

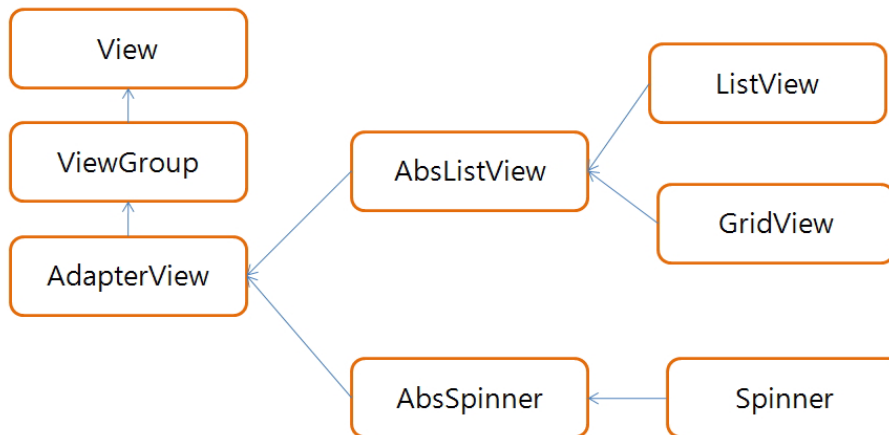
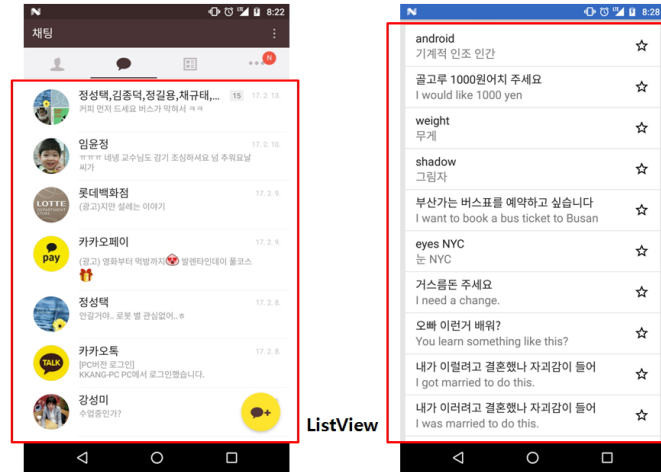


10장. AdapterView 활용

10.1 Adapter와 AdapterView

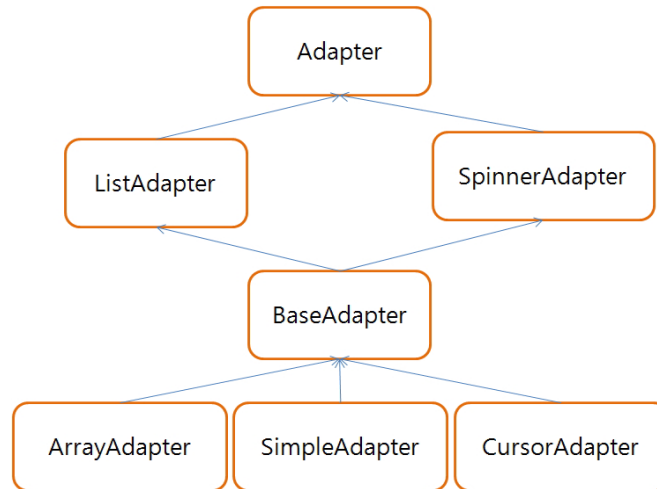
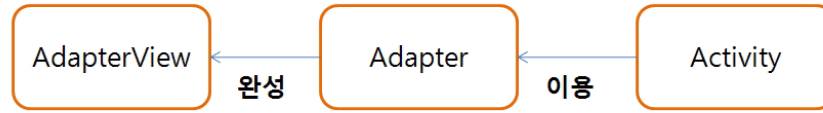
10.1.1. AdapterView의 구조

