Mobile Application Development

By,

Prof. Himanshu H Patel,

Prof. Hiten M Sadani

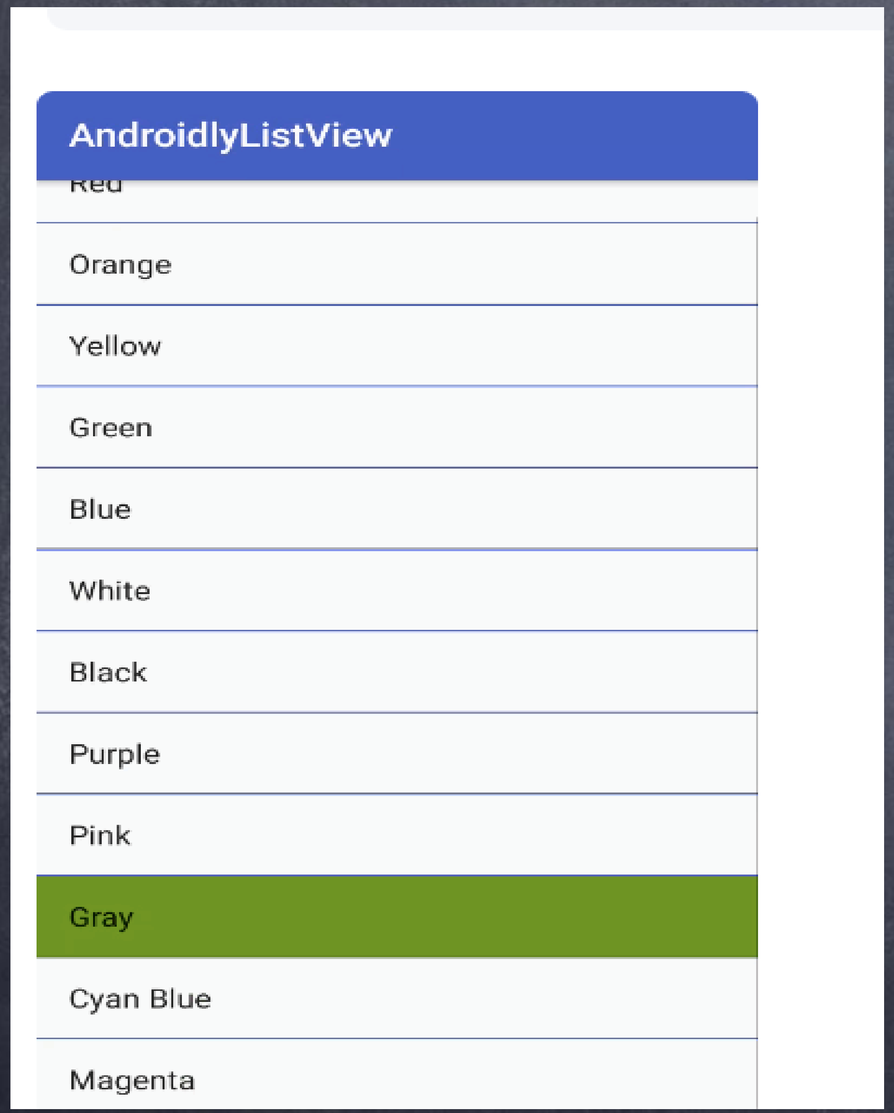
U. V. Patel College of Engineering, Ganpat University

ListView

ListView

- ***What is List view***
- ***How to Implement the ListView.***

ListView


AndroidlyListView
Red
Orange
Yellow
Green
Blue
White
Black
Purple
Pink
Gray
Cyan Blue
Magenta


2:22
ListView_Zoo
Baboon
Baboons are primates comprising the genus Papio, one of the 23 genera of Old World monkeys
Bulldog
Baboons are primates comprising the genus Papio, one of the 23 genera of Old World monkeys
Panda
Baboons are primates comprising the genus Papio, one of the 23 genera of Old World monkeys
Zebra
Baboons are primates comprising the genus Papio, one of the 23 genera of Old World monkeys
White tiger
Baboons are primates comprising the genus Papio, one of the 23 genera of Old World monkeys
swallow_bird

ListView

- ***What is List view?***

- ListView is a very common UI element in Android Applications. It is used to display a list of items separated by dividers that can be scrolled endlessly. It's generally used to display a group of related items.

