화면에 데이터 목록 출력하기

2014-08-25

이승진

**목차**

[1. 예제 E02List #1 2](#_Toc397605464)

[1) 프로젝트 생성 2](#_Toc397605465)

[2) activity\_main.xml 수정 2](#_Toc397605466)

[3) MainActivity.java 수정 6](#_Toc397605467)

[4) 실행 8](#_Toc397605468)

[2. 예제 E02List #2 9](#_Toc397605469)

[1) ListItem 클래스 생성 9](#_Toc397605470)

[2) MainActivity.java 수정 11](#_Toc397605471)

[3) 실행 13](#_Toc397605472)

[3. 예제 E02List #3 14](#_Toc397605473)

[1) list\_item.xml 생성 14](#_Toc397605474)

[2) ListItemArrayAdapter 생성 19](#_Toc397605475)

[3) MainActivity.java 수정 20](#_Toc397605476)

[4) 실행 21](#_Toc397605477)

**학습목표**

1. 예제 E02List #1

문자열(String)의 목록을 화면에 표시하기

간단한 문자열의 목록을 화면에 표시한다.

2. 예제 E02List #2

임의의 객체의 목록을 화면에 표시하기

임의의 객체의 내부값을 하나의 문자열로 요약하여, 그 요약된 문자열 목록을 화면에 표시한다.

3. 예제 E02List #3

임의의 객체의 목록을 내가 원하는 화면 배치로 표시하기

임의의 객체의 내부 속성값 몇개를 내가 원하는 화면 배치로 표시한다.

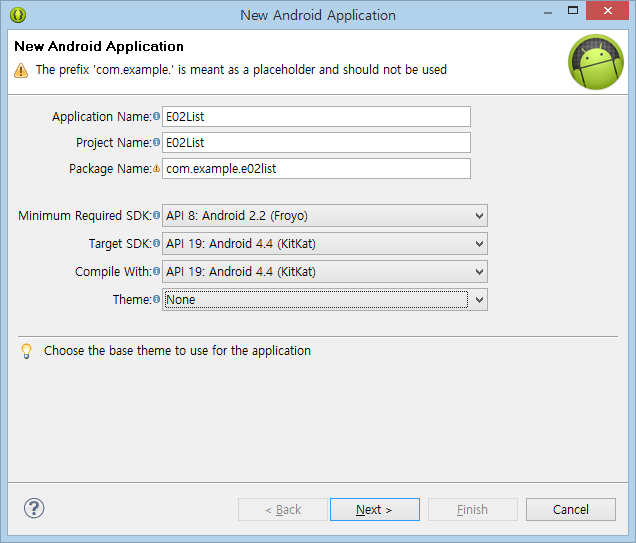
# 예제 E02List #1

문자열(String)의 목록을 화면에 표시하기

간단한 문자열의 목록을 화면에 표시한다.

## 프로젝트 생성

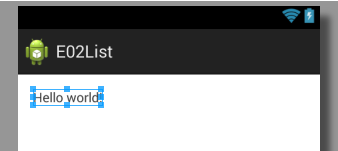
이클립스 메뉴: File – New – Andoid Applicaton Project



위와 같이 입력하고 Next를 계속 눌러서 진행하자.

## activity\_main.xml 수정

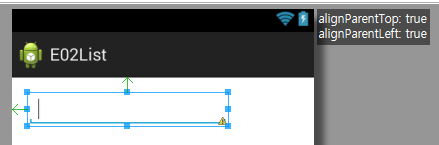
### textView1 삭제



Hello world! 라고 표시된 textView1 객체를 클릭하여 선택하고 Delete 키를 눌러서 삭제하자.

### editText1 추가

이전 예제에서 했던 것처럼, 문자열을 입력하기 위한 EditText를 추가하자



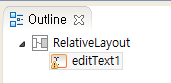
화면에 작은 녹색 화살표 두개를 주목하자.

그리고 오른쪽에 alignParentTop: true, alignParentLeft: true 표시를 주목하자.

윗방향 녹색 화살표와 alignParentTop은 같은 얘기를 하고 있다.

editText1의 윗 경계가 화면의 상단에 붙었다는 얘기다.

정확히 얘기하면 화면 상단이 아니고, editText1을 포함하고 있는 부모 객체의 상단이다.



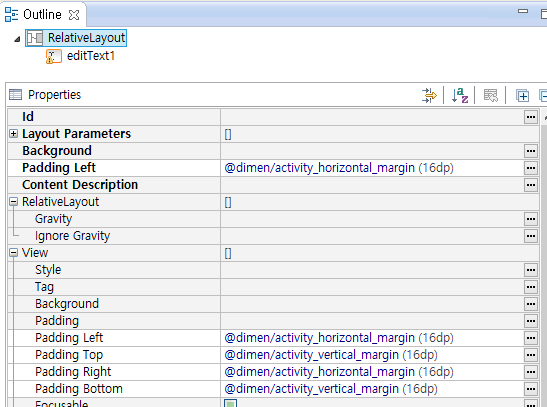
화면 오른쪽 Outline 창을 보면 RelativeLayout 객체가 editText1 객체를 포함하고 있는 부모 객체이다.

RelativeLayout 객체가 화면 전체 영역을 다 차지하고 있으면서, 자신의 내부에 포함된 자식 객체들에게 공간을 나누어주고 있다.

editText1을 화면 상단에 더 붙이려고 올려봐도 더 이상 올라가지 않는데,

그 이유는 editText1을 포함하고 있는 부모인 RelativeLayout이 허용하지 않기 때문이다.

허용하지 않는 이유는 다음과 같다.



