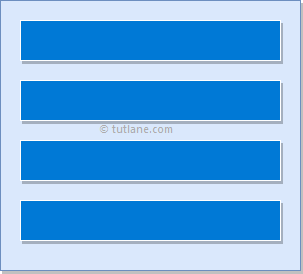
**7. Android ListView with Examples**

In android, **ListView** is a **ViewGroup** which is used to display the list of scrollable of items in multiple rows and the list items are automatically inserted to the list using an **adapter**.

Generally, the adapter pulls data from a sources such as an array or database and converts each item into a result view and that’s placed into the list.

Following is the pictorial representation of listview in android applications.



**Android Adapter**

In android, **Adapter** will act as an intermediate between the data sources and adapter views such as [ListView](https://www.tutlane.com/tutorial/android/android-listview-with-examples" \o "Android List View with Examples" \t "_blank), [Gridview](https://www.tutlane.com/tutorial/android/android-gridview-with-examples" \o "Android Gridview with Examples" \t "_blank) to fill the data into adapter views. The adapter will hold the data and iterates through an items in data set and generate the views for each item in the list.

Generally, in android we have a different types of adapters available to fetch the data from different data sources to fill the data into adapter views, those are

| **Adapter** | **Description** |
| --- | --- |
| ArrayAdapter | It will expects an Array or List as input. |
| CurosrAdapter | It will accepts an instance of cursor as an input. |
| SimpleAdapter | It will accepts a static data defined in the resources. |
| BaseAdapter | It is a generic implementation for all three adapter types and it can be used for [ListView](https://www.tutlane.com/tutorial/android/android-listview-with-examples" \o "Android List View with Examples" \t "_blank), [Gridview](https://www.tutlane.com/tutorial/android/android-gridview-with-examples" \o "Android Gridview with Examples" \t "_blank) or Spinners based on our requirements. |

**Android ListView Example**

Following is the example of creating a **ListView** using **arrayadapter** in android application.

Create a new android application using android studio and give names as **ListView**. In case if you are not aware of creating an app in android studio check this article [Android Hello World App](https://www.tutlane.com/tutorial/android/android-hello-world-app-example).

Now open an **activity\_main.xml** file from **\res\layout** path and write the code like as shown below

**activity\_main.xml**

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout\_width="match\_parent"  
    android:layout\_height="match\_parent"  
    android:orientation="vertical">  
    <ListView  
        android:id="@+id/userlist"  
        android:layout\_width="match\_parent"  
        android:layout\_height="wrap\_content" >  
    </ListView>  
</LinearLayout>

Once we are done with creation of layout, now we will bind data to our **ListView**  using

**ArrayAdapter**, for that open main activity file **MainActivity.java** from **\java\com.tutlane.listview**

path and write the code like as shown below.

**MainActivity.java**

package com.tutlane.listview;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.widget.ArrayAdapter;  
import android.widget.ListView;  
  
public class MainActivity extends AppCompatActivity {  
    private ListView mListView;  
    private ArrayAdapter aAdapter;  
    private String[] users = { "Suresh Dasari", "Rohini Alavala", "Trishika Dasari", "Praveen Alavala", "Madav Sai", "Hamsika Yemineni"};  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity\_main);  
        mListView = (ListView) findViewById(R.id.userlist);  
        aAdapter = new ArrayAdapter(this, android.R.layout.simple\_list\_item\_1, users);  
        mListView.setAdapter(aAdapter);  
    }  
}

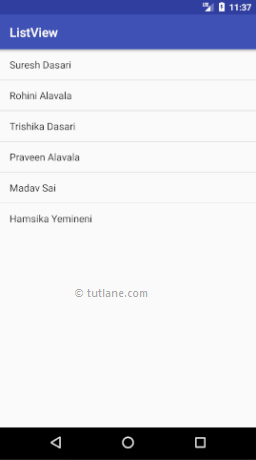
If we observe above code, we are binding static array (**users**) details to **ListView** using

**ArrayAdapter** and calling our layout using **setContentView** method in the form of **R.layout.layout\_file\_name**. Here our xml file name is **activity\_main.xml** so we used file name **activity\_main**.

