RecyclerView

2019-09-06

이승진

**학습목표**

RecyclerView 객체를 사용하여 문자열 목록을 화면에 표시하기

RecyclerView Adapter 구현하기

**목차**

[1. RecyclerView 구현 방법 2](#_Toc18868907)

[1) 문자열 목록을 표시하기 위해 필요한 클래스 2](#_Toc18868908)

[2) layout resource XML 파일을 구현해야 함 2](#_Toc18868909)

[2. RecyclerView1Activity 화면 만들기 3](#_Toc18868910)

[1) 새 액티비티 생성 3](#_Toc18868911)

[2) activity\_recycler\_view1.xml 수정 4](#_Toc18868912)

[3) item1.xml 생성 10](#_Toc18868913)

[4) RecyclerView Adapter 구현 12](#_Toc18868914)

[5) RecyclerView1Activity.java 수정 15](#_Toc18868915)

[3. MainActivity 수정 16](#_Toc18868916)

[1) activity\_main.xml 16](#_Toc18868917)

[2) MainActivity.java 17](#_Toc18868918)

[4. 결과 18](#_Toc18868919)

# RecyclerView 구현 방법

## 문자열 목록을 표시하기 위해 필요한 클래스

RecyclerView - ArrayAdapter<String> - ArrayList<String>

화면에 문자열 목록을 표시하기 위해, 위의 3 클래스 객체가 필요하다.

### RecyclerView 클래스

RecyclerView 클래스 객체는 화면에 문자열 목록을 표시하는 뷰(View) 객체이다.

좀 더 정확히 말하면, RecyclerView 클래스는 ViewGroup 클래스의 자식 클래스이다.

RecyclerView 객체는 단지 레이아웃 객체이다. 우리 눈에 보이는 것은, RecyclerView에 채워진 뷰(View) 객체들이다.

### ArrayList<String> 클래스

ArrayList<String> 클래스 객체는 화면에 표시할 문자열 목록을 보관하기 위한, 문자열 목록 클래스이다.

배열 형태의 자료구조 클래스이다.

### MyRecyclerViewAdapter 클래스

MyRecyclerViewAdapter 클래스를 구현해야 한다.

이 클래스 객체는 ArrayList<String> 객체와 RecyclerView 객체를 연결해 주는 중간 객체이다.

좀 더 자세히 말하자면, MyRecyclerViewAdapter 객체는, ArrayList<String> 객체에서 데이터를 한 개씩 가져다가,

이 데이터를 화면에 표시하기 위한 뷰(view) 개게를 생성해서, 이것을 RecyclerView 객체에 전달한다.

## layout resource XML 파일을 구현해야 함

MyRecyclerViewAdapter가 뷰 객체들을 생성할 때, layout inflation 기능을 활용한다.

layout infloation 기능은,

layout resource XML 파일의 내용대로 뷰(view) 객체들을 자동으로 생성해 주는 기능이다.

데이터 항목 한 개를 화면에 표시하기 위한 뷰(view) 객체와 그 속성을 layout resource 파일에 정의해야 한다.

즉 데이터 항목 한 개에 해당하는 layout resource 파일을 만들어야 한다.