Outline 창에서 RelativeLayout 객체를 클릭하여 선택하면,

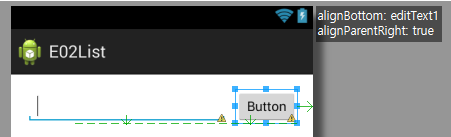
그 아래 Properties 창에서 RelativeLayout 객체의 속성이 표시된다.

이 속성에서 Padding 속성들에 16dp 값이 주어져 있다.

Padding 속성은 내부 여백을 지정한다.

즉 RelativeLayout의 Left, Top, Right, Bottom 4 변(border)과 내부 자식들과의 여백이 16dp이다.

### button1 추가



화면에 작은 녹색 화살표 두개를 주목하자.

그리고 오른쪽에 alignBottom: editText1, alignParentRight: true 표시를 주목하자.

추가된 버튼은 아래 경계를 editText1과 맞춘다.

그리고 오른쪽 경계는 부모에 오른쪽에 맞춘다.

안드로이드 기기들의 화면 크기는 다양하다.

큰 화면에서도 작은 화면에서도 적절하게 배치가 유지되어야 한다.

만약 우리가 지금 그리고 있는 화면보다 더 큰 화면에서 실행될 때, 저 버튼의 위치는 어디일까?

alignParentRight: true 이므로 화면의 오른쪽에 붙어 있어야 하기 때문에,

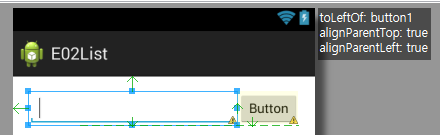
화면이 커지면 버튼은 화면 오른쪽 경계를 따라 오른쪽으로 더 이동하게 된다.

화면이 커질 경우에 editTex1의 크기는 저절로 커질까?

editText1을 배치할 때 우리가 지정한 속성은 alignParentTop: true, alignParentLeft: true 이거 둘 뿐이다.

저절로 커지라는 속성은 주지 않았다.

### editText1 속성 수정



editText1의 오른쪽 경계를 드래그하여 button1의 왼쪽에 붙이자.

그러면 화면 오른쪽에 toLeftOf: button1 속성 표시가 추가된다.

button1의 왼쪽 경계에 붙으라는 설정이다.

RelativeLayout에는 여백을 16dp 주었기 때문에 딱 붙여도 16dp의 여백을 사이에 두고 붙는데,

editText1과 button1에는 여백 속성을 주지 않았기 때문에 붙이면 저렇게 딱 붙는다.

자 이제 화면이 커지면, 화면의 오른쪽 경계에 붙어있는 button1은 화면의 오른쪽 경계를 따라 오른쪽으로 이동하게 되고, button1의 왼쪽에 붙어있고 또한 화면의 왼쪽 경계에도 붙어있는 editText1은 양쪽으로 붙어 있어야 하므로 좌우로 늘어나게 된다.

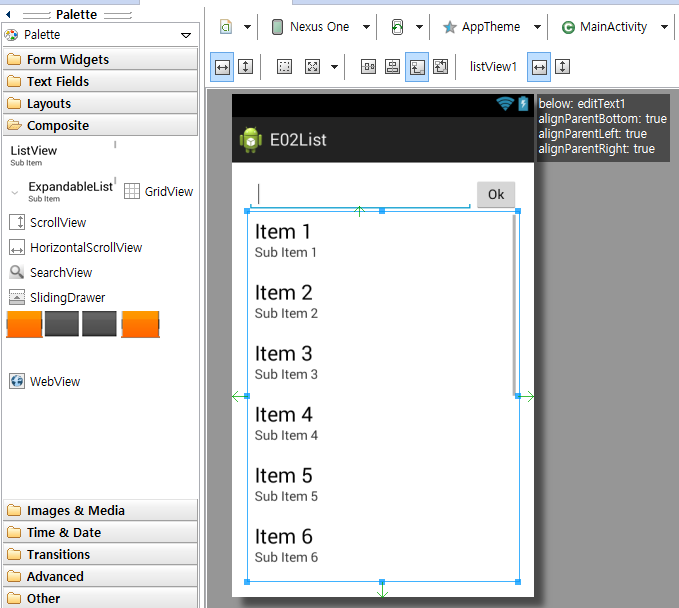
### 노란색 세모 경고 표시

editText1과 button1에 붙어있는 노란색 세모 경고 표시가 붙어있다.

이전 예제에서 했던 절차대로 속성을 설정하여 경고 표시를 끄자.



### listView1 추가



화면 왼쪽 Palette에서 Composite 아래의 ListView를 드래그하여 화면에 추가하자.

위 아래 오른쪽 왼쪽 4개의 작은 녹색 화살표 표시에 주목하자.

그리고 화면 오른쪽 4개의 속성 표시도 주목하자.

below: editText1 → editText1의 아래에 붙으라는 속성 설정이다.

alignParentBottom: true → 화면의 아래 경계에 딱 붙으라는 속성 설정이다.

alignParentLeft: true → 화면의 왼쪽 경계에 딱 붙으라는 속성 설정이다.

alignParentRight: true → 화면의 오른쪽 경계에 딱 붙으라는 속성 설정이다.

위 아래로 붙어있어야 하고, 좌우로도 붙어 있어야 하니, 화면이 커지면 위아래로 좌우로 커지게 된다.

안드로이드 기기들의 화면 크기는 다양하니, 화면의 크기에 따라 저절로 늘어나고 줄어들 수 있도록 배치해야 한다.

단축기 Ctrl-S, Ctrl-F4를 차례대로 눌러서 activty\_main.xml를 저장하고 창을 닫자.