Generally, during the launch of our [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle), onCreate() callback method will be called by android framework to get the required layout for an [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle).

**Output**

When we run above example using android virtual device (AVD) we will get a result like as shown below.



This is how we can bind data to ListView using ArrayAdapter in android applications based on our requirements.

**Android ListView with Custom Adapter Example**

In previous example, we learned simple way to bind data to **ListView** using **ArrayAdapter** in android application. Now we will see how to create our own custom adapter and bind data to ListView with example.

For this, we need to create our own custom adapter class by extending with **BaseAdapter** class and create a class which will contain a parameters for list row items.

Now create a new android application using android studio and give names as **ListView**. In case if you are not aware of creating an app in android studio check this article [Android Hello World App](https://www.tutlane.com/tutorial/android/android-hello-world-app-example).

Open an **activity\_main.xml** file from \res\layout path and write the code like as shown below

**activity\_main.xml**

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout\_width="fill\_parent"  
    android:layout\_height="fill\_parent"  
    android:orientation="vertical" >  
    <ListView  
        android:id="@+id/user\_list"  
        android:layout\_width="fill\_parent"  
        android:layout\_height="wrap\_content"  
        android:dividerHeight="1dp" />  
</LinearLayout>

Now we need to create a layout for **listview** row items, for that right click on **layouts** folder  select New  Layout resource file  Give name as **list\_row.xml** and click **OK**. Now open newly created file (**list\_row.xml**) and write the code like as shown below

list\_row.xml

<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout\_width="fill\_parent"  
    android:layout\_height="wrap\_content"  
    android:orientation="horizontal"  
    android:padding="5dip" >  
    <TextView  
        android:id="@+id/name"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:textStyle="bold"  
        android:textSize="17dp" />  
    <TextView  
        android:id="@+id/designation"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:layout\_below="@id/name"  
        android:layout\_marginTop="7dp"  
        android:textColor="#343434"  
        android:textSize="14dp" />  
    <TextView  
        android:id="@+id/location"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:layout\_alignBaseline="@+id/designation"  
        android:layout\_alignBottom="@+id/designation"  
        android:layout\_alignParentRight="true"  
        android:textColor="#343434"  
        android:textSize="14dp" />  
</RelativeLayout>

Now we need to create a custom class (**ListItem.java**) to represent each row in the list, for that right click on **java**folder  select New  Java Class  Give name as **ListItem.java** and click **OK**. Open **ListItem.java** file and write the code like as shown below

**ListItem.java**

package com.tutlane.listview;  
/\*\*  
 \* Created by tutlane on 23-08-2017.  
 \*/  
public class ListItem {  
    private String name;  
    private String designation;  
    private String location;  
    public String getName() {  
        return name;  
    }  
    public void setName(String name) {  
        this.name = name;  
    }  
    public String getDesignation() {  
        return designation;  
    }  
    public void setDesignation(String designation) {  
        this.designation = designation;  
    }  
    public String getLocation() {  
        return location;  
    }  
    public void setLocation(String location) {  
        this.location = location;  
    }  
}

Now we need to create a custom adapter (**CustomListAdapter.java**) and extend it by using **BaseAdapter**. In case if we are extending our class by using **BaseAdapter**, we need to override following methods from **BaseAdapter** class.

| **Method** | **Description** |
| --- | --- |
| getCount() | It will return total number of rows count in listview |
| getItem() | It is used to specify the object data of each row |
| getItemId() | It return the id of each row item |
| getView() | It is used to return a view instance that represents a single row in ListView item. |

To create custom adapter right click on **java** folder  select New  Java Class  Give name as **CustomListAdapter.java**and click **OK**.