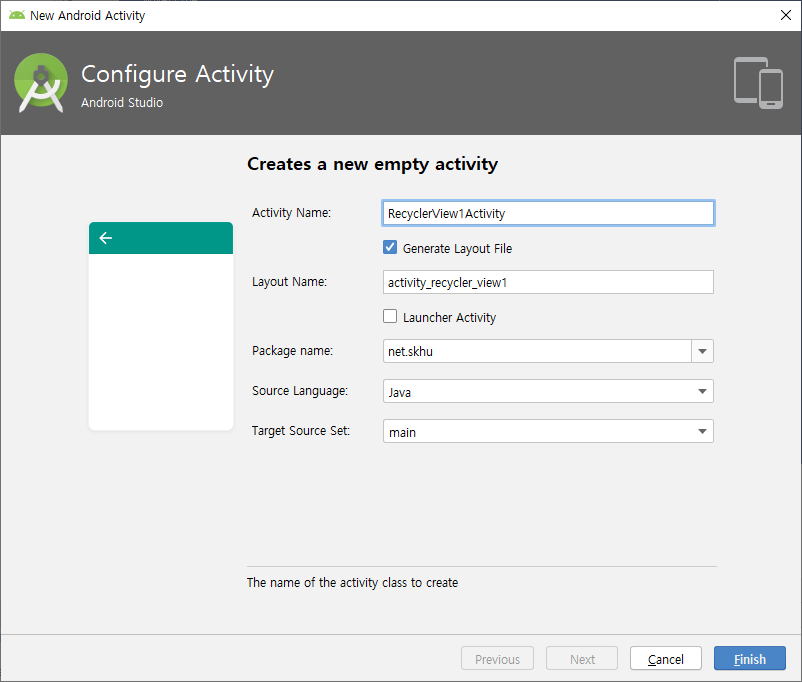
# RecyclerView1Activity 화면 만들기

## 새 액티비티 생성

E03List 프로젝트에 새 액티비티를 추가하자.