- AdapterView는 항목을 나열하는 뷰



10.1 Adapter와 AdapterView

- AdapterView에 항목이 나열되어 화 면에 데이터가 나오려면 꼭 Adapter라는 클래스를 이용
- Adapter에게 일을 시 키고 Adapter가 AdapterView를 완성해주는 구조



10.1 Adapter와 AdapterView

10.1.2. 라이브러리의 Adapter

ArrayAdapter

- 항목에 문자열 데이터를 순서대로 하나씩 나열

```
<ListView  
    android:id="@+id/main_list"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
/>
```

```
String[] datas=getResources().getStringArray(R.array.location);  
ArrayAdapter<String> adapter=new ArrayAdapter<String>(this,  
    android.R.layout.simple_list_item_1, datas);  
listView.setAdapter(adapter);
```

- this: Context 객체
- android.R.layout.simple_list_item_1: 항목 하나를 구성하기 위한 레이아웃 XML 파일 정보
- datas: 항목을 구성하는 데이터

ListView를 위해 제공되는 라이브러리의 XML 파일

- simple_list_item_1: 항목에 문자열 데이터 하나
- simple_list_item_2: 항목에 문자열 데이터 두 개 위아래 나열
- simple_list_item_multiple_choice: 문자열과 오른쪽 체크박스 제공
- simple_list_item_single_choice: 문자열과 오른쪽 라디오버튼 제공

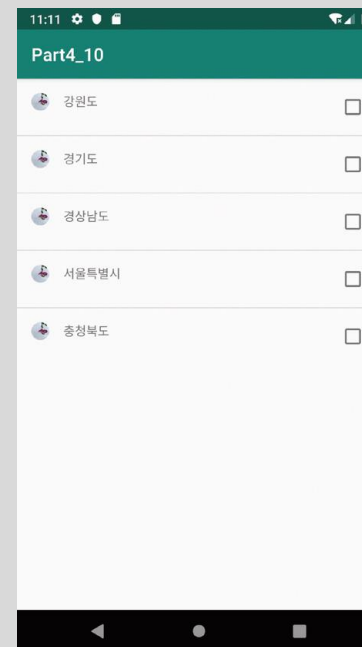


10.1 Adapter와 AdapterView

- 개발자가 항목 레이아웃 XML을 직접 만들어 적용

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">
    <ImageView
        android:id="@+id/main_item_icon"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/icon"
        android:maxWidth="20dp"
        android:maxHeight="20dp"
        android:adjustViewBounds="true"/>
    <TextView
        android:id="@+id/main_item_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@id/main_item_icon"
        android:layout_marginLeft="16dp"/>
    <CheckBox
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"/>

</RelativeLayout>
```



10.1 Adapter와 AdapterView

- ArrayAdapter에게 데이터를 출력할 TextView가 어떤 뷰인지 id값 지정

```
String[] datas=getResources().getStringArray(R.array.location);
ArrayAdapter<String> adapter=new ArrayAdapter<String>(this,
    R.layout.main_item, R.id.main_item_name, datas);
listView.setAdapter(adapter);
```

- 항목 선택 이벤트 처리

```
public class MainActivity extends AppCompatActivity implements AdapterView.OnItemClickListener{
    String[] datas;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        //..... listView.setOnItemClickListener(this);
    }

    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        Toast t=Toast.makeText(this, datas[position], Toast.LENGTH_SHORT);
        t.show();
    }
}
```

10.1 Adapter와 AdapterView

- 항목이 동적으로 추가 또는 제거

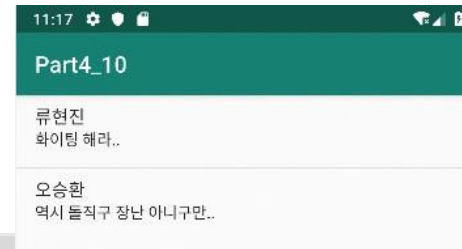
```
adapter.notifyDataSetChanged();
```

SimpleAdapter

- 항목에 문자열 데이터를 여러 개 나열
- SimpleAdapter에 항목을 구성 하기 위한 데이터

