안드로이드 앱은 화면을 출력하는 '액티비티', 백그라운드 작업을 위한 '서비스', 앱 간의 데이터 공유를 돕는 '콘텐츠 프로바이더', 그리고 이벤트 모델로 수행되는 '브로드캐스트 리시버', 이렇게 네 개의 컴포넌트로 구성된다. 이 컴포넌트들은 개발자가 만드는 자바 클래스이지만 이 클래스들은 개발자가 작성하는 자바 코드로 서로 결합하여 수행되지 않고 안드로이드 시스템이 한 클래스에서 의뢰를 받아 다든 클래스를 실행하는 구조, 즉, 생명주기를 시스템이 관리하므로 컴포넌트라고 부든다.

● 인텐트의 기본 개념: 인텐트는 컴포넌트를 실행하기 위해 시스템에 넘기는 정보이다. 실행하고자 하는 컴포넌트의 정보를 담은 인텐트를 구성해서 시스템에 넘기면, 시스템에서 인텐트 정보를 분석해서 맞는 컴포넌트를 실행해주는 구조이다. 즉, 컴포넌트 간의 실행 시 두 클래스가 직접 결합하지 않고 인텐트를 매개로 해서 실행되므로 같은 앱 내의 컴포넌트를 실행하든, 다든 앱의 컴포넌트를 실행하든 개발자는 인텐트를 시스템에 의뢰하기만 하면 된다. 같은 코드로 같은 앱이나 외부 앱의 컴포넌트를 실행할 수 있는 것이다. 인텐트에 의해 다든 컴포넌트를 실행할 때, 인텐트에 어떤 정보를 담는지에 따라 크게 명시적 인텐트와 암시적 인텐트로 구분된다.

▶ 명시적 인텐트: 실행하고자 하는 컴포넌트의 클래스명을 인텐트 생성자의 매개변수에 넣는 방법으로, 주로 같은 앱의 컴포넌트를 실행할 때 이용한다. 시스템에서는 인텐트에 있는 클래스명을 참조해서 해당 클래스를 실행한다. 액티비티를 실행하기 위해 인텐트를 시스템에 의뢰하는 함수는 startActivity() 함수로, 이 함수의 매개변수로 실행하고자 하는 컴포넌트의 정보를 담은 Intent 객체를 주면 된다.

```
// Ex. 주소록 앱의 목록 액티비티의 클래스 명이 ListActivity, 상세보기는 DetailActivity
Intent intent = new Intent(this, DetailActivity.this);
startActivity(intent);
```

▶ 암시적 인텐트: 실행하고자 하는 컴포넌트의 클래스 명이 아닌 Intent Filter 정보를 인텐트 생성자의 매개변수에 넣는 방법으로, 주로 클래스명을 알 수 없는 외부 앱의 컴포넌트를 실행할 때 이용된다. 시스템은 Intent Filter 정보를 해석해서 컴포넌트를 실행한다. Intent Filter란 AndroidManifest.xml 파일에 등록된 컴포넌트의 정보이다. setAction() 함수를 통해 실행하려는 컴포넌트의 AndroidManifest.mxl 파일에 정의된 Intent Filter의 Action 정보를 주는 것은, Action 문자열이 해당 정보로 등록된 액티비티를 실행해 달라는 의미이다. 즉, 실행하려는 컴포넌트의 AndroidManifest.xml 파일에 정의된 intent-filter 정보와 동일한 갑을 주어서 실행하는 것이 암시적 인텐트이다.

```
// AndroidManifest.xml
<activity
    android:name=".SomeActivity"
    android:exported="true" >
    <intent-filter>
        <action android:name="com.some.ACTION_VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <category android:name="com.some.cateogry.MYCATEGORY"/>
        <data android:scheme="geo"/>
</activity>
```

```
Intent intent = new Intent();
intent.setAction("com.example.ACTION_VIEW");
startActivity(intent);
```

▶ 인텐트 필터(intent-filter): AndroidManifest.xml에 컴포넌트 하위 태그로 <intent-filter> 태그가 등록된다. <action>, <category>, <data>가 등록될 수 있는데, 대부분 <action>만 등록하거나 <action>과 <data> 등록을 많이 한다.

