Nikhil Bansal Follow
Associate Android Developer and Open Source Enthusiast
Jul 30, 2017 · 5 min read

Custom Array Adapters made Easy!

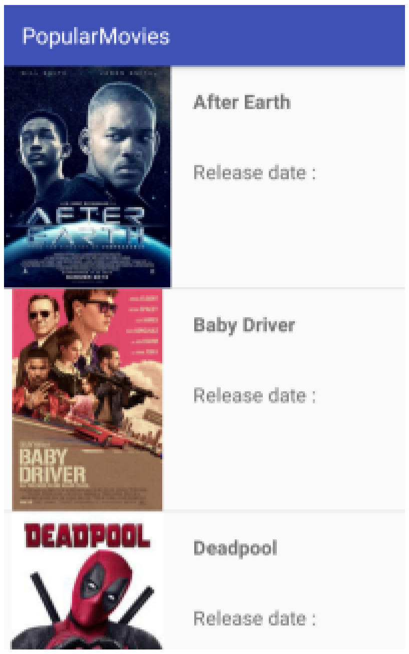
In the Field of Android Development, *Array Adapters* have always played an important role in populating and controlling the *ListViews* and *Spinners*. Whenever we need to show our data as a list to the user, Array *Adapters* always come handy to easily manage the behaviour of the *ListView* and their items. But does the default *Adapter class* help us to create our own list item that we want to show to the user?? I guess not..So I have created a sample project called Popular Movies in which we will create a custom array adapter which which will help us to add an Image and Text to the ListItem (For those who don't know, ListItem is an item of your total list whose layout will remain the same for every item and data will change).

Why Custom Array Adapter??

Android framework by default provides us the ability to create listItems which includes only a single information or a single *TextView*. But we always come across apps that show multiple information in a single ListItem such as Instagram, BookMyShow, Whatsapp and many more. Implementing your own custom array adapter can seem an intimidating job at first but once you create some more, you will get hold. As a beginner when I started with *ArrayAdapters* it was very difficult to hold on to the concepts which were used to correctly implement it but I made atleast 4–5 implementations just to make sure I understand what exactly was going on in the code. I would suggest that you should practice it because custom array adapters are very important when it comes to showing information in List.

Lets Start!

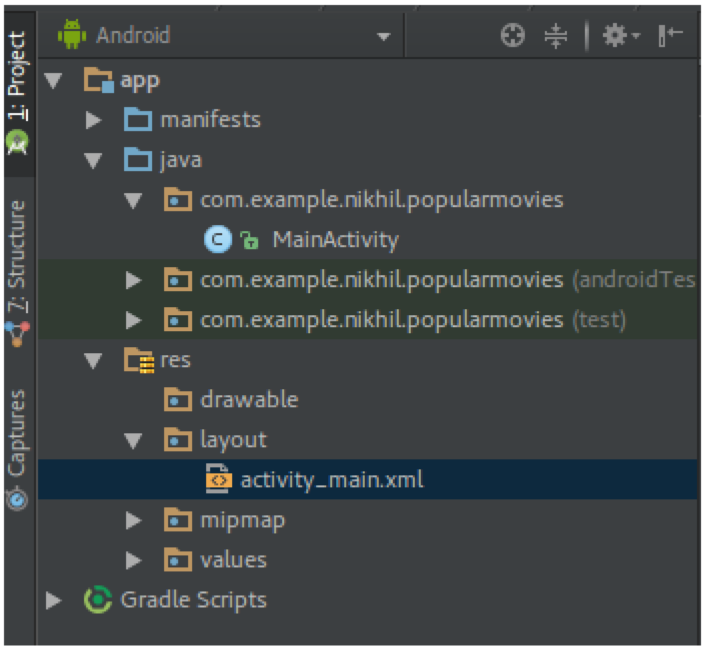
We am going to make a custom project which is called PopularMovies App in which we will list the information of some movies with their poster. This will be the final stage of the app :



So lets begin!

Create a layout xml file which will have the design of your ListItem.