Project 창에서 app 폴더를 마우스 오른쪽 버튼으로 클릭하고

메뉴에서 New - Activity - Empty Activity 클릭



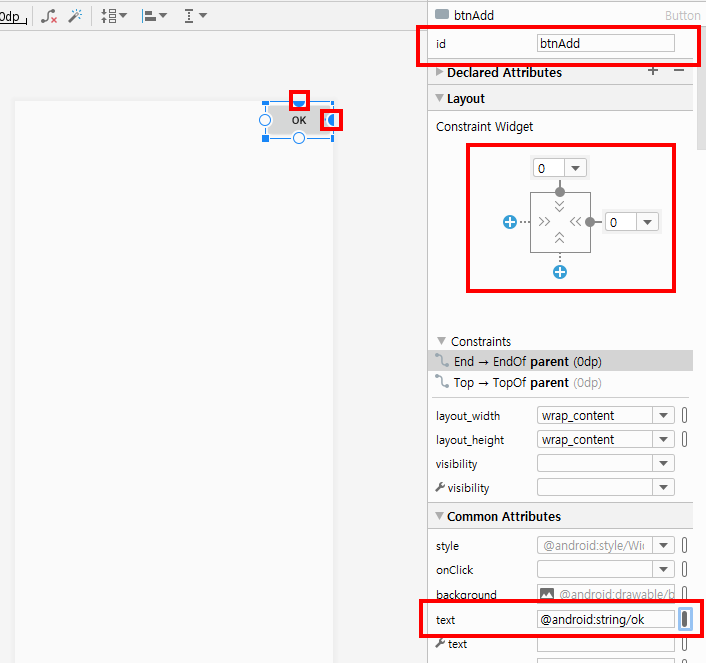
**파일 생성**

Activity Name: RecyclerView1Activity.java

Layout Name: activity\_recycler\_view1.xml

## activity\_recycler\_view1.xml 수정

### Button 객체 추가

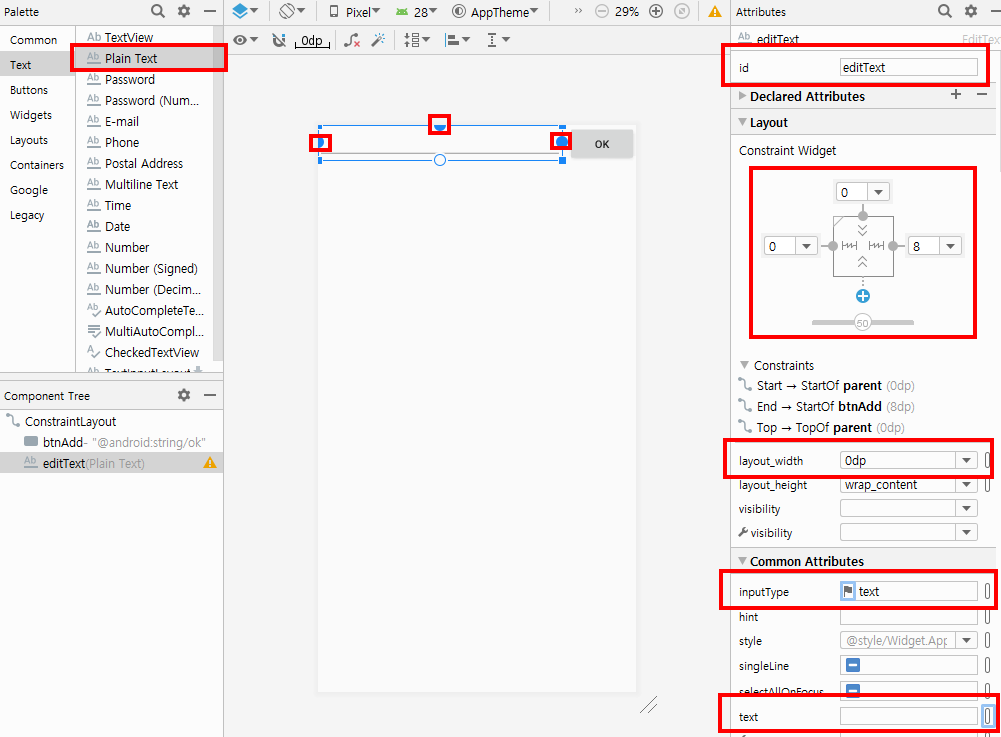


|  |  |
| --- | --- |
| 속성 | 값 |
| ID | btnAdd |
| text | @android:string/ok |

@android:string/ok 문자열 리소스는 안드로이드 시스템에 이미 정의되어 있는 문자열 리소스이다.

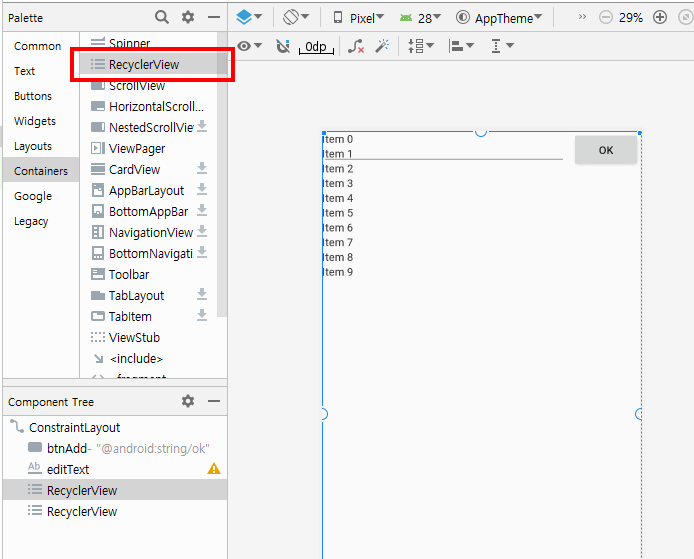
따라서, 문자열 리소스를 새로 만들 필요 없이, text 속성에 @android:string/ok 를 입력하기만 하면 된다.

### EditText 객체 추가

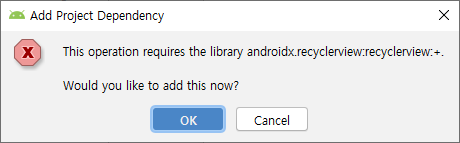


|  |  |
| --- | --- |
| 속성 | 값 |
| ID | editText |
| layout\_width | 0dp |
| inputType | text |
| text |  |

### RecyclerView 추가



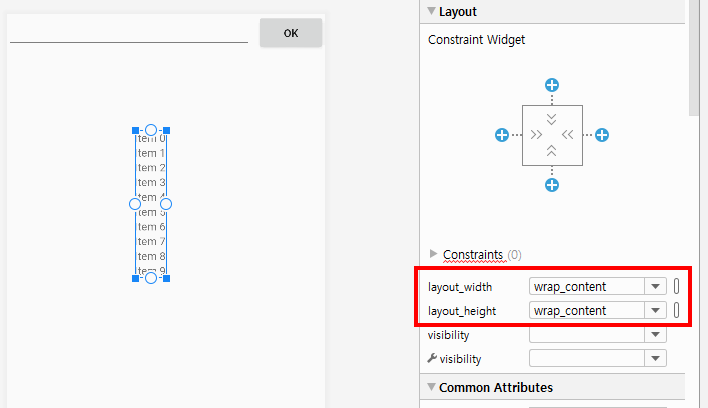
RecyclerView를 드래그 드롭하면, 아래와 같은 대화상자가 나타난다.