action	컴포넌트가 어떤 능력을 갖추고 있는지에 대한 문자열. 개발자가 임의로 지정하는 단어도 가능하며, 라이브러리에서 지정한 문자열을 이용해도 된다.	android.intent.action.VIEW: 컴포넌트가 데이터를 보여주는 능력을 갖추고 있다고 선언한 것이다. android.intent.action.EDIT: 이 컴포넌트가 데이터를 편집하는 능력을 갖추고 있다는 의미로 선언한 것이다.
category	컴포넌트에 대한 추가 정보로 어느 범주의 컴포넌트를 표현하는데 사용되는 문자열. 개발자가 임의로 지정하기도 하지만, 대부분 라이브러리 내에서 준비된 단어를 사용한다. (몇 개 라이브러리 내에서 범용으로 선언한 CATEGORY_BROWSABLE, CATEGORY_LAUNCHER) 이외에는 개발자가 지정한 이름으로 추가해서 사용해야 할 일은 많지 않다.	android.intent.cateogry.LAUNCHER: 컴포넌트가 런처에 의해 처리될 범주에 포함하겠다는 의미이다. android.intent.cateogry.BROWSABLE: 컴포넌트가 브라우저에 의해 처리될 범주에 포함하겠다는 의미이다.
data	컴포넌트를 실행하기 위해 필요한 데이터에 대한 상세 정보를 명시하기 위해 사용된다. URL 형식으로 표현되어 android:scheme, android:host, android:port, android:mimeType 등으로 선언하게 된다.	

선언된 액티비티를 코드로 실행할 때, intent-filter의 action 정보만 주고 별도의 category 정보를 주지 않아도, startActivity() 함수가 인텐트를 발생시킬 때 자동으로 android.intent.category.DEFAULT를 추가하게 되어 있어서 실행이 가능하다. 또한 개발자가 지정한 문자열의 category가 등록되더라도 라이브러리 내에서 준비된 단어를 사용한 category 문자열 (ex. android.intent.category.DEFAULT)는 꼭 선언해 주어야 한다.

```
// category 정보를 추가하지는 않았지만 startActivity() 함수가 android.intent.cateogry.DEFUA
// 자동으로 추가하여 인텐트에는 action 문자열과 category 정보가 하나씩 담기게 된다.
// 인텐트에 담긴 android.intent.category.DEFAULT 이름의 category가 intent-filter에 선언되어
// 있으므로 해당 컴포넌트가 실행이 된다.
Intent intent = new Intent();
Intent.setAction("com.some.ACTION_VIEW");
startActivity(intent);
```

```
// addCategory() 함수를 이용해서 개발자가 지정한 문자의 category(com.some.category.MTCATEGO
// 를 등록하면 인텐트에는 android.intent.category.DEFAULT와 com.some.category.MYCATEGORY 0
// 두 개의 cateogry 정보와 action 정보가 담기게 되고, 이 두 개가 모두 <intent-filter>에
// 선언되어 있으므로 실행이 된다.
Intent intent = new Intent();
Intent.setAction("com.some.ACTION_VIEW");
Intent.addCategory("com.some.category.MYCATEGORY");
startActivity(intent);
```

data 정보에는 scheme 이외에 host, port, path가 함께 선언될 수 있다. data 정보는 URL 문자열로 나타내며, Uri 객체로 표현된다. intent-filter의 data 부분에 scheme이라고 선언되어 있는 것은, URL의 프로토콜을 지칭한다. 그래서 코드에서 데이터 값을 줄 때 이 프로토콜 명을 맞추어 주어야 한다.

```
Intent intent = new Intent();
intent.setAction("com.some.ACTION_VIEW");
intent.setData(Uri.parse("geo:"));
startActivity(intent);
```

→ 인텐트가 반응할 액티비티가 없을 때가 있을 수 있으며, 만약 시스템에 반응할 액티비티가 없으면 액티비티 인텐트에 에러가 발생한다. 만약 인텐트에 반응할 액티비티가 여러 개 있으면, 그중 사용자가 선택한 하나만 실행된다. 시스템에서 자동으로 띄우는 화면에서 사용자가 선택한 액티비티가 실행되는 것이다.