```
ArrayList<HashMap<String, String>> datas=new ArrayList<>();  
HashMap<String, String> map=new HashMap<>();  
map.put("name","류현진");  
map.put("content","제발 좀 MLB에서 봤으면 좋겠어..");  
datas.add(map);  
map=new HashMap<>();  
map.put("name","오승환");  
map.put("content","역시 돌직구 장난 아니구만..");  
datas.add(map);
```

```
SimpleAdapter adapter=new SimpleAdapter(this,      datas, android.R.layout.simple_list_item_2,  
    new String[]{"name","content"},      new int[]{android.R.id.text1, android.R.id.text2});  
listView.setAdapter(adapter);
```



10.1 Adapter와 AdapterView

SimpleAdapter의 생성자

- this: Context 객체
- datas: 항목 구성을 위한 데이터. ArrayList<HashMap<String,String>>
- android.R.layout.simple_list_item_2: 한 항목을 위한 레이아웃 XML
- new String[]{"name","content"}: 한 항목의 데이터를 가지는 HashMap에서 데이터를 추출하기 위한 키 값
- new int[]{android.R.id.text1, android.R.id.text2}: 추출된 데이터를 화면에 출력하기 위한 레이아웃 파일 내의 뷰 id 값

CursorAdapter

DBMS 프로그램의 select 결과값을 그대로 이용해 항목을 구성

```
CursorAdapter adapter=new SimpleCursorAdapter(this, android.R.layout.simple_list_item_2, cursor, new
String[]{"name","content"}, new int[]{android.R.id.text1, android.R.id.text2},
CursorAdapter.FLAG_REGISTER_CONTENT_OBSERVER);
listView.setAdapter(adapter);
```

- this: Context 객체
- android.R.layout.simple_list_item_2, cursor: 항목 구성 레이아웃 XML
- new String[]{"name","content"}: 데이터 추출 시 이용할 칼럼명
- new int[]{android.R.id.text1, android.R.id.text2}: 뷰의 id 값
- CursorAdapter.FLAG_REGISTER_CONTENT_OBSERVER: 플래그

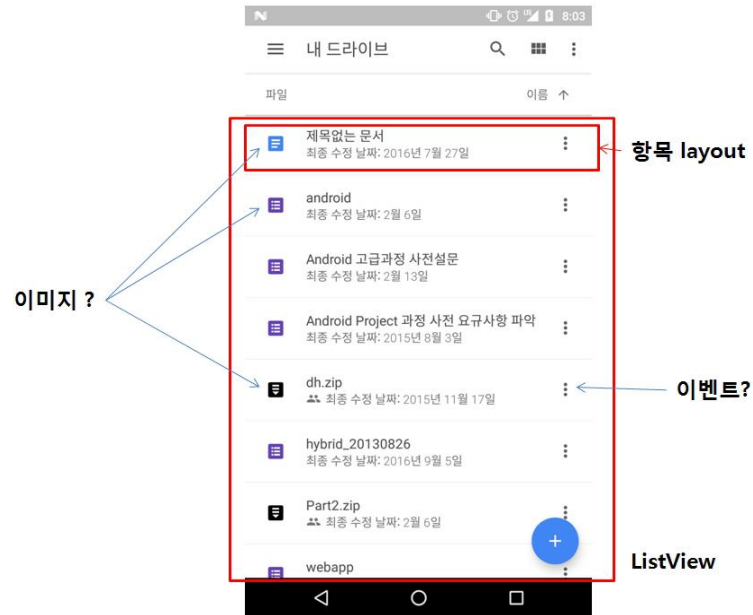
Step by Step 10-1 - Adapter

- 모듈 생성
- 파일 복사
- activity_main.xml 파일 작성
- MainActivity 작성
- 실행



10.2 커스텀 Adapter

10.2.1. 커스텀 Adapter가 필요한 예



- 개발자 알고리즘대로 항목의 데이터가 설정되어야 할 때
- 개발자 알고리즘대로 항목별 뷰의 이벤트를 다르게 처리해야 할 때
- 개발자 알고리즘대로 항목별 레이아웃을 다르게 적용해야 할 때

10.2 커스텀 Adapter

10.2.2. 커스텀 Adapter 작성 방법