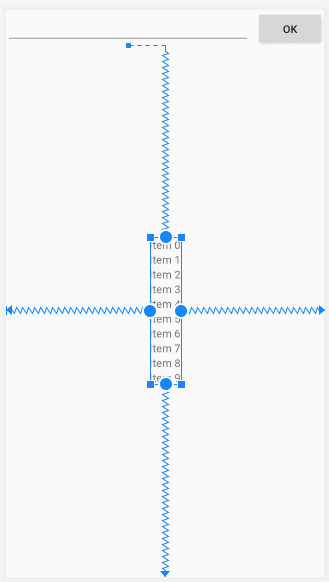
Ok 클릭하고,

메뉴: Build - Make Project 클릭

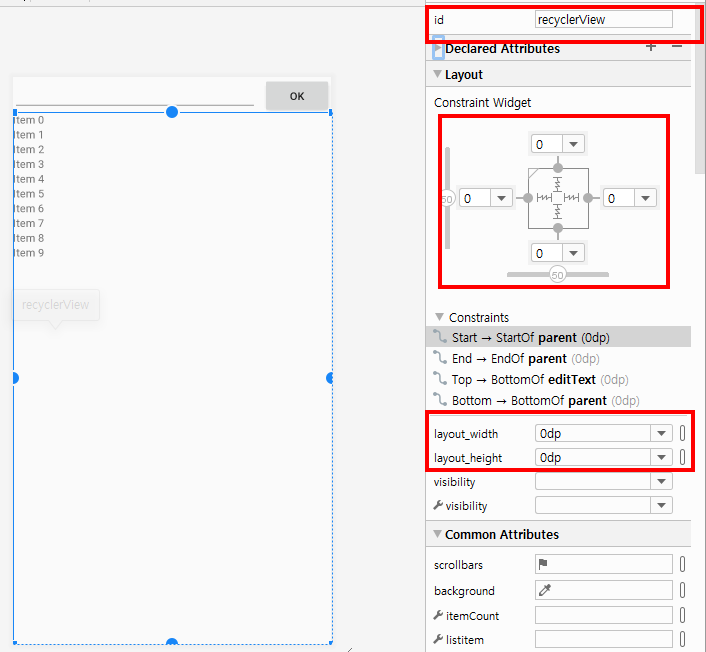
빌드 작업이 완료되면, Android Studio를 종료하고, 다시 실행하자.



layout\_width, layout\_height 속성에 wrap\_content 값을 입력하자.

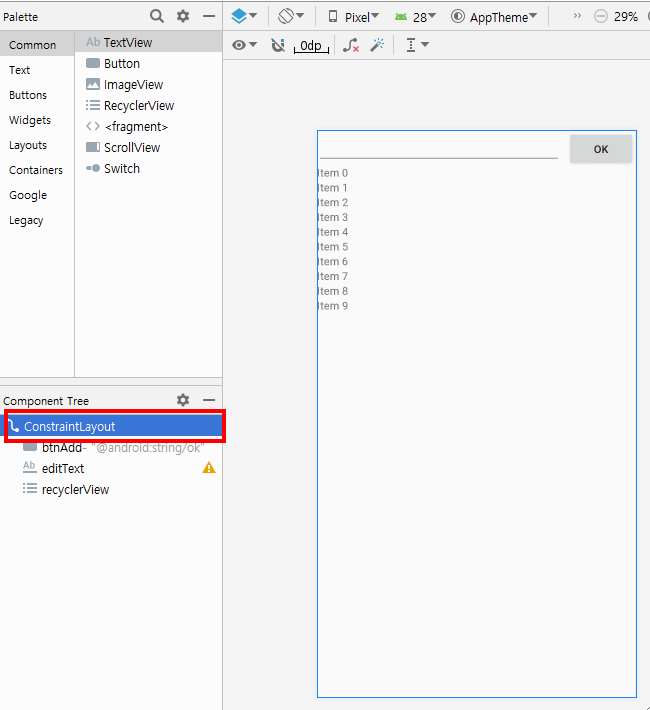


상, 하, 좌, 우 constraint 연결



|  |  |
| --- | --- |
| 속성 | 값 |
| ID | recyclerView |
| layout\_width | 0dp |
| layout\_height | 0dp |

### ConstraintLayout



지금까지 우리가 화면 배치(layout)를 편집한 것은,

ConstraintLayout 클래스의 기능을 사용하여 편집한 것이다.