▶ Extra 데이터: 인텐트를 발생시키는 엑티비티에 있는 데이터를 인텐트에 의해서 실행되는 컴포넌트에 전달하면서 실행해야 한다면, 인텐트에 데이터를 담아 전달하는 방식을 이용하는데, 이 데이터를 흔히 Extra 데이터라고 부른다.

```
// Ex.ListActivity가 가지고 있는 데이터를 DetailActivity에 넘기면서 실행해야 하는 경우
Intent intent = new Intent(this, DetailActivity.class);
intent.putExtra("data1", "hello");
intent.putExtra("data2", 100);
startActivity(intent);
```

인텐트가 발생하기 전에 putExtra() 함수를 이용하여 데이터를 인텐트 객체에 담아주면 되는데, 이 데이터는 key, value 성격으로 담아주면 되고, 문자열, 숫자, boolean, 객체 등 모든 타입의 데이터를 넘길 수 있다.

```
// DetailActivity

Intent intent = getIntent();
String data1 = intent.getStringExtra("data1");
int data2 = intent.getIntExtra("data2", 0);
// data2에 해당되는 값(value) 반환, 없으면 default 값 — 여기서는 0 반환
```

컴포넌트에서는 자신을 실행했던 인텐트 객체를 getIntent() 함수로 얻어서, 그 인텐트 객체 안에 담긴 Extra 데이터를 getXXXExtra() 함수를 이용하여 얻는 구조이다.

→ 실행되는 컴포넌트의 함수를 호출해서 매개변수로 데이터를 직접 넘기는 대신 인텐트의 Extra 데이터를 이용해야 하는 이유는 컴포넌트 클래스를 우리가 생성하는 것이 아니라 생명주기 관리를 시스템이 하므로 우리가 생성하지 않은 객체의 함수를 호출할 수 없기 때문이다.

● 결과 되돌리기: 액티비티는 화면 출력을 주목적으로 하는 컴포넌트이기 때문에 인텐트가 발생하면 화면이 전환된다. 이때 사용자가 뒤로가기 버튼을 누르면 화면이 되돌아오겠지만, 그러지 않고도 (그 컴포넌트에서 어떤 데이터를 얻은 다음) 자바 코드에서 자동으로 화면이 되돌아오게 처리해야 할 때가 있다. 예를 들어 채팅 앱에서 카메라 촬영 버튼을 눌렀을 때 카메라 앱의 촬영 액티비티가 인텐트로 실행되는데, 사용자가 사진 촬영 후에 자동으로 이전 화면으로 되돌아와 찍은 사진을 채팅방에 자동으로 출력해 주어야 하는 경우이다. 화면을 자동으로 되돌려 결과를 받기 위해서 startActivityForResult() 함수 또는 ActivityResultLauncher을 이용한다. Android 11 버전 이후로는 ActivityResultLauncher을 사용하는 것을 권장하고 있다.

▶ startActivityForResult() 함수

```
// ListActivity

Intent intent = new Intent(this, DetailActivity.class);
startActivityForResult(intent, 10);
```

두 번째 매개변수로 requestCode 값을 넘겨 주는데, 이 값은 개발자가 0 이상의 숫자를 지정하여 결과를 되돌려받았을 때 어느 요청이 되돌아온 것인지를 구분하기 위해 사용된다.

```
// DetailActivity

Intent intent = getIntent();
intent.putExtra("location", textView.getText().toString());
setResult(RESULT_OK, intent);
finish();
```

startActivityForResult() 함수로 수행된 액티비티에서 결과를 되돌리고 싶으면 자신의 액티비티를 종료하면 된다. 종료하기 전에 인텐트 객체에 putExtra() 함수를 통해 Extra 데이터로 결과를 담을 수 있으며, 자신의 상태를 setResult() 함수를 이용해 지정할 수 있다(resultCode). 여기서는 RESULT_OK로 지정하여 정상으로 처리되어 되돌린 것임을 명시한 예이다 (또 다른 예: RESULT_CANCELED, RESULT_FIRST_USER). 그리고 finish() 함수를 호출하여 자신을 종료한다.

```
// ListActivity

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == 10 && resultCode == RESULT_OK) {
    // 10번 요청이 되돌아왔다면, 그리고 요청 처리가 잘 되었다면 사후처리
    }
}
```