- ***A list view is an adapter view that does not know the details, such as type and contents, of the views it contains. Instead list view requests views on demand from a ListAdapter as needed, such as to display new views as the user scrolls up or down.***

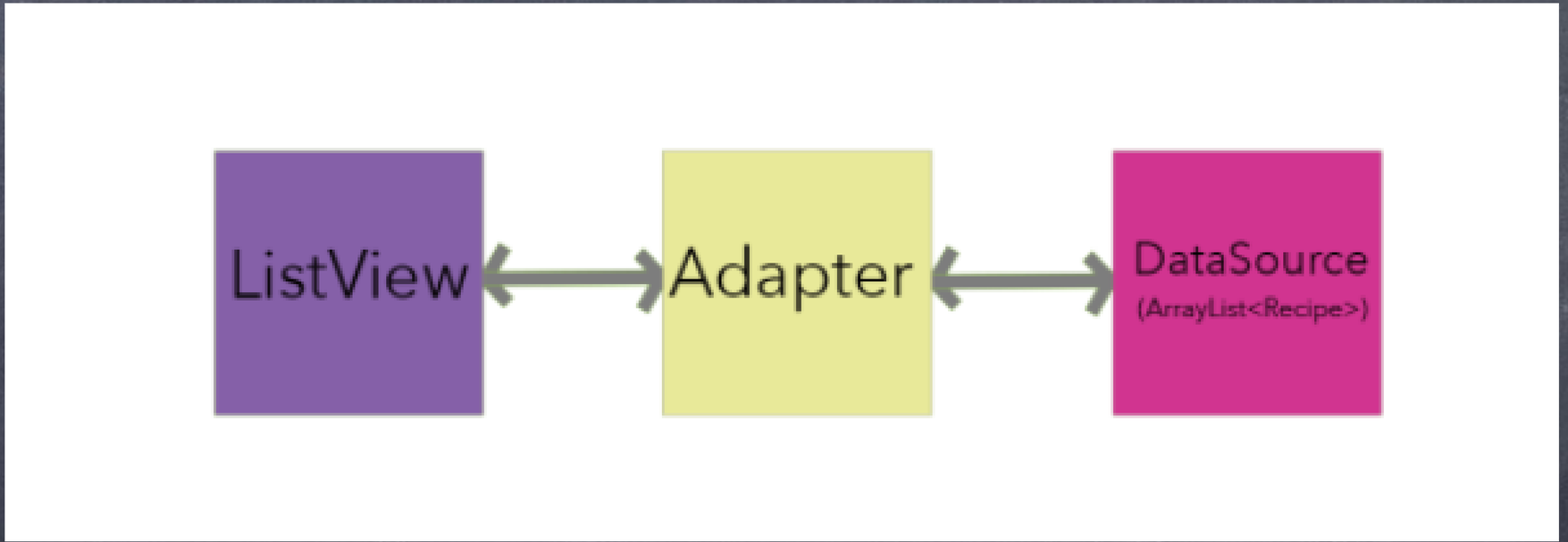
ListView

To implement a ListView we have to create the following:

- *of course the ListView itself which we should add to our screen layout.*
- *a layout for each row in the list.*
- *an adapter which holds the data and bind them to the list.*

Adapters: Servants of the ListView

What Exactly is an Adapter?



ListView

- *How to Implement it:*
- *ListView XML Layout*
- *<ListView*
  android:id="@+id/recipe_list_view"
  android:layout_width="match_parent"
  android:layout_height="wrap_content />

List View XML attributes

- *android:divider : Drawable or color to draw between list items.*
- *android:dividerHeight : Height of the divider.*
- *android:entries : An array resource can be passed here to be displayed as a List.*
- *android:footerDividersEnabled : When set to false, there will be no divider at the end of the ListView.*
- *android:headerDividersEnabled : When set to false, there will be no divider at the top of the ListView.*
- *android:clickable: This makes the ListView rows clickable.*
- *android:listSelector: Set the background color of the list view row when it is clicked.*

String.xml

```
<string-array name="Colors">
        <item name="color">Red</item>
        <item name="color">Orange</item>
        <item name="color">Yellow</item>
        <item name="color">Green</item>
        <item name="color">Blue</item>
        <item name="color">White</item>
        <item name="color">Black</item>
        <item name="color">Purple</item>
        <item name="color">Pink</item>
        <item name="color">Gray</item>
        <item name="color">Cyan Blue</item>
        <item name="color">Magenta</item>
    </string-array>
```

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:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:entries="@array/Colors"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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

ListView Adapter

A ListView class by itself cannot populate the entries. An Adapter is responsible for populating the data in the ListView. We have built-in adapter classes(like the one above) which come with a built-in layout for each row

getView() : We can inflate our own layouts in the Adapter inside this method.

notifyDataSetChanged() method on the adapter is called if the data has changed or if new data is available.