## MainActivity.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  50  51  52  53  54  55  56  57  58  59  60 | package com.example.e02list;  import java.util.ArrayList;  import android.support.v7.app.ActionBarActivity;  import android.os.Bundle;  import android.view.Menu;  import android.view.MenuItem;  import android.view.View;  import android.view.View.OnClickListener;  import android.widget.ArrayAdapter;  import android.widget.Button;  import android.widget.EditText;  import android.widget.ListView;  public class MainActivity extends ActionBarActivity {    ArrayList<String> arrayList;  ArrayAdapter<String> adapter;    @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main);    arrayList = new ArrayList<String>();  arrayList.add("One");  arrayList.add("Two");  adapter = new ArrayAdapter<String>(this, android.R.layout.simple\_list\_item\_1, arrayList);  ListView listView = (ListView)findViewById(R.id.listView1);  listView.setAdapter(adapter);    Button b = (Button)findViewById(R.id.button1);  b.setOnClickListener(new OnClickListener() {  @Override  public void onClick(View arg0) {  EditText e = (EditText)findViewById(R.id.editText1);  CharSequence s = e.getText();  e.setText("");  arrayList.add(s.toString());  adapter.notifyDataSetChanged();  }  });  }  @Override  public boolean onCreateOptionsMenu(Menu menu) {  getMenuInflater().inflate(R.menu.main, menu);  return true;  }  @Override  public boolean onOptionsItemSelected(MenuItem item) {  int id = item.getItemId();  if (id == R.id.action\_settings) {  return true;  }  return super.onOptionsItemSelected(item);  }  } |

(줄18) Java에서 객체의 목록을 저장하는 자료구조로 ArrayList 클래스를 자주 사용한다.

ArrayList<String> 타입 선언은 ArrayList에 저장할 객체의 타입이 String이라는 뜻이다.

arrayList는 멤버 변수일 뿐이가 객체가 아니다.

(줄26) 여기에서 ArrayList<String> 타입의 객체가 생성되고, 그 객체에 대한 참조가 멤버 변수 arrayList에 저장된다.

(줄27,28) arrayList에 문자열 객체 두 개를 저장한다.

(줄19,29) ArrayAdapter 클래스는 ArrayList에 들어있는 데이터 목록을 ListView를 통해서 화면에 출력하기 위해 필요한 중간 각체이다. ArrayList에 저장할 데이터가 String이므로, ArrayList<String>, ArrayAdapter<String> 타입으로 선언해야한다.

(줄29) ArrayAdapter 생성자의 첫째 파라미터 this는 액티비티 객체이다.

둘째 파라미터는 데이터 항목 하나를 화면에 어떤 형태로 표시할지를 지정한다. 문자열 하나를 간단히 표시할 때 android.R.layout.simple\_list\_item\_1 이다.

셋째 파라미터는 화면에 표시할 데이터 목록에 해당하는 ArrayList 객체이다.

(줄24) R.layout.activity\_main 레이아웃 리소스의 내용대로 액티비티 화면을 만들라는 메소드 호출이다.

이 파일의 내용대로 EditText, Button, ListView 객체가 자동으로 만들어진다.

(줄30) id가 R.id.listView1인 ListView객체에 대한 참조를 얻는 메소드 호출이다.

이 ListView 객체는 줄24의 메소드 호출에 의해서 생성되었다.

(줄31) ListView 객체에 ArrayAdapter 객체를 연결한다. ArrayAdapter는 또 ArrayList와 연결되어 있으므로, ListView, ArrayAdapter, ArrayList 3개의 객체가 연결되었다. 자 이제 ArrayList에 들어있는 데이터 목록이 화면에 표시된다.

(줄33) id가 R.id.button1인 Button객체에 대한 참조를 얻는 메소드 호출이다.

이 Button 객체는 줄24의 메소드 호출에 의해서 생성되었다.

(줄34~43) 온클릭리스너(OnClickListener) 객체를 하나 생성하여, 버튼 객체에 연결한다.

이제 버튼이 클릭되면 연결된 온클릭리스너 객체의 onClick 메소드가 즉시 호출될 것이다.

여기서는 온클릭리스너를 버튼에 연결만 할 뿐이지, 아직 onClick 메소드는 호출되지 않는다.

(줄37~41) 버튼이 클릭되면 호출되는 메소드이다.

(줄37) id가 R.id.editText1인 EditText 객체에 대한 참조를 얻는 메소드 호출이다.

이 EditText 객체는 줄24의 메소드 호출에 의해서 생성되었다.

(줄38) EditText 객체에 입력된 문자열을 얻는다.

(줄39) EditText에 입력된 문자열을 빈문자열로 교체한다. 즉 문자열 입력 텍스트 박스의 내용이 지워진다.

(줄40) EditText에 입력되었던 문자열을 ArrayList에 추가한다. 이 ArrayList 객체는 ListView와 연결되어 있다. ArrayList에 들어있는 문자열 목록은 ListView에 의해 화면에 표시된다. 따라서 ArrayList에 문자열을 추가하면 추가된 내용도 화면에 표시된다. 그런데 ArrayList에 데이터가 추가되거나 삭제된 사실을 ListView가 즉시 알지는 못하므로 추가된 내용이 화면에 표시되려면 시간이 좀 걸린다.

(줄41) ArrayList에 들어있는 데이터가 변경되었으니 화면을 다시 그리라고 ListView에게 통보하기 위한 메소드 호출이다. ArrayAdapter의 notifyDataSetChanged 메소드를 호출하면 ListView에게 통보되어 화면이 다시 그려진다.