액티비티가 종료되면 이전 화면으로 되돌아가며, 이전 액티비티의 onActivityResult() 함수가 자동으로 호출된다. onActivityResult() 함수의 첫 번째 매개변수인 requestCode는 인텐트를 발생시킨 곳에서 지정한 값이고, 두 번째 매개 변수인 resultCode는 호출되엇던 곳에서 되돌리기 전에 지정한 값이다.

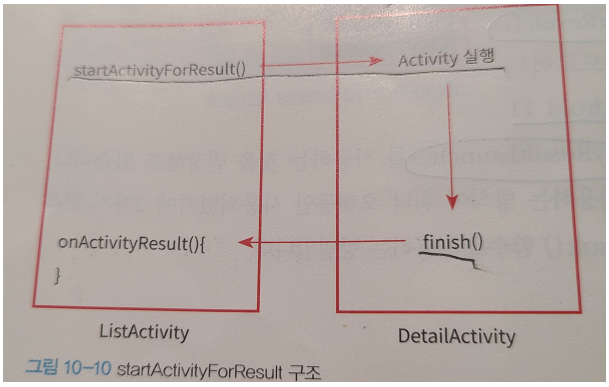

그림 10-10 startActivityForResult 구조

▶ ActivityResultLauncher: androidx에서 제공한다. 인텐트에 의한 결과를 되돌려 받을 때뿐만 아니라, 액티비티에서 무언가를 실행하여 결과를 획득해 무언가를 처리할 때도 사용된다. 또한, 퍼미션을 요청할 때도 이용한다. ActivityResultLauncher에 Contract와 CallBack 객체를 등록한 후 launch() 함수로 실행하는 구조이다. 그러면 ActivityResultLaucher에 등록된 Contract 객체가 실행 요청을 처리하여 그 결과를 CallBack에 전달해 사후 처리가 되도록 하는 구조이다. (액티비티 화면을 벗어났다가 되돌아와서 사후 처리하는 것, 그게 다든 엑티비티 실행이든지 퍼미션 요청이든지 간에 같은 코드로 이용하기 위해서 만든 것이다).

Contract (요청 처리자)는 ActivityResultContract을 상속받은 서브 클래스이며, 직접 만들거나 API에서 제공되는 클래스를 이용한다. 다음은 API에서 제공하는 Contract 클래스들이다.

PickContract	선택한 연락처의 Uri 획득
RequestPermission	권한 요청, 허락 여부 파악
RequestMultiplePermissions	여러 권한을 동시에 요청
StartActivityForResult	인텐트 발생으로 액티비티 실행, 결과 획득
TakePicturePreview	사진 촬영 후 비트맵 획득
TakePicture	사진 촬영, 저장, 비트맵 획득

ActivityResultLauncher로 액티비티를 실행시키고 결과를 되돌려 받아야 할 때는 StartActivityForResult를 이용한다. 또한 퍼미션 조정을 요청하고 그 결과를 판단할 때는 RequestPermission을 사용한다.

ActivityResultLauncher 객체는 registerForActivityResult() 함수로 만들어지며, 함수의 매개변수에 작업자인 Contract 객체와 결과를 처리할 Callback 객체를 등록한다.

```
ActivityResultLauncher<Intent> resultLauncher = registerForActivityResult(
    // 이 laucher가지고 처리하는 게 인텐트 발생 - 요청 처리자
    new ActivityResultContracts.StartActivityForResult(),
    // 실행된 엑티비티에서 담은 결과값이 result로 들어온다
    new ActivityResultCallback<ActivityResult>() {
        @Override
        public void onActivityResult(ActivityResult result) {
            Intent intent = result.getData();
            binding.resultTextView.setText(intent.getStringExtra("result"));
        }
    }
);
```

