Prerequisite: One must know "How to code a custom List View"?

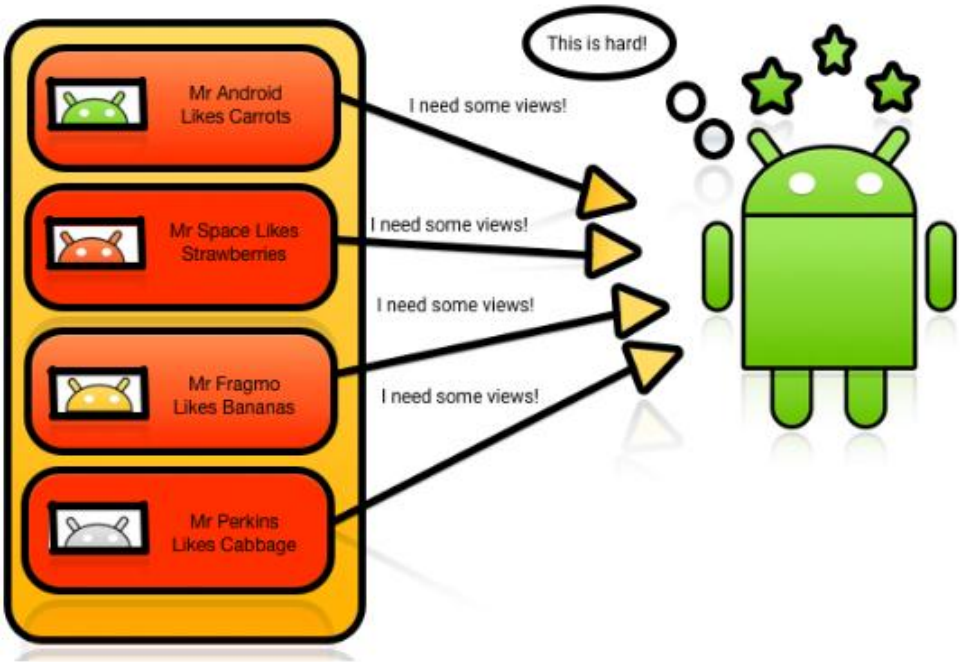
(One can refer *How to code a custom List View* document, on http://android.suven.net)

Why You Need RecyclerView?

Imagine you're creating a ListView where the custom items you want to show are quite complicated. You take time to lovingly create a row layout for these items, and then use that layout inside your adapter.

Inside your **getView()** method, you inflate your new item layout. You then reference every view within by using the unique ids you provided in your XML to customize and add some view logic. Once finished, you pass that view to the ListView, ready to be drawn on the screen.

The truth is that ListViews and GridViews only do half the job of achieving true memory efficiency. They recycle the item ***layout***, but don't keep references to the layout children, forcing you to call **findViewById()** for every child of your item layout every time you call **getView()**.
All this calling around can become *very* processor-intensive, especially for complicated layouts. Furthermore, the situation can cause your ListView scrolling to become jerky or non-responsive *at times on slow processor speeds mobiles*.



Android RecyclerView is more advanced version of ListView with improved performance and other benefits. Using RecyclerView and CardView together, both lists and grids can be created very easily. Here is the complete information about RecyclerView..

RecyclerView and Layouts

The arrival of the RecyclerView changed everything. It still uses an ***Adapter*** to act as a data source; however, you have to create ***ViewHolders*** to keep references in memory. When you need a new view, it either creates a new ViewHolder object to inflate the layout and hold those references, or it recycles one from the existing stack.

Now you know why it's called a RecyclerView!

Another advantage of using RecyclerViews is that they come with default animations.
Thanks to the requirement for a ViewHolder, the RecyclerView knows exactly which animation to apply to which item. Best of all, it just does it as required. You can even create your own animations and apply them as needed.

The last and most interesting component of a RecyclerView is its ***LayoutManager***. This object positions the RecyclerView's items and tells it when to recycle items that have transitioned off-screen. Layout Managers come in three default flavors:

- ***LinearLayoutManager*** positions your items to look like a standard ListView

- ***GridLayoutManager*** positions your items in a grid format similar to a GridView
- ***StaggerGridLayoutManager*** positions your items in a staggered grid format.

Steps to code a simple Recycler (List) View:

Below is the RecyclerView widget with necessary attributes.

```
<android.support.v7.widget.RecyclerView

        android:id="@+id/recycler_view"

        android:scrollbars="vertical"

        android:layout_width="match_parent"

        android:layout_height="wrap_content"/>
```

Before you start, make sure that you updated your Android Studio to latest version.

1. Creating New Project

1. In Android Studio, go to **File ⇒ New Project** and fill all the details required to create a new project. When it prompts to select a default activity, select **Blank Activity** and proceed.

2. Open **build.gradle** and add recycler view dependency. `com.android.support:recyclerview-v7:25.x.x` and rebuild the project.

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support:appcompat-v7:25.0.1'
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:design:25.0.1'
    compile 'com.android.support:recyclerview-v7:25.0.1'
}
```

3. in **activity_main.xml file:**

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout

    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:showIn="@layout/activity_main"
    tools:context=".MainActivity">

    <android.support.v7.widget.RecyclerView

        android:id="@+id/recycler_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:scrollbars="vertical" />

</RelativeLayout>
```

4. Open **colors.xml** located under **res ⇒ values** and add below colors.

```
<?xml version="1.0" encoding="utf-8"?>

<resources>

    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
    <color name="year">#999999</color>
    <color name="title">#222222</color>

</resources>
```

2. **Writing the Adapter Class**

After adding the RecyclerView widget, let's start writing the adapter class to render the data. The RecyclerView adapter is same as ListView but the override methods are different.