All the files related to the layout of the app are contained in the layout folder of the res directory. So first navigate into the directory



Notice that this folder contains only single file i.e **activity_main.xml.** This file will contain the layout and design to your Main Activity which contains your ListView. We are going to make a ***list_item.xml*** file that contains the code to the basic structure and design of the list item of the List view. Here is a snippet of the list_item.xml file :

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

    <ImageView
        android:id="@+id/imageView_poster"
        android:layout_width="150dp"
        android:layout_height="200dp"/>

    <TextView
        android:id="@+id/textView_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@id/imageView_poster"
        android:layout_margin="20dp"
        android:textSize="20sp"
        android:textStyle="bold"/>

```

In this file we added a *ImageView* and 2 *TextViews to display the movie details.*

Create a Java class which will define your ListItem and store data for every Movie.

It is important to create a Java Class that will store the details for your movie. These classes are called pojos (plain old java objects). These are used to define a collection of related information which can be used to bind that information together and use it to show in Lists or store in databases. In a pojo you should define the type of information you are about to show in the list. For example, if I intent to show an image and textView so I will create variables that will store the image and the data. Pojo should contain *getter()* and *setter()* methods which are used to set values to the pojo object and get the data from the object. Here is a code snippet for a pojo class for Movie :

```java
package com.example.nikhil.popularmovies;

public class Movie {

    // Store the id of the  movie poster
    private int mImageDrawable;
    // Store the name of the movie
    private String mName;
    // Store the release date of the movie
    private String mRelease;

    // Constructor that is used to create an instance of th
    public Movie(int mImageDrawable, String mName, String m
        this.mImageDrawable = mImageDrawable;
        this.mName = mName;
        this.mRelease = mRelease;
    }

    public int getmImageDrawable() {
        return mImageDrawable;
    }

    public void setmImageDrawable(int mImageDrawable) {
        this.mImageDrawable = mImageDrawable;
    }

    public String getmName() {
```

It is important that your Java class should be declared public because you will need to create an instance of it in the MainActivity Class. The variables should be declared as private as there are cases in which you need to make variables with same names in different objects. This is the reason we need to create the getter and setter methods which are used to add and extract to and from the object.

Create a Java class which will extend the Default *ArrayAdapter* class.

Now is the time when you implement your own array adapter. Create a java class and give it any relative name. I named it MovieAdapter.class . Your class should extend ArrayAdapter<T> class where T should be replaced by the pojo class you just made. It is done to tell the adapter about what type of data the adapter has to display. The first step is to define a constructor. You can use any type of constructor mentioned in the documentation here. We will make a constructor which takes 2 arguments i.e the Context and the list of movies. The arguments received in the constructor should be saved in a private variable as they are used to inflate and add data into the view.

Next, you need to Override a method called *getView()* method. This view is called when a listItem needs to be created and populated with the data.In this method first the View is inflated using the LayoutInflator.inflate() method. It is important that you check that if the view you are trying to inflate is new or reused. If *convertView == null* then the view should be inflated. In this view, you should set the data into the views. First get the correct movie from the list of movies using the *get()* method on moviesList and passing position as the argument.

Now using the getter methods defined in the pojo class, store the information encapsulated in the object in different variables. This information is to be set in the view you just inflated so, *findViewById()* is called on the inflated view and the information is set. At last the view is returned. Here is code snippet of the *MovieAdapter.class* file :

```java
package com.example.nikhil.popularmovies;

import android.content.Context;
import android.support.annotation.LayoutRes;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;

import java.util.ArrayList;
import java.util.List;


public class MovieAdapter extends ArrayAdapter<Movie> {

    private Context mContext;
    private List<Movie> moviesList = new ArrayList<>();

    public MovieAdapter(@NonNull Context context, @LayoutRe
        super(context, 0 , list);
        mContext = context;
        moviesList = list;
    }

    @NonNull
    @Override
    public View getView(int position, @Nullable View conver
        View listItem = convertView;
```

Add a *ListView* in the appropriate layout where you want to show your list.

After creating the Custom Adapter you need a ListView which will show the adapter contents. So in the layout file of you Activity (in this case MainActivity) add a <*ListView*> tag. Here is code snipet of the *activity_main.xml* file :

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.nikhil.popularmovies.MainAct

    <ListView
        android:id="@+id/movies_list"
```

Create an instance of the Custom ArrayAdapter class which you just created and add it to the ListView

This is the final step. After the Custom Adapter is created and ListView is added to the layout file of the activity, an instance of the adapter is to be created and is set to the listView by calling *setAdapter()* method on the listView. Here is code snippet of the MainActivity.java file :

```java
package com.example.nikhil.popularmovies;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.widget.ListView;

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

    private ListView listView;
    private MovieAdapter mAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        listView = (ListView) findViewById(R.id.movies_list
        ArrayList<Movie> moviesList = new ArrayList<>();
        moviesList.add(new Movie(R.drawable.movie_after_ear
        moviesList.add(new Movie(R.drawable.movie_baby_driv
        moviesList.add(new Movie(R.drawable.movie_deadpool,
```

It is always a good practice to go through the documentation of the topic as you may find the best method for your use. You can now start by customizing the list item to the way you want to show your data and easily make a custom array adapter to control the flow of list.

Thank you for reading this far! If you found this useful , kindly show some love and share it! Stay tuned for more blogs…