Callback 객체는 ActivityResultCallBack을 구현한 객체이며, 결과를 처리할 때 onActivityResult() 함수가 자동으로 호출된다. 이렇게 만들어진 ActivityResultLauncher 객채의 launch 함수의 매개 변수를 호출하면서 실행하고자 하는 Intent 객체로 지정해 주면 된다.

```
// 인텐트를 발생시키고
Intent intent = new Intent(this, DetailActivity.class);
// 돌려 받은 결과값을 laucher에서 처리한다
resultLauncher.launch(intent);
```

```
// MainActivity.java

package com.android.practiceintentactivityresultlauncher;

import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.ViewGroup;
import android.widget.Toast;
```

```java
import androidx.activity.result.ActivityResult;
import androidx.activity.result.ActivityResultCallback;
import androidx.activity.result.ActivityResultLauncher;
import androidx.activity.result.contract.ActivityResultContract;
import androidx.activity.result.contract.ActivityResultContracts;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.DividerItemDecoration;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.android.practiceintentactivityresultlauncher.databinding.ActivityMainBinding;
import com.android.practiceintentactivityresultlauncher.databinding.ItemMainBinding;

import java.util.ArrayList;
import java.util.List;

public class MainActivity extends AppCompatActivity {
    ArrayList<String> datas;
    ActivityResultLauncher<Intent> resultLauncher;
    String category;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ActivityMainBinding binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        // MainActivity에서 화면에 항목을 출력하고, 출력된 항목 중 하나를 사용자가 클릭했을 때
        // 인텐트로 실행한다. 실행 시 항목 정보를 DetailActivity에 Extra 데이터로 넘겨주며, 결
        // ActivityResultLauncher로 인텐트를 발생시켜 준다.

        resultLauncher = registerForActivityResult(
                new ActivityResultContracts.StartActivityForResult(),
                new ActivityResultCallback<ActivityResult>() {
                    @Override
                    public void onActivityResult(ActivityResult result) {
                        Intent intent = result.getData();
                        String txt = category + " " + intent.getStringExtra("result");
                        Toast.makeText(MainActivity.this, txt, Toast.LENGTH_SHORT).show(
                    }
                }
        );

        // 데이터는 DBHelper.java에서 가상 데이터로 데이터베이스에 삽입되었다
        DBHelper helper=new DBHelper(this);
        SQLiteDatabase db=helper.getWritableDatabase();
        Cursor cursor=db.rawQuery("select location from tb_data where category='0'", nul

        // ArrayList datas에 데이터베이스의 첫번째 콜룸 값들 담기
        datas = new ArrayList<>();
        while (cursor.moveToNext()){
            datas.add(cursor.getString(0));
        }
        db.close();

        // RecyclerView 설정
        binding.mainRecyclerView.setLayoutManager(new LinearLayoutManager(this));
        binding.mainRecyclerView.addItemDecoration(new DividerItemDecoration(this, new L
        binding.mainRecyclerView.setAdapter(new MyMainAdapter(datas));
```

```
    }

    // RecyclerView의 필수 구성 요소에는 ViewHolder(각 항목 구성 뷰), Adaptor (항목 구성), L


    private class MyMainAdapter extends RecyclerView.Adapter<MyMainViewHolder> {
        private List<String> list;

        public MyMainAdapter(List<String> list) {
            this.list = list;
        }

        // ViewHolder 객체 반환
        @Override
        public MyMainViewHolder onCreateViewHolder(ViewGroup viewGroup, int i) {
            ItemMainBinding binding = ItemMainBinding.inflate(LayoutInflater.from(viewGr
            return new MyMainViewHolder(binding);
        }

        // 각 항목을 구성하기 위해서 호출
        @Override
        public void onBindViewHolder(MyMainViewHolder viewHolder, int position) {
            String text = list.get(position);
            // 항목 뷰의 text을 설정
            viewHolder.binding.itemMainView.setText(text);
        }

        @Override
        public int getItemCount() {
            return list.size();
        }
    }

    // ViewHolder의 역할은 항목을 구성하기 위한 뷰들을 findViewById 해주는 역할을 한다.
    private class MyMainViewHolder extends RecyclerView.ViewHolder {
        // 항목을 구성하기 위한 뷰 (TextView)
        ItemMainBinding binding;

        // 생성자
        public MyMainViewHolder(ItemMainBinding binding) {