안드로이드에서, 여러 뷰(View) 객체들을 자식으로 거느리면서,

자식 뷰 객체들의 화면 배치(layout)를 관리하는 클래스를

레이아웃 클래스라고 부른다.

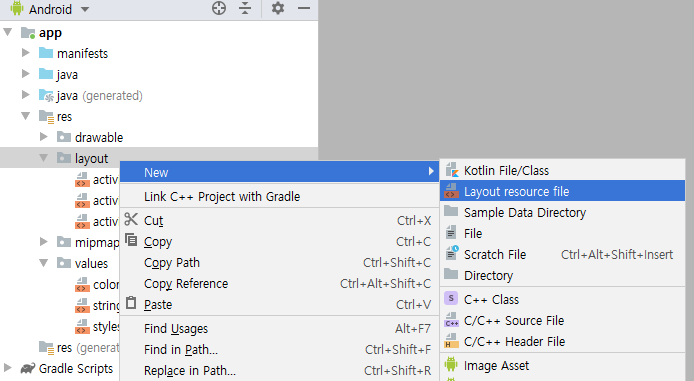
ConstraintLayout 클래스는 가장 대표적인 레이아웃 클래스이다.

ConstraintLayout 클래스는, 상하좌우 constaint 연결선으로 배치를 결정한다.

## item1.xml 생성

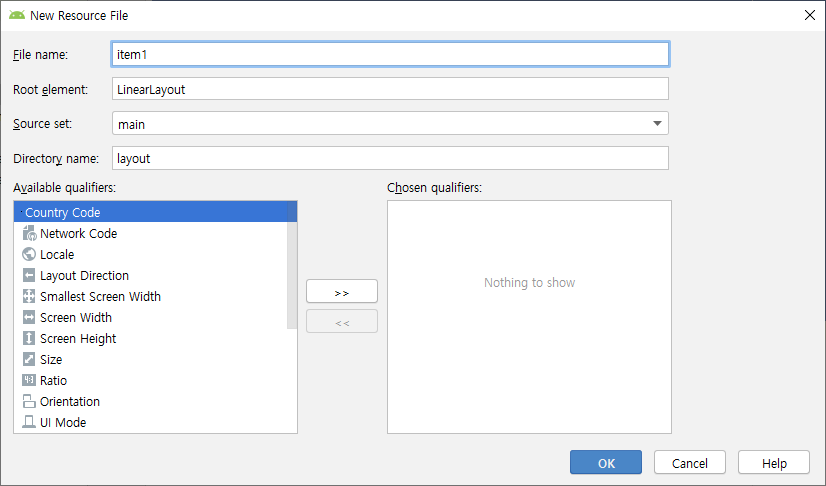
화면에 표시할 데이터 항목 한 개를 화면에 표시하기 위해서, 뷰 객체를 생성해야 한다.

이 뷰 객체를 자동 생성하기 위해 필요한 레이아웃 리소스 파일을 만들자.



