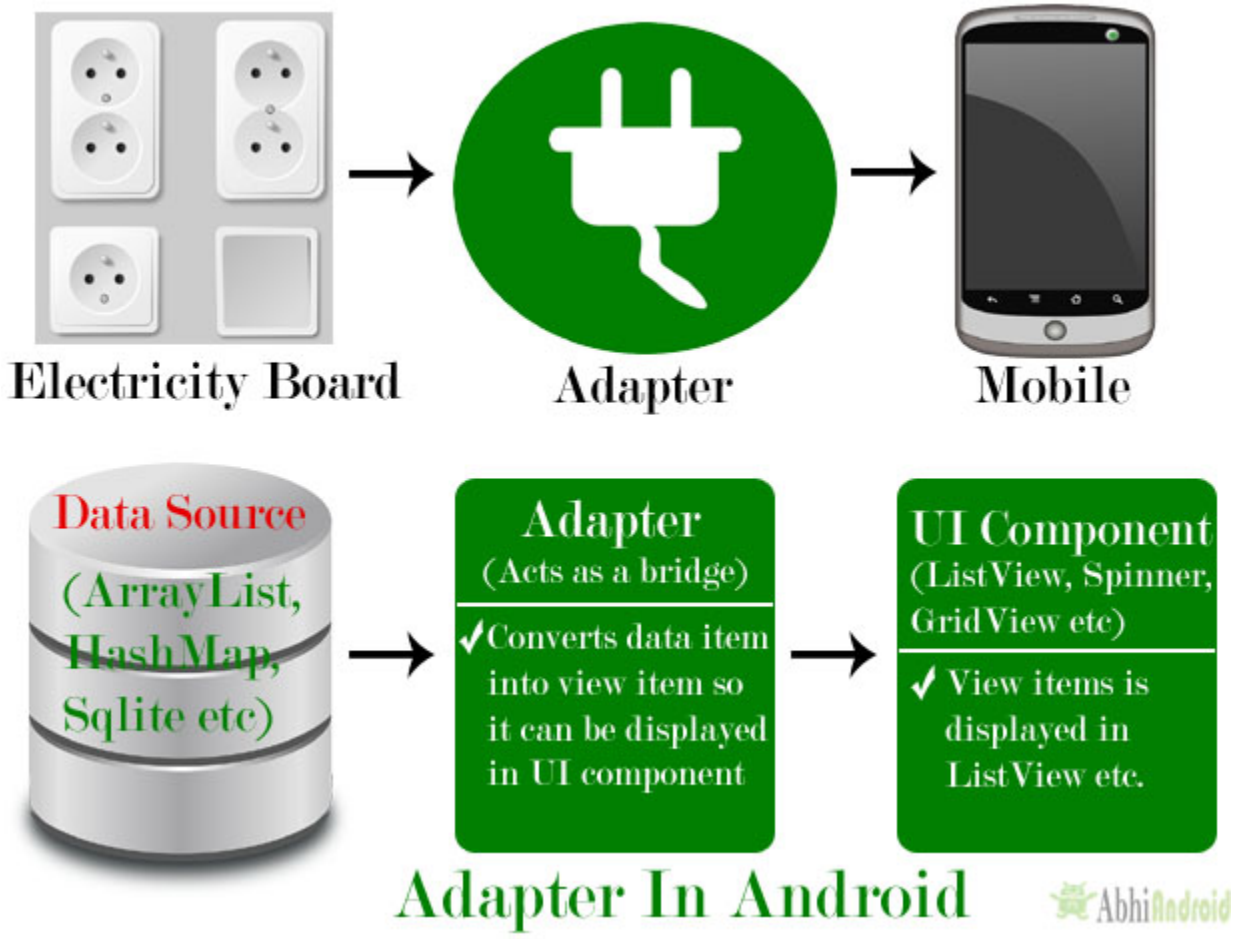
Adapter

In Android, Adapter is a bridge between UI component and data source that helps us to fill data in UI component. It holds the data and send the data to an Adapter view then view can takes the data from the adapter view and shows the data on different views like as ListView, GridView, Spinner etc. For more customization in Views we uses the base adapter or custom adapters.

To fill data in a list or a grid we need to implement Adapter. Adapters acts like a bridge between UI component and data source. Here data source is the source from where we get the data and UI components are list or grid items in which we want to display that data.

Below is a conceptual diagram of Adapter:



Adapters In Android:

There are the some commonly used Adapter in Android used to fill the data in the UI components.

BaseAdapter – It is parent adapter for all other adapters

ArrayAdapter – It is used whenever we have a list of single items which is backed by an array

Custom ArrayAdapter – It is used whenever we need to display a custom list

SimpleAdapter – It is an easy adapter to map static data to views defined in your XML file

Custom SimpleAdapter – It is used whenever we need to display a customized list and needed to access the child items of the list or grid

Now we describe each Adapters one by one in detail:

1. BaseAdapter In Android:

BaseAdapter is a common base class of a general implementation of an Adapter that can be used in ListView, GridView, Spinner etc. Whenever we need a customized list in a ListView or customized grids in a GridView we create our own adapter and extend base adapter in that. Base Adapter can be extended to create a custom Adapter for displaying a custom list item. ArrayAdapter is also an implementation of BaseAdapter.