MainActivity 객체에 create 이벤트가 발생했을 때 실행되는 줄 번호 (순서대로)

22, 23, 24, 26, 27, 28, 29, 30, 31, 33, 34, 44

줄34를 실행할 때 일어난 일

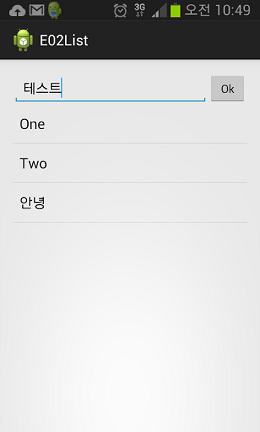
- 온 클릭 리스너 객체가 하나 생성되고

- 그 객체에 대한 참조를 파라미터 값으로 버튼 객체의 setOnClickListener 메소드가 호출된다

버튼 객체에 click 이벤트가 발생했을 때 실행되는 줄 번호 (순서대로)

36, 37, 38, 39, 40, 41, 42

## 실행



목록의 길이가 길어서 한 화면을 넘어가면 화면을 터치하여 스크롤 할 수 있다.

# 예제 E02List #2

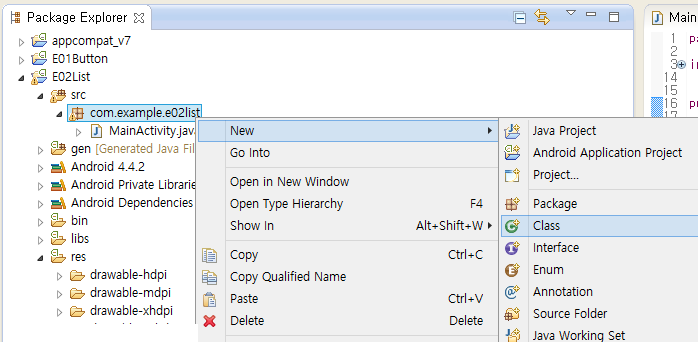
임의의 객체의 목록을 화면에 표시하기

임의의 객체의 내부값을 하나의 문자열로 요약하여, 그 요약된 문자열 목록을 화면에 표시한다.

E02List 예제를 개선해 보자. 문자열뿐만 아니라 그 문자열이 입력된 시각도 화면에 표시해 보자.

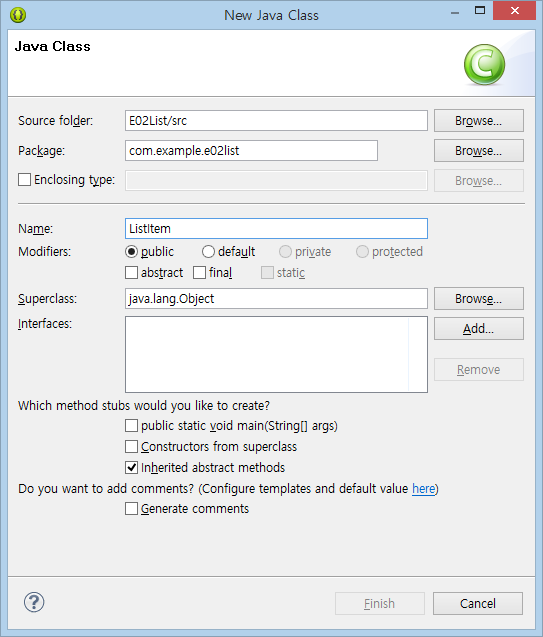
먼저 문자열과 시각 데이터를 멤버로 갖는 클래스부터 하나 새로만들자.

## ListItem 클래스 생성



위 그림과 같이 com.example.e02list 패키지를 우 클릭하고

메뉴: New - Class



Package: com.example.e02list

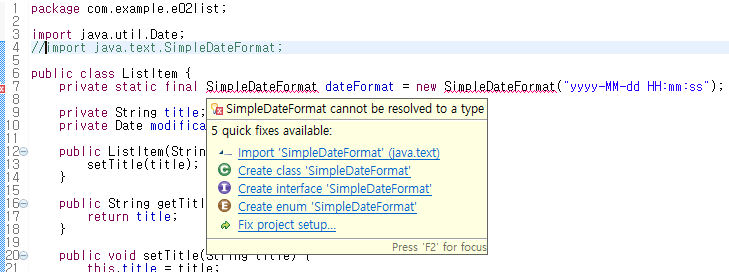
Name: ListItem

### ListItem.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 | package com.example.e02list;  import java.util.Date;  import java.text.SimpleDateFormat;  public class ListItem {  private static final SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");  private String title;  private Date modifiedTime;    public ListItem(String title) {  setTitle(title);  }  public String getTitle() {  return title;  }  public void setTitle(String title) {  this.title = title;  modifiedTime = new Date();  }  public String getModifiedTime() {  return dateFormat.format(modifiedTime);  }  @Override  public String toString() {  return getTitle() + " " + getModifiedTime();  }  } |

(줄4) 소스코드에서 SimpleDateFormat 클래스를 사용하려면, 이 import 선언이 필요하다.

다음 그림과 같이 줄4를 주석으로 막아보자.



import 선언이 없으면 위와 같이 SimpleDatFormat 클래스명에 컴파일 에러를 뜻하는 빨간색 밑줄이 표시된다.

그 빨간색 밑줄에 마우스 커스를 놓고 조금 기다리면 에러메시지에 대한 설명과 해결책을 제시하는 노란색 사각형이 위와 같이 나타난다.

import 선언이 없으면 발생하는 에러 메시지는 "클래스명 cannot be resolved to a type" 이다.

자주 발생하는 에러이므로, 이 에러 메시지를 기억해 두자.

import 선언을 안했거나, 아니면 클래스명에 오타가 있을 때 이 컴파일 에러가 발생한다.

오타가 아니라면 import 선언을 추가하면 되는데, 위 노란색 사각형에서 "import 'SimpleDateFormat' (java.text)"를 클릭하면 자동 추가된다.

(줄7) Date 객체에 들어있는 날짜와 시각을 원하는 형태로 출력하려면 클래스 객체가 필요하다.