Types of ListView Adapters

There are four main types of Adapters:

1.BaseAdapter – As the name suggests this abstract is extended by all the other adapters. When creating a custom adapter using this as the parent class, you need to override all the methods mentioned above along with getCount(), getId() etc.

2.ArrayAdapter – This populates the ListView with the array supplied. It is defined as:

```
var arrayAdapter = ArrayAdapter<String>(context,layout,array);
```

Types of ListView Adapters

3.ListAdapter – Unlike an ArrayAdapter, this is an interface. So it can be used only with concrete adapter classes. Concrete Adapter classes are ListActivity and ListFragment.

4.SimpleCursorAdapter – This is used when the data needs to populated from a Database. In its constructor, we must specify the layout for each row and also the Cursor instance which contains the fields that need to be displayed.

ListView

- ***How to Implement it:***

```
Class MainActivity : AppCompatActivity() {

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

        // use arrayadapter and define an array
        val arrayAdapter: ArrayAdapter<*>

        val users = arrayOf( "Virat Kohli", "Rohit Sharma", "Steve Smith",
            "Kane Williamson", "Ross Taylor" )
```

ListView

```
// access the listView from xml file
        var mListView = findViewById<ListView>(R.id.userlist)
        arrayAdapter = ArrayAdapter(this,
            android.R.layout.simple_list_item_1, users)
        mListView.adapter = arrayAdapter
```

ListView

```
listView.setOnItemClickListener(){adapterView, view, p, id ->

        val itemAtPos = adapterView.getItemAtPosition(p)

        val itemIdAtPos = adapterView.getItemIdAtPosition(p)

        Toast.makeText(Context,Variable name that contain name,
Toast.LENGTH_LONG).show()
    }
```

Grid View

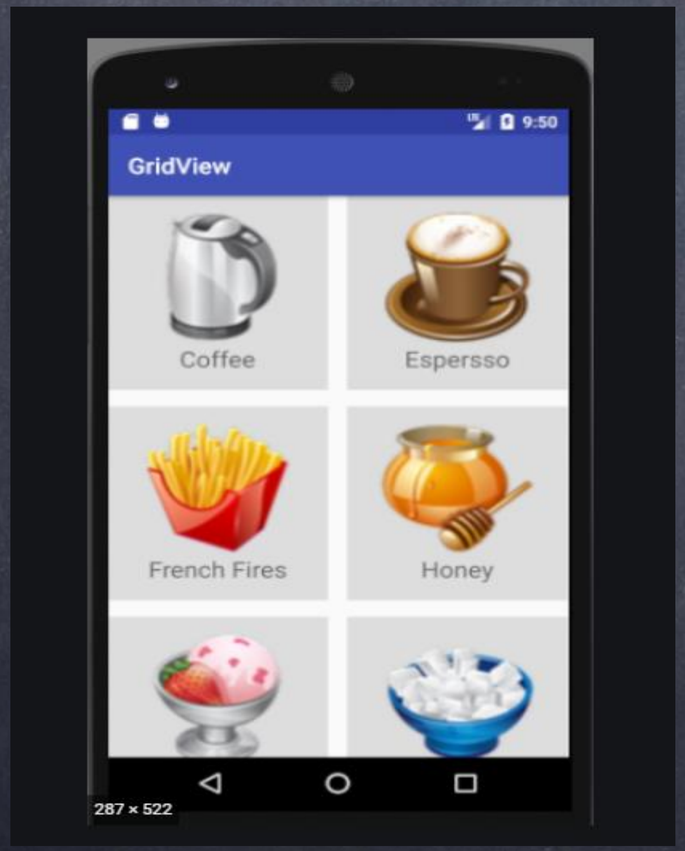
- ***What is Grid View***
- ***How to Implement the Grid view.***