layout 폴더를 우클릭하고,

메뉴: New - Layout resource file

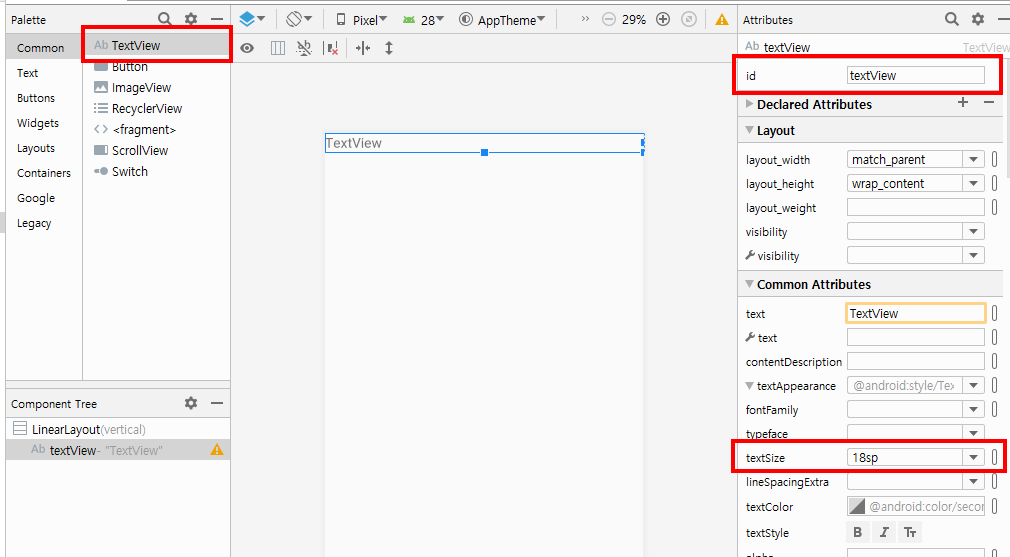


Root element 항목에 LinearLayout 클래스를 선택하였다.

### LinearLayout 클래스

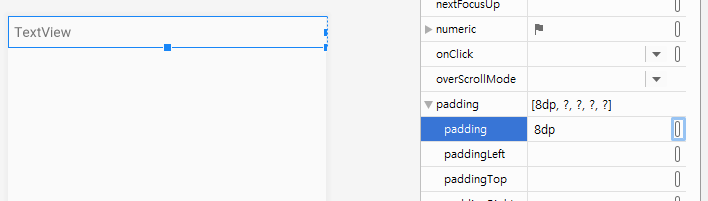
LinearLayout 클래스는 자식 뷰(view) 객체들을 단순하게 일렬로 배치한다.

### TextView 추가

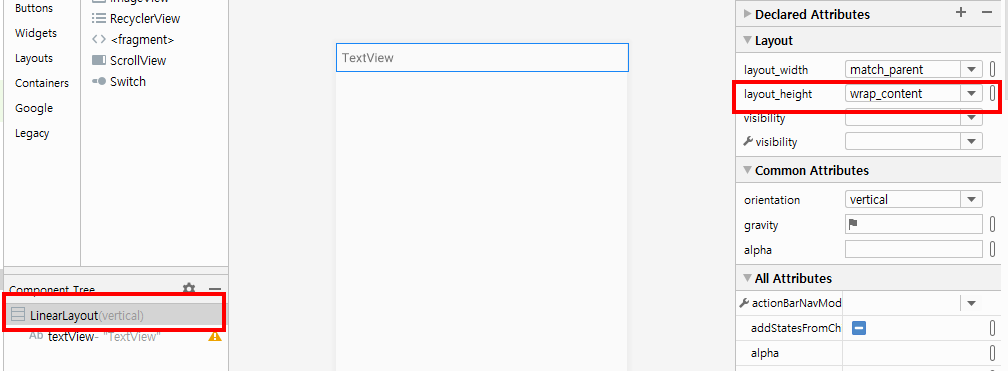


위 화면 왼쪽 아래의 Component Tree 창을 주목하자.

LinearLayout 객체의 자식 목록에, TextView 객체가 한 개 추가되었다.



TextView 객체의 padding 속성에 8dp 값을 대입하자.



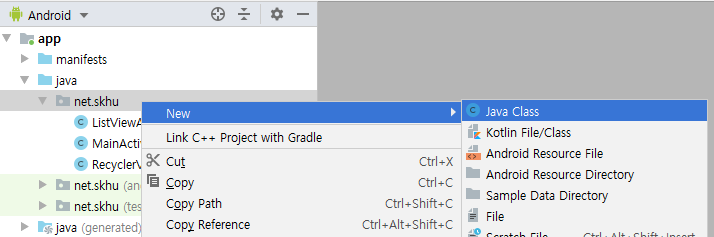
LinearLayout 객체의 layout\_height 속성에 warp\_content 값을 대입하자.

이것은 데이터 항목 한 개에 대한 레이아웃이므로, 이 항목의 높이는 데이터 항목 한 개의 높이여야 한다.

데이터 항목 한 개의 높이가 화면을 가득 채우면 곤란하다.