Date 객체에 저장된 시각으로부터 '2014-09-02 13:25:46' 형태의 문자열을 얻기 위해 SimpleDateFormat 클래스를 사용한다. SimpleDateFormat 클래스의 생성자 파라미터로 날짜와 시각 출력 형태를 지정한다.

Java 문법에서 static final 멤버 변수가 무엇인지 조사하라.

(줄9) 화면에 표시할 문자열 하나를 저장할 멤버 변수

(줄10) 문자열이 수정된 시각을 저장할 멤버 변수

(줄22) Date 클래스 객체를 생성하여, 그 객체에 대한 참조를 멤버 변수에 대입한다.

생성된 Date 클래스 객체에는 현재 날짜와 시각이 저장되어 있다.

(줄26) SimpleDateFormat 클래스의 format 메소드를 호출한다.

이 메소드의 파라미터는 날짜와 시각을 가지고 있는 Date 객체이다.

이 메소드는 파라미터로 전달된 날짜와 시각을 정해진 형태의 문자열로 만들어 리턴한다.

(줄30) toString() 메소드는 Object 클래스의 메소드이다.

Java에서 모든 클래스의 가장 조상이되는 부모 클래스는 Object 클래스이다.

Object 클래스의 toString() 메소드는 Java의 모든 클래스에 상속된다.

상속받은 Object 클래스의 toString() 메소드 구현이 적당하지 않으면 위와 같이 자식 클래스에서 재정의(override) 하면 된다. 상속 받은 메소드를 재정의할 때, 줄29의 @Override 어노테이션을 메소드 앞에 붙인다.

Java 객체를 문자열 형태로 출력하려고 할 때 toString() 메소드를 호출하여 그 리턴 값을 사용한다.

줄31의 멤버 변수 title의 문자열과 SimpleDateFormat 객체의 format 메소드가 리턴한 문자열을 합해서 리턴한다.

## MainActivity.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  50  51  52  53  54  55  56  57  58  59  60 | package com.example.e02list;  import java.util.ArrayList;  import android.support.v7.app.ActionBarActivity;  import android.os.Bundle;  import android.view.Menu;  import android.view.MenuItem;  import android.view.View;  import android.view.View.OnClickListener;  import android.widget.ArrayAdapter;  import android.widget.Button;  import android.widget.EditText;  import android.widget.ListView;  public class MainActivity extends ActionBarActivity {    ArrayList<ListItem> arrayList;  ArrayAdapter<ListItem> adapter;    @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main);    arrayList = new ArrayList<ListItem>();  arrayList.add(new ListItem("One"));  arrayList.add(new ListItem("Two"));  adapter = new ArrayAdapter<ListItem>(this, android.R.layout.simple\_list\_item\_1, arrayList);  ListView listView = (ListView)findViewById(R.id.listView1);  listView.setAdapter(adapter);    Button b = (Button)findViewById(R.id.button1);  b.setOnClickListener(new OnClickListener() {  @Override  public void onClick(View arg0) {  EditText e = (EditText)findViewById(R.id.editText1);  CharSequence s = e.getText();  e.setText("");  arrayList.add(new ListItem(s.toString()));  adapter.notifyDataSetChanged();  }  });  }  @Override  public boolean onCreateOptionsMenu(Menu menu) {  getMenuInflater().inflate(R.menu.main, menu);  return true;  }  @Override  public boolean onOptionsItemSelected(MenuItem item) {  int id = item.getItemId();  if (id == R.id.action\_settings) {  return true;  }  return super.onOptionsItemSelected(item);  }  } |

E02List #1 예제의 MainActivity.java는 String 객체의 목록을 ArrayList에 저장하고 화면에 표시하였다.

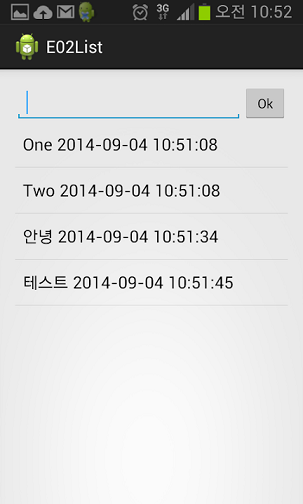
여기서는 String 객체 대신 ListItem 객체의 목록을 ArrayList에 저장하고 화면에 표시한다.

위 소스코드에서 수정된 부분만 형광색으로 표시되었다.

(줄18, 19, 26, 29) 목록에 저장된 객체의 타입이 String 클래스에서 ListItem 클래스로 수정.

(줄27, 28, 40) 목록에 String 객체를 저장하는 대신, ListItem 객체를 생성하여 저장.

## 실행



위와 같이 입력된 문자열 뿐만 아니라 그 시각도 화면에 표시된다.

화면에 표시되는 내용은 목록에 저장된 객체의 toString 메소드의 리턴값이다.

# 예제 E02List #3

임의의 객체의 목록을 내가 원하는 화면 배치로 표시하기

임의의 객체의 내부 속성값 몇개를 내가 원하는 화면 배치로 표시한다.

지금까지 ArrayAdapter를 이용하여 객체의 목록을 화면에 표시하였다.

ArrayAdapter는 화면에 표시될 객체의 내용이 문자열 하나라는 제약이 있다.

객체의 toString() 메소드의 리턴값인 문자열이 화면에 TextView 하나로 표시된다.

화면에 문자열 두 개를 표시해 보자.

ListItem 클래스의 title 값을 표시하는 TextView와, modifiedDate 값을 표시하는 TextView가 필요하다.