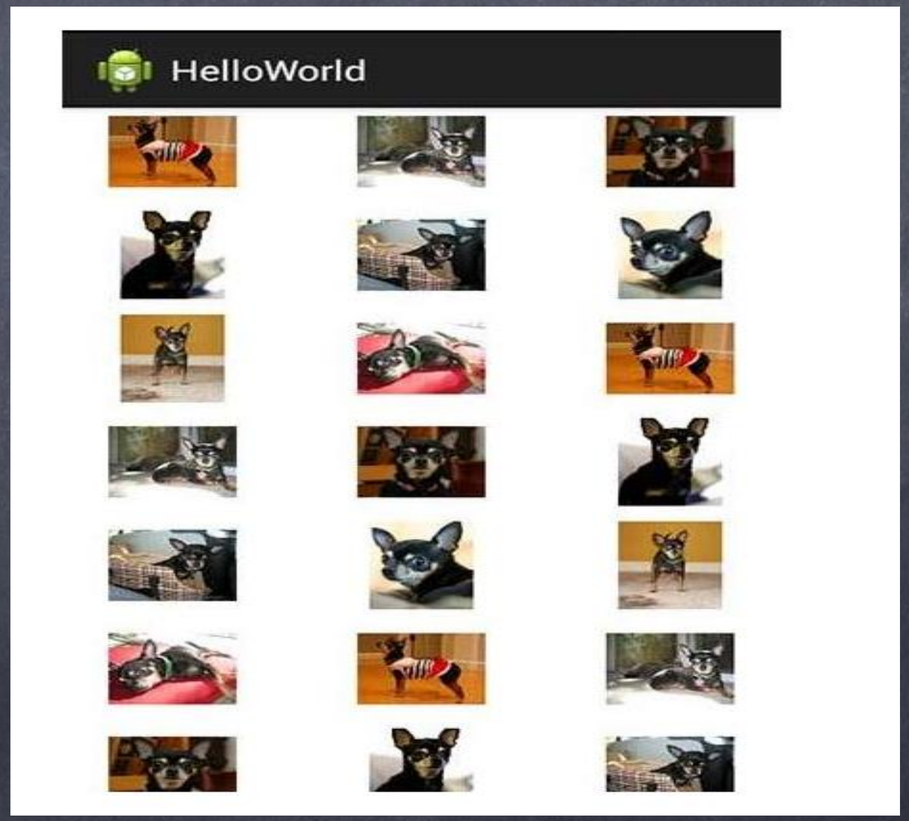
Grid View

- ***What is Grid View***
- ***How to Implement the Grid view.***

Grid View

- ***What is Grid View***
- ***How to Implement the Grid view.***





Grid View

- ***What is Grid View***
- Android GridView shows items in two-dimensional scrolling grid (rows & columns) and the grid items are not necessarily predetermined but they automatically inserted to the layout using a ListAdapter

Grid View

- ***What is Grid View***
- Android GridView shows items in two-dimensional scrolling grid (rows & columns) and the grid items are not necessarily predetermined but they automatically inserted to the layout using a ListAdapter

Grid View Attributes

Sr.No	Attribute & Description
1	**android:id** This is the ID which uniquely identifies the layout.
2	**android:columnWidth** This specifies the fixed width for each column. This could be in px, dp, sp, in, or mm.
3	**android:gravity** Specifies the gravity within each cell. Possible values are top, bottom, left, right, center, center_vertical, center_horizontal etc.
4	**android:horizontalSpacing** Defines the default horizontal spacing between columns. This could be in px, dp, sp, in, or mm.
5	**android:numColumns** Defines how many columns to show. May be an integer value, such as "100" or auto_fit which means display as many columns as possible to fill the available space.

Grid View Attributes

6	**android:stretchMode** Defines how columns should stretch to fill the available empty space, if any. This must be either of the values − ▫ none − Stretching is disabled. ▫ spacingWidth − The spacing between each column is stretched. ▫ columnWidth − Each column is stretched equally. ▫ spacingWidthUniform − The spacing between each column is uniformly stretched..
7	**android:verticalSpacing** Defines the default vertical spacing between rows. This could be in px, dp, sp, in, or mm.