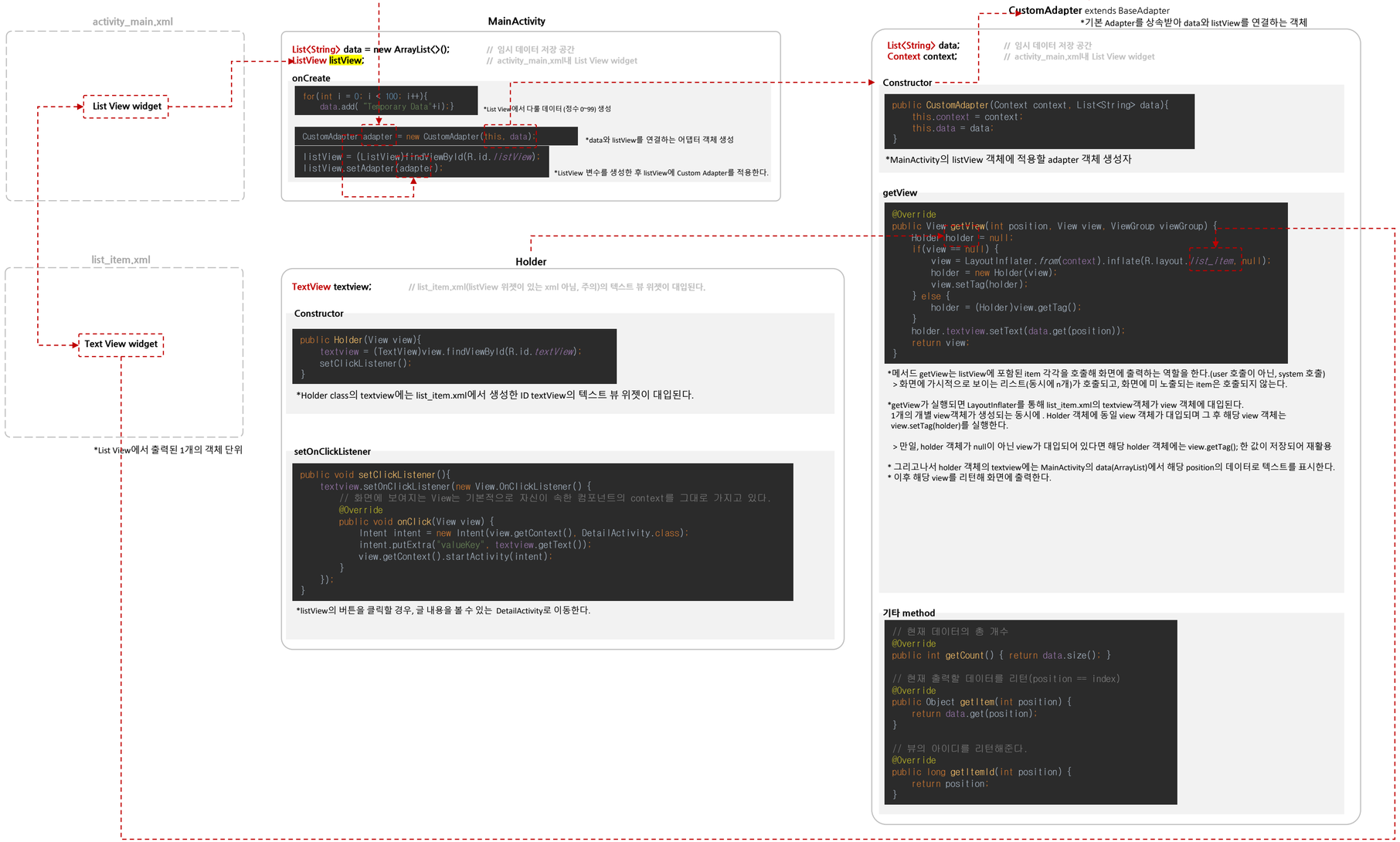
activity_main.xml
List View widget
list_item.xml
Text View widget
*List View에서 출력된 1개의 객체 단위
MainActivity
List<String> data = new ArrayList<>(); // 임시 데이터 저장 공간
ListView listView; // activity_main.xml내 List View widget
onCreate
for(int i = 0; i < 100; i++){ data.add("Temporary Data"+i);}
*List View에서 다룰 데이터(정수 0~99) 생성
CustomAdapter adapter = new CustomAdapter(this, data);
*data와 listView를 연결하는 어댑터 객체 생성
listView = (ListView)findViewById(R.id.listView);
listView.setAdapter(adapter);
*ListView 변수를 생성한 후 listView에 Custom Adapter를 적용한다.
Holder
TextView textview; // list_item.xml(listView 위젯이 있는 xml 아님, 주의)의 텍스트 뷰 위젯이 대입된다.
Constructor
public Holder(View view){ textview = (TextView)view.findViewById(R.id.textView); setClickListener(); }
*Holder class의 textview에는 list_item.xml에서 생성한 ID textView의 텍스트 뷰 위젯이 대입된다.
setOnClickListener
public void setClickListener(){ textview.setOnClickListener(new View.OnClickListener() { // 화면에 보여지는 View는 기본적으로 자신이 속한 컴포넌트의 context를 그대로 가지고 있다. @Override public void onClick(View view) { Intent intent = new Intent(view.getContext(), DetailActivity.class); intent.putExtra("valueKey", textview.getText()); view.getContext().startActivity(intent); } }); }
*listView의 버튼을 클릭할 경우, 글 내용을 볼 수 있는 DetailActivity로 이동한다.
CustomAdapter extends BaseAdapter
*기본 Adapter를 상속받아 data와 listView를 연결하는 객체
List<String> data; // 임시 데이터 저장 공간
Context context; // activity_main.xml내 List View widget
Constructor
public CustomAdapter(Context context, List<String> data){ this.context = context; this.data = data; }
*MainActivity의 listView 객체에 적용할 adapter 객체 생성자
getView
@Override public View getView(int position, View view, ViewGroup viewGroup) { Holder holder = null; if(view == null) { view = LayoutInflater.from(context).inflate(R.layout.list_item, null); holder = new Holder(view); view.setTag(holder); } else { holder = (Holder)view.getTag(); } holder.textview.setText(data.get(position)); return view; }
*메서드 getView는 listView에 포함된 item 각각을 호출해 화면에 출력하는 역할을 한다.(user 호출이 아닌, system 호출)
> 화면에 가시적으로 보이는 리스트(동시에 n개)가 호출되고, 화면에 미 노출되는 item은 호출되지 않는다.
*getView가 실행되면 LayoutInflater를 통해 list_item.xml의 textview객체가 view 객체에 대입된다.
1개의 개별 view객체가 생성되는 동시에 , Holder 객체에 동일 view 객체가 대입되며 그 후 해당 view 객체는 view.setTag(holder)를 실행한다.
> 만일, holder 객체가 null이 아닌 view가 대입되어 있다면 해당 holder 객체에는 view.getTag(); 한 값이 저장되어 재활용
* 그리고나서 holder 객체의 textview에는 MainActivity의 data(ArrayList)에서 해당 position의 데이터로 텍스트를 표시한다.
* 이후 해당 view를 리턴해 화면에 출력한다.
기타 method
// 현재 데이터의 총 개수 @Override public int getCount() { return data.size(); }
// 현재 출력할 데이터를 리턴(position == index) @Override public Object getItem(int position) { return data.get(position); }
// 뷰의 아이디를 리턴해준다. @Override public long getItemId(int position) { return position; }