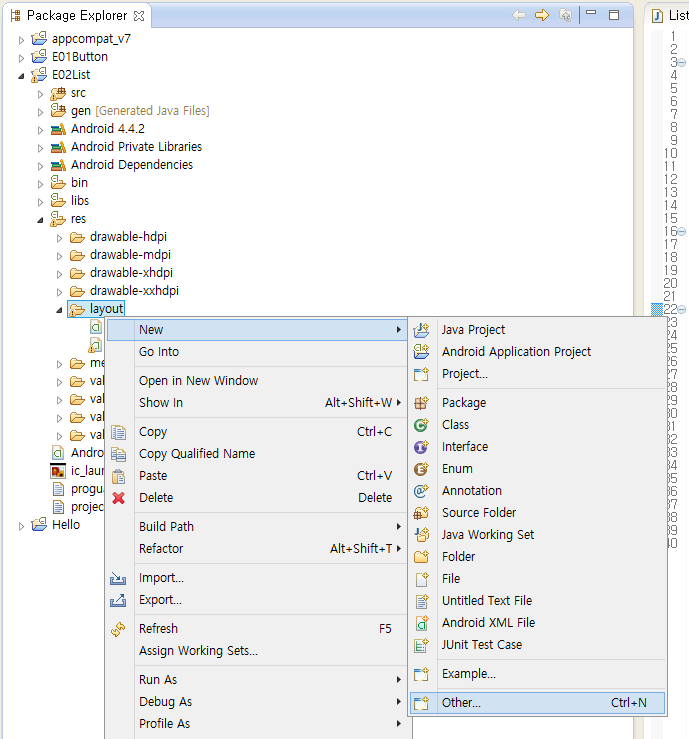
객체의 toString() 메소드의 리턴값인 문자열이 화면에 TextView 하나로 표시하는 ArrayAdapter로는

이것을 구현할 수 없다. 아탑타를 새로 만들어야 한다.

## list\_item.xml 생성

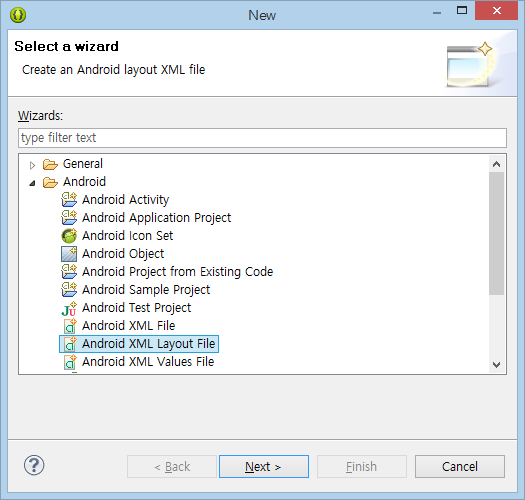
list\_item.xml 레이아웃 리소스 파일을 생성하자.

ListItem을 화면에 표시할 화면 배치 형태(layout)을 지정하기 위한 레이아웃 리소스 파일을 생성하자.

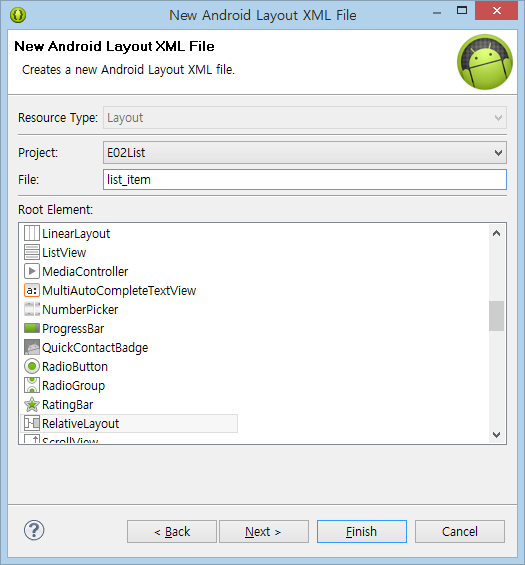


res 폴더 아래 layout 폴더를 우클릭하고

메뉴: New - Other...

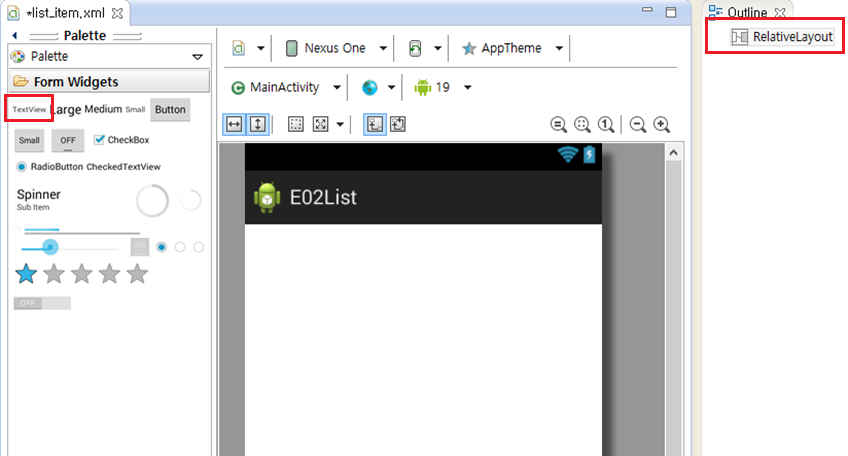


Android XML Layout File 항목 선택하고 Next 버튼 클릭



File: list\_item

RelativeLayout 선택하고 Finish 버튼 클릭



위와 같이 화면 배치(layout)을 만들기 위한 편집창이 열린다.

지금 여기서 우리가 편집해야 하는 레이아웃은 액티비티 화면 전체의 레이아웃이 아니고

ListItem 객체 하나를 화면에 표시하기 위한 레이아웃이다.