5. Create a class named **Movie.java** and declare title, genre and year. Also add the getter/setter methods to each variable.

```
package suvenconsultants.com.recyclerviewdemo;

public class Movie {
    private String title, genre, year;

    public Movie() {
    }

    public Movie(String title, String genre, String year) {
        this.title = title;
        this.genre = genre;
        this.year = year;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String name) {
        this.title = name;
    }

    public String getYear() {
        return year;
    }

    public void setYear(String year) {
        this.year = year;
    }

    public String getGenre() {
        return genre;
    }

    public void setGenre(String genre) {
        this.genre = genre;
    }
}
```

6. Create an layout xml named **movie_list_row.xml** with the below code. This layout file renders a single row in recycler view by displaying movie name, genre and year of release.

```
<?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="match_parent"
    android:focusable="true"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="10dp"
    android:paddingBottom="10dp"
    android:clickable="true"
    android:background="?android:attr/selectableItemBackground"
    android:orientation="vertical">

    <TextView
        android:id="@+id/title"
        android:textColor="@color/title"
        android:textSize="16dp"
        android:textStyle="bold"
        android:layout_alignParentTop="true"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <TextView
        android:id="@+id/genre"
        android:layout_below="@id/title"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <TextView
        android:id="@+id/year"
        android:textColor="@color/year"
        android:layout_width="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_height="wrap_content" />

</RelativeLayout>
```

7. Now create a class named **MoviesAdapter.java** and add the below code. Here **onCreateViewHolder()** method inflates **movie_list_row.xml**. In **onBindViewHolder()** method the appropriate movie data (title, genre and year) set to each row.

```
package suvenconsultants.com.recyclerviewdemo;

import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import java.util.List;

public class MoviesAdapter extends RecyclerView.Adapter<MoviesAdapter.MyViewHolder> {

    private List<Movie> moviesList;

    public class MyViewHolder extends RecyclerView.ViewHolder {
        public TextView title, year, genre;

        public MyViewHolder(View view) {
            super(view);
            title = (TextView) view.findViewById(R.id.title);
            genre = (TextView) view.findViewById(R.id.genre);
            year = (TextView) view.findViewById(R.id.year);
        }
    }


    public MoviesAdapter(List<Movie> moviesList) {
        this.moviesList = moviesList;
    }

    @Override
    public MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View itemView = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.movie_list_row, parent, false);

        return new MyViewHolder(itemView);
    }

    @Override
    public void onBindViewHolder(MyViewHolder holder, int position) {
        Movie movie = moviesList.get(position);
        holder.title.setText(movie.getTitle());
        holder.genre.setText(movie.getGenre());
```

```
        holder.year.setText(movie.getYear());
    }

    @Override
    public int getItemCount() {
        return moviesList.size();
    }
}
```

8. Now open **MainActivity.java** and do the below changes. Here **prepareMovieData()** method adds sample data to list view.

```
package suvenconsultants.com.recyclerviewdemo;

import android.content.Context;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.DefaultItemAnimator;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.support.v7.widget.Toolbar;
import android.view.GestureDetector;
import android.view.MotionEvent;
import android.view.View;
import android.widget.Toast;

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

public class MainActivity extends AppCompatActivity {
    private List<Movie> movieList = new ArrayList<>();
    private RecyclerView recyclerView;
    private MoviesAdapter mAdapter;


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


        recyclerView = (RecyclerView) findViewById(R.id.recycler_view);

        mAdapter = new MoviesAdapter(movieList);
        RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(getApplicationContext());
```

```
        recyclerView.setLayoutManager(mLayoutManager);
        recyclerView.setItemAnimator(new DefaultItemAnimator());
        recyclerView.setAdapter(mAdapter);
        recyclerView.setHasFixedSize(true);

        prepareMovieData();
    }

    private void prepareMovieData() {
        Movie movie = new Movie("Mad Max: Fury Road", "Action & Adventure", "2015");
        movieList.add(movie);

        movie = new Movie("Inside Out", "Animation, Kids & Family", "2015");
        movieList.add(movie);

        movie = new Movie("Star Wars: Episode VII - The Force Awakens", "Action", "2015");
        movieList.add(movie);

        movie = new Movie("Shaun the Sheep", "Animation", "2015");
        movieList.add(movie);

        movie = new Movie("The Martian", "Science Fiction & Fantasy", "2015");
        movieList.add(movie);

        movie = new Movie("Mission: Impossible Rogue Nation", "Action", "2015");
        movieList.add(movie);

        movie = new Movie("Up", "Animation", "2009");
        movieList.add(movie);

        movie = new Movie("Star Trek", "Science Fiction", "2009");
        movieList.add(movie);

        movie = new Movie("The LEGO Movie", "Animation", "2014");
        movieList.add(movie);

        movie = new Movie("Iron Man", "Action & Adventure", "2008");
        movieList.add(movie);

        movie = new Movie("Aliens", "Science Fiction", "1986");
        movieList.add(movie);

        movie = new Movie("Chicken Run", "Animation", "2000");
        movieList.add(movie);

        movie = new Movie("Back to the Future", "Science Fiction", "1985");
        movieList.add(movie);
```

```
        movie = new Movie("Raiders of the Lost Ark", "Action & Adventure", "1981");
        movieList.add(movie);

        movie = new Movie("Goldfinger", "Action & Adventure", "1965");
        movieList.add(movie);

        movie = new Movie("Guardians of the Galaxy", "Science Fiction & Fantasy", "2014");
        movieList.add(movie);

        mAdapter.notifyDataSetChanged();
    }
}
```

Now if you run the app, you can see the movies displayed in a list manner.