            super(binding.getRoot());
            this.binding = binding;

            // 항목의 뷰를 사용자가 클릭했을 때 이벤트: DetailActivity를 인텐트로 실행한다.
            // 실행 시 항목 정보를 DetailActivity에 Extra 데이터로 넘겨준다.
            binding.getRoot().setOnClickListener(view -> {
                // 뷰의 텍스트로 지정된 데이터를 String으로 변환해 category에 저장
                category = binding.itemMainView.getText().toString();
                // 인텐트 생성
                Intent intent = new Intent(MainActivity.this, DetailActivity.class);
                // 항목 정보인 category를 Extra Data로 DetailActivity에 넘겨주기
                intent.putExtra("category", category);
                // ActivityResultLauncher로 인텐트를 발생시킨다.
                resultLauncher.launch(i
```

```
// DetailActivity.java
```

```java
package com.android.practiceintentactivityresultlauncher;

import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.ViewGroup;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.DividerItemDecoration;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.android.practiceintentactivityresultlauncher.databinding.ActivityDetailBindir
import com.android.practiceintentactivityresultlauncher.databinding.ActivityMainBinding;
import com.android.practiceintentactivityresultlauncher.databinding.ItemDetailBinding;

import java.util.ArrayList;
import java.util.List;

public class DetailActivity extends AppCompatActivity {
    ArrayList<String> datas;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ActivityDetailBinding binding = ActivityDetailBinding.inflate(getLayoutInflater(
        setContentView(binding.getRoot());

        // 자신의 실행했더 인텐트 객체 얻기
        Intent intent = getIntent();
        // 그 인텐트 객체에 담긴 데이터 얻기
        String category = intent.getStringExtra("category");

        DBHelper helper=new DBHelper(this);
        SQLiteDatabase db=helper.getWritableDatabase();
        Cursor cursor=db.rawQuery("select location from tb_data where category=?", new S

        datas = new ArrayList<>();
        while (cursor.moveToNext()){
            datas.add(cursor.getString(0));
        }
        db.close();

        binding.detailRecyclerView.setLayoutManager(new LinearLayoutManager(this));
        binding.detailRecyclerView.addItemDecoration(new DividerItemDecoration(this, new
        binding.detailRecyclerView.setAdapter(new MySubAdapter(datas));
    }

    private class MySubAdapter extends RecyclerView.Adapter<MySubViewHolder> {
        private List<String> list;

        public MySubAdapter(List<String> list) {
            this.list = list;
        }

        @Override
        public MySubViewHolder onCreateViewHolder(ViewGroup viewGroup, int i) {
```

```
            ItemDetailBinding binding = ItemDetailBinding.inflate(LayoutInflater.from(vi
            return new MySubViewHolder(binding);
        }

        @Override
        public void onBindViewHolder(MySubViewHolder viewHolder, int position) {
            String text = list.get(position);
            viewHolder.binding.itemDetailView.setText(text);
        }

        @Override
        public int getItemCount() {
            return list.size();
        }
    }

    private class MySubViewHolder extends RecyclerView.ViewHolder {
        ItemDetailBinding binding;

        public MySubViewHolder(ItemDetailBinding binding) {
            super(binding.getRoot());
            this.binding = binding;

            // 나열된 항목 중 하나가 선택되면 데이터를 포함해서 이전 화면으로 자동 전환 해주기
            binding.getRoot().setOnClickListener(view -> {
                Intent intent = getIntent();
                intent.putExtra("result", binding.itemDetailView.getText().toString());
                setResult(Activity.RESULT_OK, intent);
                finish();
            });

        }
    }

}
```

```
// activity_main.xml

<?xml version="1.0" encoding="utf-8"?>
<androidx.recyclerview.widget.RecyclerView xmlns:android="http://schemas.android.com/apk
    android:id="@+id/mainRecyclerView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

```
// item_main.xml

<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/itemMainView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="15sp"
    android:padding="8dp">
</TextView>
```