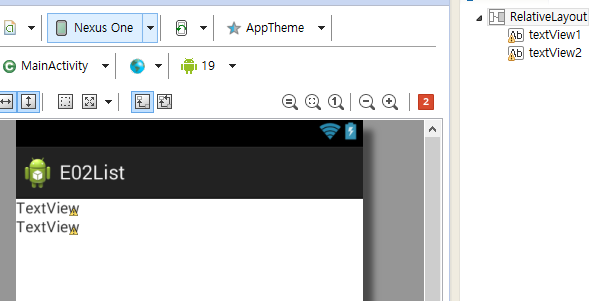
화면 오른쪽 Outline 창에 표시된 RelativeLayout에 주목하자.

Outline창은 현재 편집창에 배치된 뷰 객체들의 목록을 보여준다.

우리 눈에는 보이지 않지만, 지금 편집창에 RelativeLayout 객체가 하나 들어있다.

우리가 추가할 뷰 객체들은 이 RelativeLayout 객체의 자식 객체가 된다.

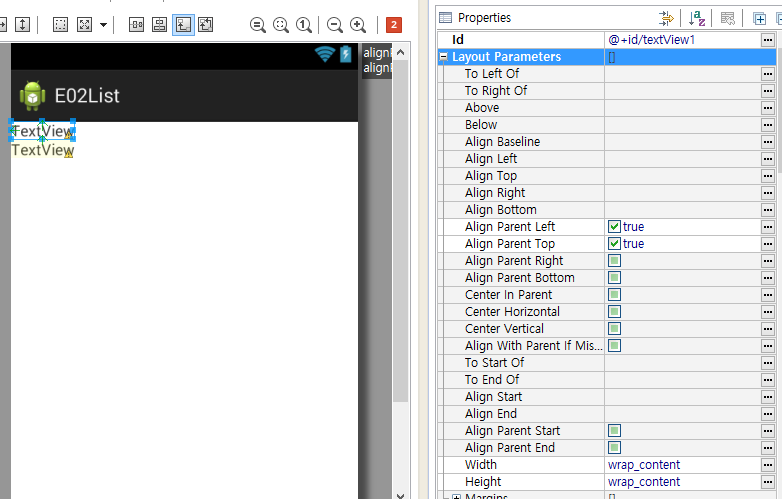
화면 왼쪽의 Palette 창의 TextView를 드래그 드롭하여 편집창에 두 번 추가하자.



화면 왼쪽의 Layout 창에 주목하자.

RelativeLayout 객체가 루트(root)이고, 방금 우리가 추가한 TextView 객체들이 RelativeLayout 객체의 자식으로 아래에 추가된 것을 확인할 수 있다. 그리고 이들의 id가 각각 textView1, textView2임을 확인할 수 있다.

첫번째 TextView를 선택해 보자.



textView1을 선택하고 오른쪽 Properties창을 보면, 선택된 textView1의 속성값을 볼 수 있다.

속성창의 속성 목록에서 Layout Parameters 항목 아래를 펼쳐보면 width 속성값이 warp\_context이다.

이 속성값의 의미는 TextView의 폭(width)이 내부 문자열(content)의 크기와 같다는 것.

왼쪽 편집창을 봐도, TextView의 폭이 문자열의 크기와 같음을 알 수 있다.

이 width 속성값을 match\_parent로 바꾸자.

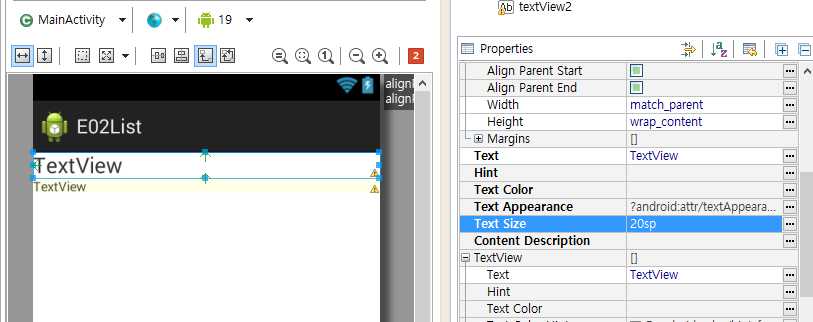
그리고 Text Size 속성값을 20sp로 입력하자.

textView1 속성 편집

|  |  |
| --- | --- |
| 속성 | 값 |
| Wdth | match parent |
| Text Size | 20sp |

textView2속성 편집

|  |  |
| --- | --- |
| 속성 | 값 |
| Wdth | match parent |
| Text Size | 12sp |



편집한 내용을 저장하고 편집창을 닫자.

## ListItemArrayAdapter 생성

### ListItemArrayAdapter.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 | package com.example.e02list;  import java.util.ArrayList;  import android.content.Context;  import android.view.LayoutInflater;  import android.view.View;  import android.view.ViewGroup;  import android.widget.ArrayAdapter;  import android.widget.TextView;  public class ListItemArrayAdapter extends ArrayAdapter<ListItem> {    private ArrayList<ListItem> arrayList;  private int resource;  public ListItemArrayAdapter(Context context, int resource, ArrayList<ListItem> arrayList) {  super(context, resource, arrayList);  this.resource = resource;  this.arrayList = arrayList;  }    @Override  public View getView(int position, View convertView, ViewGroup parent) {  View view = convertView;  if (view == null) {  LayoutInflater inflater =  (LayoutInflater)getContext().getSystemService(Context.LAYOUT\_INFLATER\_SERVICE);  view = inflater.inflate(resource, null);  }  ListItem item = arrayList.get(position);  if (item != null) {  TextView textView1 = (TextView)view.findViewById(R.id.textView1);  TextView textView2 = (TextView)view.findViewById(R.id.textView2);  if (textView1 != null) textView1.setText(item.getTitle());  if (textView2 != null) textView2.setText(item.getModifiedTime());  }  return view;  }  } |

(줄11) ArrayAdapter<ListItem>를 상속 받아서 ListItemArrayAdapter 클래스를 구현한다.

