개발 문서

사전 지식

컴포넌트 : 4개의 컴포넌트 (액티비티, 서비스, 콘텐츠 프로바이더, 브로드캐스트 리시버)

액티비티	사용자 UI 구성
서비스	데몬/백그라운드 수행
콘텐츠 프로바이더	애플리케이션 서로간 데이터 공유를 위함
브로드캐스트 리시버	이벤트 알람

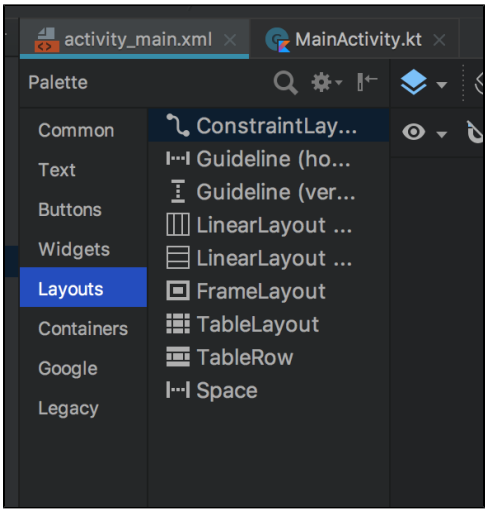
가장 많이 사용되고 많이 쓸 수 밖에 없는 컴포넌트는 액티비티! 결국 액티비티가 없는 앱은 스터디가 원하는 앱이 아니다.

애플리케이션을 실행했을때 애플리케이션 화면 전체가 액티비티로 보면 된다. 액티비티내에 각 뷰 컴포넌트가 배치된다.

먼저 게시판을 떠올리면 "목록 액티비티" , "상세 액티비티", "수정 액티비티"등등이 모두 별도의 액티비티가 된다.

액티비티 화면은 코드(Java)와 리소스(레이아웃 XML)로 구성할 수 있다.

Layout 종류



가장 많이 쓰는 레이아웃으로 게시판 앱에서도 해당 레이아웃 하나면 될 듯 한데..

2016년 발표 이후로는 ConstraintLayout 를 많이 사용하고 또, 초기 프로젝트 생성시 기본 레이아웃으로 지정되어 있다.

https://academy.realm.io/kr/posts/constraintlayout-it-can-do-what-now/

View

액티비티에 화면 구성을 위한 작은단위의 콤포넌트.

종류로는 Button, ImageView, ListView, TextView..등이 있다.

액티비티에 ViewGroup이 계층구조로 존재할 수 있다.

AdapterView

View에 데이터를 나열하기위한 View? ListView등에 데이터를 나열할 수 있도록 해준다.

게시판 리스트

기본 xml

activity_main.xml

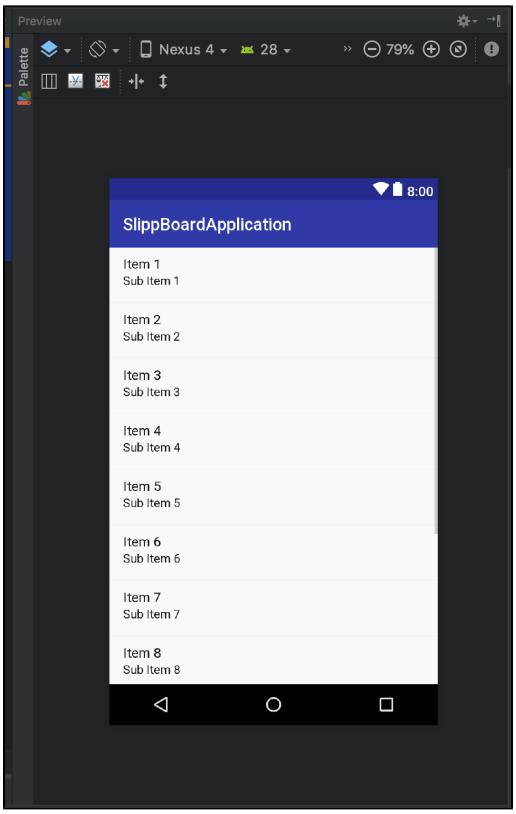
```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ListView
        android:id="@+id/listView"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
    </ListView>

</android.support.constraint.ConstraintLayout>
```

List 내 반복될 ViewGroup

board_row.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">


    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:orientation="vertical">
        <TextView
            android:id="@+id/txtName"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Name"
            android:textStyle="bold"
            />
        <TextView
            android:id="@+id/txtTitle"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Title"/>
    </LinearLayout>
</RelativeLayout>
```

Preview



Sources

우선 리스트에서 각 Row로 채울 DTO를 간단하게 만들어 본다.

```
data class Board (var title:String, var userName:String)
```

Kotlin에서는 꼭 파일 명과 클래스 명을 같게 만들 필요는 없지만 여지껏 해오던 대로 Board.kt 로 파일명을 정한다.

해당 클래스는 단순한 property의 집합이므로 data class로 생성한다. (주의할 점은 val가 아니라 var이다. 누군가 올려주신 브런치 https://brunch.co.kr/@mystoryg/8)

이제 ListView에 데이터를 넘기기 위한 Adapter를 만들어야 한다. 기존 Adapter로는 우리가 원하는 형태의 데이터를 보낼 수 없으니 Custom Adapter를 만들어야 한다.

누군가 올려주신 티스토리들

http://androidhuman.tistory.com/205

https://medium.com/@jsuch2362/adapter-%EB%88%84%EA%B5%AC%EB%83%90-%EB%84%8C-data-view-2db7eff11c20

```
// Kotlin    extends BaseAdapter
class CustomListAdapter(private var activity: Activity, private var items: ArrayList<Board>) : BaseAdapter() {
...
}
```

BaseAdapter를 상속하면 기본으로 구현해야 하는 메소드가 있다.