Open **CustomListAdapter.java** file and write the code like as shown below

**CustomListAdapter.java**

package com.tutlane.listview;  
import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.BaseAdapter;  
import android.widget.TextView;  
  
import java.util.ArrayList;  
/\*\*  
 \* Created by tutlane on 23-08-2017.  
 \*/  
public class CustomListAdapter extends BaseAdapter {  
    private ArrayList<ListItem> listData;  
    private LayoutInflater layoutInflater;  
    public CustomListAdapter(Context aContext, ArrayList<ListItem> listData) {  
        this.listData = listData;  
        layoutInflater = LayoutInflater.from(aContext);  
    }  
    @Override  
    public int getCount() {  
        return listData.size();  
    }  
    @Override  
    public Object getItem(int position) {  
        return listData.get(position);  
    }  
    @Override  
    public long getItemId(int position) {  
        return position;  
    }  
    public View getView(int position, View v, ViewGroup vg) {  
        ViewHolder holder;  
        if (v == null) {  
            v = layoutInflater.inflate(R.layout.list\_row, null);  
            holder = new ViewHolder();  
            holder.uName = (TextView) v.findViewById(R.id.name);  
            holder.uDesignation = (TextView) v.findViewById(R.id.designation);  
            holder.uLocation = (TextView) v.findViewById(R.id.location);  
            v.setTag(holder);  
        } else {  
            holder = (ViewHolder) v.getTag();  
        }  
        holder.uName.setText(listData.get(position).getName());  
        holder.uDesignation.setText(listData.get(position).getDesignation());  
        holder.uLocation.setText(listData.get(position).getLocation());  
        return v;  
    }  
    static class ViewHolder {  
        TextView uName;  
        TextView uDesignation;  
        TextView uLocation;  
    }  
}

If we observe above class we are extending our custom adapter by using **BaseAdapter** and we override all **BaseAdapter**methods in our custom adapter.

Now we need to combine all the custom classes in main activity file (**MainActivity.java**) to bind the data to our listview.

Open main activity file (**MainActivity.java**) and write the code like as shown below.

**MainActivity.java**

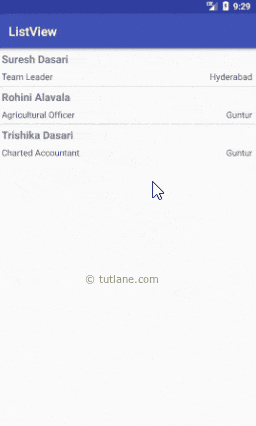
package com.tutlane.listview;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.AdapterView;  
import android.widget.ListView;  
import android.widget.Toast;  
  
import java.util.ArrayList;  
  
public class MainActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity\_main);  
        ArrayList userList = getListData();  
        final ListView lv = (ListView) findViewById(R.id.user\_list);  
        lv.setAdapter(new CustomListAdapter(this, userList));  
        lv.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
            @Override  
            public void onItemClick(AdapterView<?> a, View v, int position, long id) {  
                ListItem user = (ListItem) lv.getItemAtPosition(position);  
                Toast.makeText(MainActivity.this, "Selected :" + " " + user.getName()+", "+ user.getLocation(), Toast.LENGTH\_SHORT).show();  
            }  
        });  
    }  
    private ArrayList getListData() {  
        ArrayList<ListItem> results = new ArrayList<>();  
        ListItem user1 = new ListItem();  
        user1.setName("Suresh Dasari");  
        user1.setDesignation("Team Leader");  
        user1.setLocation("Hyderabad");  
        results.add(user1);  
        ListItem user2 = new ListItem();  
        user2.setName("Rohini Alavala");  
        user2.setDesignation("Agricultural Officer");  
        user2.setLocation("Guntur");  
        results.add(user2);  
        ListItem user3 = new ListItem();  
        user3.setName("Trishika Dasari");  
        user3.setDesignation("Charted Accountant");  
        user3.setLocation("Guntur");  
        results.add(user3);  
        return results;  
    }  
}

If we observe above code we are building and binding data to ListView using our custom adapter and calling our layout using **setContentView** method in the form of **R.layout.layout\_file\_name**. Here our xml file name is **activity\_main.xml** so we used file name **activity\_main**.

Generally, during the launch of our [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle), **onCreate()** callback method will be called by android framework to get the required layout for an [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle).

**Output**

When we run above example using android virtual device (AVD) we will get a result like as shown below.



 This is how we can bind data to ListView using custom adapter in android applications based on our requirements.