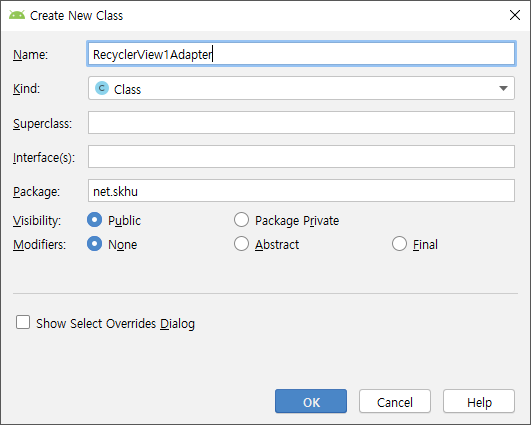
## RecyclerView Adapter 구현

### RecyclerView1Adapter.java 생성



net.skhu 패키지를 우클릭하고

메뉴: New - Java Class



### RecyclerView1Adapter.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45 | **package** net.skhu;  **import** android.content.Context;  **import** android.view.LayoutInflater;  **import** android.view.View;  **import** android.view.ViewGroup;  **import** android.widget.TextView;  **import** androidx.recyclerview.widget.RecyclerView;  **import** java.util.ArrayList;  **public class** RecyclerView1Adapter **extends** RecyclerView.Adapter<RecyclerView1Adapter.ViewHolder> {  **static class** ViewHolder **extends** RecyclerView.ViewHolder {  TextView **textView**;  **public** ViewHolder(View view) {  **super**(view);  **textView** = view.findViewById(R.id.***textView***);  }  }  LayoutInflater **layoutInflater**;  ArrayList<String> **arrayList**;  **public** RecyclerView1Adapter(Context context, ArrayList<String> arrayList) {  **this**.**layoutInflater** = LayoutInflater.*from*(context);  **this**.**arrayList** = arrayList;  }  @Override  **public int** getItemCount() {  **return arrayList**.size();  }  @Override  **public** ViewHolder onCreateViewHolder(ViewGroup viewGroup, **int** viewType) {  View view = **layoutInflater**.inflate(R.layout.***item1***, viewGroup, **false**);  **return new** ViewHolder(view);  }  @Override  **public void** onBindViewHolder(**final** ViewHolder viewHolder, **final int** index) {  viewHolder.**textView**.setText(**arrayList**.get(index));  }  } |

### ViewHolder 클래스

ViewHolder 객체는, 내부에 뷰(View) 객체를 보유하고있어야 한다.

이 뷰 객체는 데이터 항목 한 개를 화면에 보여주기 위한 뷰 객체이다.

이 데이터 항목 뷰 객체에 대한 작업을 구현해야 한다면, ViewHolder 클래스의 메소드로 구현한다. (예: 클릭)

ViewHolder 클래스는 보통 Adpater 클래스의 inner 클래스로 구현한다.

ViewHolder 클래스는 MyRecyclerViewAdpater 클래스의 inner class 이므로

outter class 클래스인 MyRecyclerViewAdapter 클래스의 멤버 변수인 arrayList 변수를 사용할 수 있다.

### ViewHolder 클래스의 getAdapterPosition() 메소드

ViewHolder 클래스의 getAdapterPosition() 메소드는

ViewHolder에 채워진 데이터 항목의 index 값을 리턴한다.

### getItemCount 메소드

데이터 항목의 수를 리턴한다.

### onCreateViewHolder 메소드

데이터 항목 뷰 객체를 생성하고

ViewHolder 객체를 생성해서 리턴한다.

### onBindViewHolder 메소드

ViewHolder 객체 내부의 데이터 항목 객체에 데이터를 채운다.

### template method 패턴

getItemCount, onCreateViewHolder, onBindViewHolder 메소드들은 간단한 작업을 구현한

단위 작업 메소드들이다.

이 메소드들을 호출하는 소스 코드는, 라이브러리의 RecyclerView 클래스에 구현되어 있다.

이 메소드 호출은 다형성 호출이어서, 프로그래머가 구현한 RecyclerView.Adapter 클래스의 자식 클래스의 메소드가

호출된다.

(template method 패턴 구조)

언제나 동일한 작업 절차의 지휘 통제 메소드는 부모 클래스에 구현되어 있고,

매번 달라지는 단위 작업 메소드는 자식 클래스에서 재정의한다.

부모 클래스의 지휘 통제 메소드가, 단위 작업 메소드를 호출하는 것은 다형성 호출이다.

GUI는 template method 패턴을 광범위하게 사용한다.

GUI 앱을 구현할 때, 부모 클래스를 상속해서, 몇몇 메소드를 재정의한다.