```
public class DriveVO {  
    public String type;  
    public String title;  
    public String date;  
}
```

```
public class DriveAdapter extends ArrayAdapter<DriveVO>{  
    Context context;  
    int resId;  
    ArrayList<DriveVO> datas;  
  
    public DriveAdapter(Context context, int resId, ArrayList<DriveVO> datas){  
        //...  
    }  
  
    @Override  
    public int getCount() {  
        //...  
    }  
  
    @NonNull  
    @Override  
    public View getView(int position, View convertView, ViewGroup parent) {  
        //...  
    }  
}
```

10.2 커스텀 Adapter

- getCount() 함수는 전체 항목의 개수를 판단

```
public int getCount() {  
    return datas.size();  
}
```

- getView() 함수는 한 항목을 구성하기 위해 자동 호출

```
public View getView(int position, View convertView, ViewGroup parent) {  
    LayoutInflater inflater=(LayoutInflater)context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);  
    convertView=inflater.inflate(resId, null);  
    ImageView typeImageView=convertView.findViewById(R.id.custom_item_type_image);  
    //.....  
    final DriveVO vo=datas.get(position);  
    titleView.setText(vo.title);  
    dateView.setText(vo.date);  
    if(vo.type.equals("doc")){  
        typeImageView.setImageDrawable(ResourcesCompat.getDrawable(context.getResources(),  
R.drawable.ic_type_doc, null));  
    }  
    //.....  
    menuImageView.setOnClickListener(new View.OnClickListener(){  
        @Override  
        public void onClick(View v) {  
            Toast toast=Toast.makeText(context, vo.title+" menu click",Toast.LENGTH_SHORT);  
            toast.show();  
        }  
    });  
    return convertView;  
}
```

10.2 커스텀 Adapter

10.2.3. 커스텀 Adapter 추가 고려 사항

- 레이아웃 초기화 성능 이슈 : LayoutInflater
- 레이아웃 초기화를 최초에 한 번만 수행

```
public View getView(int position, View convertView, ViewGroup parent) {  
    if(convertView==null){  
        LayoutInflater inflater=(LayoutInflater)context.getSystemService (Context.LAYOUT_INFLATER_SERVICE);  
        convertView=inflater.inflate(resId, null);  
        DriveHolder holder=new DriveHolder(convertView);  
        convertView.setTag(holder);  
    }  
    //...  
    return convertView;  
}
```

- 뷰 획득 시 성능 이슈 : findViewById
- 획득한 뷰를 저장했 다가 그다음 이용 시 findViewById() 함수를 호출하지 않고 저장된 뷰를 그대로 이용

10.2 커스텀 Adapter

```
public class DriveHolder {  
    public ImageView typeImageView;  
    public TextView titleView;  
    public TextView dateView;  
    public ImageView menuImageView;  
  
    public DriveHolder(View root){  
        typeImageView=root.findViewById(R.id.custom_item_type_image);  
        titleView=root.findViewById(R.id.custom_item_title);  
        dateView=root.findViewById(R.id.custom_item_date);  
        menuImageView=root.findViewById(R.id.custom_item_menu);  
    }  
}
```

- Holder를 Adapter에서 메모리에 유지

```
DriveHolder holder=new DriveHolder(convertView);  
convertView.setTag(holder);
```

- 저장한 객체를 다시 획득해서 사용

```
DriveHolder holder=(DriveHolder)convertView.getTag();
```


Step by Step 10-2 – Custom Adapter

- 액티비티 생성
- 파일 복사
- DriveVO 클래스 작성
- DriveHolder 클래스 작성
- DriveAdapter 클래스 작성
- Lab10_2Activity 작성
- 실행

