프로바이더 객체를 ContentResolver 클래스를 이용해 획득하여 사용한다. ContentResolver는 콘텐츠 프로바이더로 생성된 객체들을 담고 있는 관리자 역할의 클래스이다. ContentResolver에 등록된 모든 앱의 모든 콘텐츠 프로바이더 중 내가 이용하고자 하는 객체를 식별하기 위해 이용되는 것이 Uri 객체이다. 그래서 콘텐츠 프로바이더를 흔히 "Uri 모델로 식별되어 이용되는 컴포넌트"라고 표현한다. Uri 객체를 적절한 URL로 만들어 주어야 하는데, 콘텐츠 프로바이더를 식별하기 위해 사용되는 URL은 규칙이 있다.

```
content:// com.example.test.Provider
```

콘텐츠 프로바이더를 식별하기 위한 URL의 프로토콜명(scheme)은 content를 이용한다. 그리고 host 부분(도메인)이 이용하고자 하는 콘텐츠 프로바이더의 식별자인데, 이는 이용하고자 하는 콘텐츠 프로바이더가 자신의 AndroidManifest.xml 파일에 등록될 때 <provider> 태그의 authorities 속성값으로 작성된 문자열이다.

```
// URL 구조
content:// com.example.test.Provider/user/1
// Scheme               host          path
```

콘텐츠 프로바이더를 이용하기 위한 식별자 URL 문자열이 이렇게 설정되어 있는 경우, 각 부분은 Scheme, host, path로 나뉜다. Scheme은 "content" 단어로 고정되고, host 부분은 콘텐츠 프로바이더의 식별자이다. Scheme, host까지가 URL을 구성해야 할 때 꼭 들어가야 하는 저옵이고, 이 정보가 맞아야 이용하고자 하는 콘텐츠 프로바이더를 식별하여 사용할 수 있다. 그런데 host 하위의 path 정보는 필수 요소가 아니며, 콘텐츠 프로바이더를 이용하는 곳에서 임의로 추가할 수 있는 정보이다. path의 의미는 그 단어로 표현되는 모든 데이터를 지칭하게 되며, path의 끝 부분이 숫자로 끝나면 id 값을 지칭하여 해당 id 값의 데이터를 지칭한다.

```
content:// com.example.test.Provider/user
// content://com.example.test.Provide로 식별되는 콘텐츠 프로바이더의 모든 user 데이터를 지칭

content://com.example.test.Provider/user/1
// content://com.example.test.Provider로 식별되는 콘텐츠 프로바이더의 user 데이터 중
// id 값이 1인 데이터를 지칭하는 의미이다.
```

path 부분은 Provider에 전달된다. 물론 이용하는 곳에서 path 값을 설정하였더라도 콘텐츠 프로바이더를 만든 곳에서 path 값을 무시하면(프로바이더를 구현하면서 무시했다면) 의미는 없다. 하지만 path로 일정 데이터의 where 조건을 명시할 수 있지 않느냐는 의도이고, 그런 의미에서 path 값이 설정되면 데이터의 조건으로 사용할 것을 권장하고 있다. (ex. 안드로이드 기본 앱들은 대부분 path 값에 반응해서, 예를 들어 주소록의 provider을 연동했을 때 query함수에 조건을 주지 않아도 path (uri)에 조건이 있으면 해당 데이터를 돌려준다).

ContentResolver을 이용해서 적절한 Uri 객체로 콘텐츠 프로바이더를 식별할 수만 있다면, 이후 콘텐츠 프로바이더를 이용해 데이터를 획득하거나 추가하는 등의 작업은 DBMS 프로그램과 유사하다. 콘텐츠 프로바이더의 query(), insert(), update(), delete() 함수를 적절하게 호출해 주면 된다.

```
getContentResolver().query(Uri.parse("content://com.example.provider"), null, null, null
```

getContentResolver() 함수로 ContentResolver 객체를 획득하고 (안드로이드에 깔려 있는 모든 앱의 모든 콘텐츠 프로바이더의 객체가 담겨 있는 객체) 사용할 함수 — query, insert, update나 delete 함수에 첫 번째 매개변수로 Uri 값을 주어 어느 콘텐츠 프로바이더를 이용할 것인지 판단하게 된다. 그리고 나머지 매개변수는 데이터 획득을 위한 조건이다.

- query(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder)
- insert(Uri url, ContentValues values)
- update(Uri url, ContentValues values, String where, String[] selectionArgs)
- delete(Uri url, String where, String[] selectionArgs)

Resources: 깡쌤의 안드로이드 프로그래밍