위 소스코드를 지금 다 이해할 필요는 없다.

지금은 이 아답타가 하는 일이 무엇인지만 이해하자.

(가) 아답타는 데이터를 화면에 보여주기 위한 뷰(View) 객체를 만들고,

(나) ArrayList에 들어있는 객체의 데이터를 뷰 객체에 채우고,

(다) 그 뷰 객체를 리턴하여 ListView에 전달하는 일을 한다.

이 일을 하는 아답타의 메소드가 (줄23)의 getView 메소드이다.

getView 메소드의 리턴 타입이 View 객체이다.

안드로이드 화면을 구성하는 요소들을 뷰(View) 객체라고 부른다.

화면의 구성 요소인 TextView, Button, EditText, ListView, RelativeLayout 들이 뷰 객체이다.

이들 뷰 객체들의 공통 부모 클래스가 View 클래스이다.

(줄24~29) 화면에 데이터를 보여주기 위한 뷰 객체를 만든다. (가)

(줄30) 화면에 보여줄 데이터 객체에 대한 참조를 얻는다. (나)

(줄32~35) 뷰 객체들에 데이터를 채운다. (나)

(줄37) 뷰 객체를 리턴한다. 리턴된 뷰 객체를 Listview가 받아서 화면에 보여준다. (다)

이 getView 메소드는 화면에 표시할 데이터 객체 하나 당 한 번씩 호출된다.

즉 화면의 ListView의 한 줄 당 한 번씩 호출된다.

지금은 이정도로만 이해하자.

만약 다른 데이터 객체를 화면에 표시하기 위한 아답타를 만들게 될 때, 위 소스 코드에서 수정할 부분은

그 데이터 객체의 클래스명인 ListItem을 다른 클래스명으로 다 바꿔주어야하고,

줄32~35의 코드만 수정하면 대부분 충분할 것이다. 나머지 코드는 수정 없이 그대로 사용할 수 있을 것이다.

그래서 지금은 줄32~35 부분만 자세히 살펴보자.

(줄32) 레이아웃 리소스 파일에서 만들었던, id가 textView1인 객체에 대한 참조를 얻는다.

(줄33) 레이아웃 리소스 파일에서 만들었던, id가 textView2인 객체에 대한 참조를 얻는다.

(줄34) textView1에 문자열을 채운다.

(줄35) textView2에 문자열을 채운다.

(줄32,33) findViewById 메소드 호출을 view.findViewById(..) 형태로 했음을 주목하자.

지금까지 findViewById 메소드를 호출할 때는 this.findViewById(..) 형태였다.

이 이유를 간단히 설명하자면,

(줄28)의 inflate 메소드 호출에 의해서 RelativeLayout 객체와 그 내부의 TextView 객체 두개가 자동으로 만들어진다. inflate 메소드의 첫째 파라미터인 resource 변수 값은 list\_item.xml 파일의 id인 R.layout.list\_item 값이 들어있을 것이다. 그래서 앞에서 만든 list\_item.xml 레이아웃 파일에서 지정한 형태대로 뷰 객체들이 만들어진다. 그리고 만들어진 뷰 객체들의 루트(root)인 RelativeLayout 객체에 대한 참조가 리턴된다. 이 참조를 지역변수 view에 저장한다.

(줄32,33)에서 view의 findViewById 메소드를 호출한다. 여기서 view는 줄28에서 생성된 RelativeLayout 객체를 참조하고 있다. findViewById 메소드는 이 RelativeLayout 객체 아래에서만 찾는다.

액티비티의 onCreate 메소드 내부에서 this.findViewById(...)의 경우 this는 액티비티이다.

액티비티 내부에서 뷰를 찾으라는 메소드 호출이다.

## MainActivity.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  50  51  52  53  54  55  56  57  58  59  60 | package com.example.e02list;  import java.util.ArrayList;  import android.support.v7.app.ActionBarActivity;  import android.os.Bundle;  import android.view.Menu;  import android.view.MenuItem;  import android.view.View;  import android.view.View.OnClickListener;  import android.widget.Button;  import android.widget.EditText;  import android.widget.ListView;  public class MainActivity extends ActionBarActivity {    ArrayList<ListItem> arrayList;  ListItemArrayAdapter adapter;    @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main);    arrayList = new ArrayList<ListItem>();  arrayList.add(new ListItem("One"));  arrayList.add(new ListItem("Two"));  adapter = new ListItemArrayAdapter(this, R.layout.list\_item, arrayList);  ListView listView = (ListView)findViewById(R.id.listView1);  listView.setAdapter(adapter);    Button b = (Button)findViewById(R.id.button1);  b.setOnClickListener(new OnClickListener() {  @Override  public void onClick(View arg0) {  EditText e = (EditText)findViewById(R.id.editText1);  CharSequence s = e.getText();  e.setText("");  arrayList.add(new ListItem(s.toString()));  adapter.notifyDataSetChanged();  }  });  }  @Override  public boolean onCreateOptionsMenu(Menu menu) {  getMenuInflater().inflate(R.menu.main, menu);  return true;  }  @Override  public boolean onOptionsItemSelected(MenuItem item) {  int id = item.getItemId();  if (id == R.id.action\_settings) {  return true;  }  return super.onOptionsItemSelected(item);  }  } |

#2예제의 소스에서 수정된 부분만 형광색으로 표시하였다.

ArrayAdapter<ListItem> 클래스 대신 방금 우리가 생성한 ListItemArrayAdapter 클래스로 바뀌었다.

그리고 레이아웃 리소스 파일 id가 android.R.layout.simple\_list\_item\_1 에서 R.layout.list\_item으로 바뀌었다.

## 실행