```
override fun getView(position: Int, convertView: View?, parent: ViewGroup): View {
    ..
}
override fun getItem(i: Int): Board {
    ..
}

override fun getItemId(i: Int): Long {
    ..
}

override fun getCount(): Int {
    ..
}
```

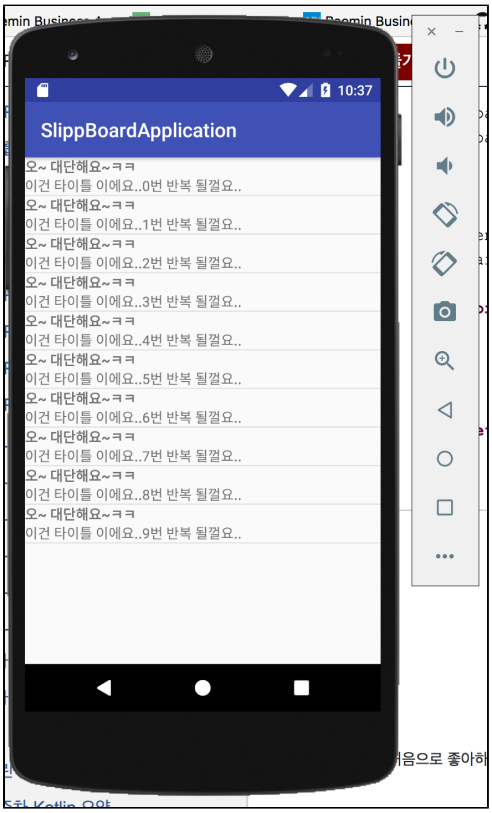
전체소스

```
class CustomBoardListAdapter(private var activity: Activity, private var items: ArrayList<Board>) :
BaseAdapter() {

    private class BoardViewHolder(view: View?) {
        var txtName: TextView? = null
        var txtTitle: TextView? = null
        init {
            this.txtName = view?.findViewById<TextView>(R.id.txtName)
            this.txtTitle = view?.findViewById<TextView>(R.id.txtTitle)
        }
    }

    override fun getView(position: Int, convertView: View?, parent: ViewGroup): View {
        val pair = initializeViewAndViewHolder(convertView)
        addRowData(position, pair.second)
        return pair.first as View
    }

    private fun initializeViewAndViewHolder(convertView: View?): Pair<View?, BoardViewHolder> {
        if (convertView == null) {
            val inflater = activity?.getSystemService(Context.LAYOUT_INFLATER_SERVICE) as LayoutInflater
            var view = inflater.inflate(R.layout.board_row, null)
            var viewHolder = BoardViewHolder(view)
            view?.tag = viewHolder
            return Pair(view, viewHolder)
        }
        return Pair(convertView, convertView.tag as BoardViewHolder)
    }

    private fun addRowData(position: Int, viewHolder: BoardViewHolder) {
        var board = items[position]
        viewHolder.txtName?.text = board.userName
        viewHolder.txtTitle?.text = board.title
    }

    override fun getItem(i: Int): Board {
        return items[i]
    }

    override fun getItemId(i: Int): Long {
        return i.toLong()
    }

    override fun getCount(): Int {
        return items.size
    }

}
```

이제 MainActivity에서 실행 할 수 있도록

```
class MainActivity : AppCompatActivity() {

    var boardListView: ListView? = null
    var BoardListViewAdapter: CustomBoardListAdapter? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        boardListView = findViewById<ListView>(R.id.listView)
        BoardListViewAdapter = CustomBoardListAdapter(this, generateData())

        boardListView?.adapter = BoardListViewAdapter
        BoardListViewAdapter?.notifyDataSetChanged()

    }

    fun generateData(): ArrayList<Board> {
        var result = ArrayList<Board>()

        for (i in 0..9) {
            var user: Board = Board("  ..10  ..", "~ ~")
            result.add(user)
        }

        return result
    }
}
```

최종결과



이제 항목에 클릭 이벤트를 넣어보자.

기본적으로 게시판 목록에서 Row클릭시 이벤트가 발생하는 것이니 해당 이벤트의 리스너는 ListView에 적용될 것이고, 기타 인자로 처리되어야 하는 속성이 position이나 id, 부모등이 있을 것이다.

해서 listview에서 어떤 메소드가 있는지 찾아보면 onItemClickListener 가 뚜시궁 하고 나올것이다.

코틀린에 맞게 onItemClickListener를 구현하고 Toast를 발생시켜 보자.

```
boardListView!!.onItemClickListener = AdapterView.OnItemClickListener { parent, view, position, id ->
    toast("." + position)
}
```

이제 목록에서 하나의 row를 클릭하면 Toast가 보일 것이다.

참고 유투브 :

새로운 Activity(상세) 띄워보기

클릭까지 만들었으니 이제 상세용 Activity를 만들고 실행하도록 해보자.