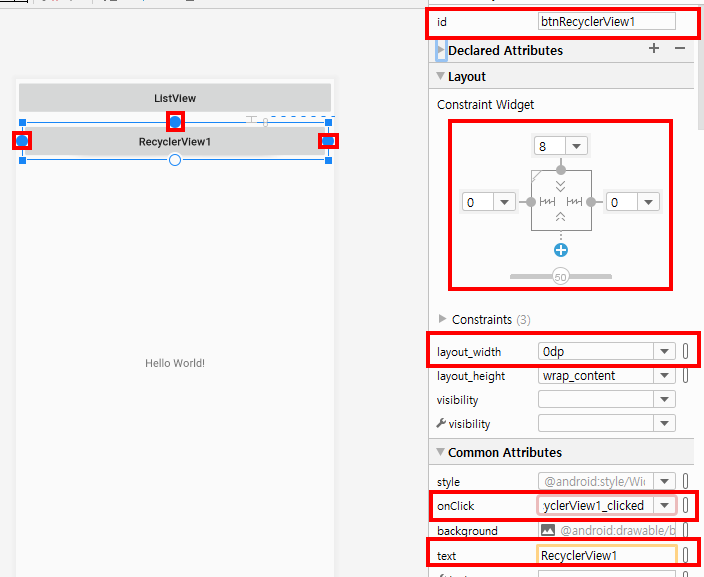
재정의된 메소드를 호출하는 소스코드는 라이브러리에 들어있다.

## RecyclerView1Activity.java 수정

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49 | **package** net.skhu;  **import** androidx.appcompat.app.AppCompatActivity;  **import** androidx.recyclerview.widget.DefaultItemAnimator;  **import** androidx.recyclerview.widget.DividerItemDecoration;  **import** androidx.recyclerview.widget.LinearLayoutManager;  **import** androidx.recyclerview.widget.RecyclerView;  **import** android.os.Bundle;  **import** android.view.View;  **import** android.widget.Button;  **import** android.widget.EditText;  **import** java.util.ArrayList;  **public class** RecyclerView1Activity **extends** AppCompatActivity {  RecyclerView1Adapter **recyclerView1Adapter**;  ArrayList<String> **arrayList**;  @Override  **protected void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.***activity\_recycler\_view1***);  **arrayList** = **new** ArrayList<String>();  **arrayList**.add(**"one"**);  **arrayList**.add(**"two"**);  **recyclerView1Adapter** = **new** RecyclerView1Adapter(**this**, **arrayList**);  RecyclerView recyclerView = (RecyclerView) findViewById(R.id.***recyclerView***);  recyclerView.addItemDecoration(**new** DividerItemDecoration(**this**, DividerItemDecoration.***VERTICAL***));  recyclerView.setLayoutManager(**new** LinearLayoutManager(**this**));  recyclerView.setItemAnimator(**new** DefaultItemAnimator());  recyclerView.setAdapter(**recyclerView1Adapter**);  Button b = (Button)findViewById(R.id.***btnAdd***);  b.setOnClickListener(**new** View.OnClickListener() {  @Override  **public void** onClick(View arg0) {  EditText e = (EditText) findViewById(R.id.***editText***);  String s = e.getText().toString();  e.setText(**""**);  **arrayList**.add(s);  **recyclerView1Adapter**.notifyDataSetChanged();  }  });  }  } |

# MainActivity 수정

## activity\_main.xml



|  |  |
| --- | --- |
| id | btnRecyclerView1 |
| layout\_width | 0dp |
| onClick | btnRecyclerView1\_clicked |
| text | RecyclerView1 |

## MainActivity.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | **package** net.skhu;  **import** androidx.appcompat.app.AppCompatActivity;  **import** android.content.Intent;  **import** android.os.Bundle;  **import** android.view.View;  **public class** MainActivity **extends** AppCompatActivity {  @Override  **protected void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.***activity\_main***);  }  **public void** btnListView\_clicked(View view) {  Intent intent = **new** Intent(**this**, ListViewActivity.**class**);  startActivity(intent);  }  **public void** btnRecyclerView1\_clicked(View view) {  Intent intent = **new** Intent(**this**, RecyclerView1Activity.**class**);  startActivity(intent);  }  } |

# 결과

E03List\_src2.zip