Custom Adapter code which extends the BaseAdapter in that:

```
public class CustomAdapter extends BaseAdapter {

@Override
public int getCount() {
return 0;
}

@Override
public Object getItem(int i) {
return null;
}

@Override
public long getItemId(int i) {
return 0;
}

@Override
public View getView(int i, View view, ViewGroup viewGroup) {
```

```
return null;
}
```

In above code snippet we see the overrided functions of BaseAdapter which are used to set the data in a list, grid or a spinner. These functions are described in BaseAdapter tutorial with example.

2. ArrayAdapter In Android:

Whenever we have a list of single items which is backed by an Array, we can use ArrayAdapter. For instance, list of phone contacts, countries or names.

Here is how android ArrayAdapter looks :

```
ArrayAdapter(Context context, int resource, int textViewResourceId, T[] objects)
```

The above function are described in ArrayAdapter tutorial with example.

3. Custom ArrayAdapter In Android:

ArrayAdapter is also an implementation of BaseAdapter, so if we want more customization then we can create a custom adapter and extend ArrayAdapter in that. Since array adapter is an implementation of BaseAdapter, so we can override all the function's of BaseAdapter in our custom adapter.

Below Custom adapter class MyAdapter extends ArrayAdapter in that:

```
public class MyAdapter extends ArrayAdapter {

public MyAdapter(Context context, int resource, int textViewResourceId, List objects) {
super(context, resource, textViewResourceId, objects);
}
@Override
public int getCount() {
return super.getCount();
}

@Override
public View getView(int position, View convertView, ViewGroup parent) {
return super.getView(position, convertView, parent);
}
}
```

These functions are described in Custom ArrayAdapter tutorial with example.

4. SimpleAdapter In Android:

In Android SimpleAdapter is an easy Adapter to map static data to views defined in an XML file(layout). In Android we can specify the data backing to a list as an ArrayList of Maps(i.e. hashmap or other). Each entry in a ArrayList is corresponding to one row of a list.

The Map contains the data for each row. Here we also specify an XML file(custom list items file) that defines the views which is used to display the row, and a mapping from keys in the Map to specific views.

Whenever we have to create a custom list we need to implement custom adapter. As we discuss earlier ArrayAdapter is used when we have a list of single item's backed by an Array. So if we need more customization in a ListView or a GridView we need to implement simple adapter.

SimpleAdapter code in Android:

```
SimpleAdapter (Context context, List<? extends Map<String, ?>> data, int resource, String[]
from, int[] to)
```

The above parameters of a simple Adapter is described in SimpleAdapter tutorial with example.

5. Custom SimpleAdapter In Android:

Whenever we have to create a custom list we need to implement custom adapter. As we discuss earlier ArrayAdapter is used when we have a list of single item's backed by an Array. So if we need customization in a ListView or a GridView we need to implement simple Adapter but when we need more customization in list or grid items where we have many view's in a list item and then we have to perform any event like click or any other event to a particular view then we need to implement a custom adapter who fulfills our requirement's and quite easy to be implemented.

BaseAdapter is the parent adapter for all other adapters so if we extends a SimpleAdapter then we can also override the base adapter's function in that class.

Important Note: We can't perform events like click and other event on child item of a list or grid but if we have some requirements to do that then we can create our own custom adapter and extends the simple adapter in that.

Custom Adapter extends SimpleAdapter in that:

```
public class CustomAdapter extends SimpleAdapter {
public CustomAdapter(Context context, List<? extends Map<String, ?>> data, int resource,
String[] from, int[] to) {
super(context, data, resource, from, to);

}

@Override
public View getView(int position, View convertView, ViewGroup parent) {
return super.getView(position, convertView, parent);

}

@Override
public int getCount() {
return super.getCount();
}
}
```

The above overrided functions of simple adapter are already described in Custom SimpleAdapter article.

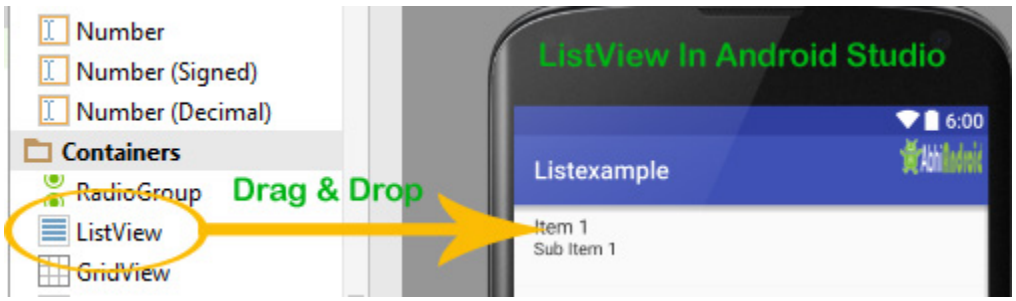
ListView

ListView is a list of scrollable items can be displayed in Android using ListView. It helps you to displaying the data in the form of a scrollable list. Users can then select any list item by clicking on it. ListView is default scrollable so we do not need to use scroll View or anything else with ListView.

ListView is widely used in android applications. A very common example of ListView is your phone contact book, where you have a list of your contacts displayed in a ListView and if you click on it then user information is displayed.

Adapter: To fill the data in a ListView we simply use adapters. List items are automatically inserted to a list using an Adapter that pulls the content from a source such as an arraylist, array or database.

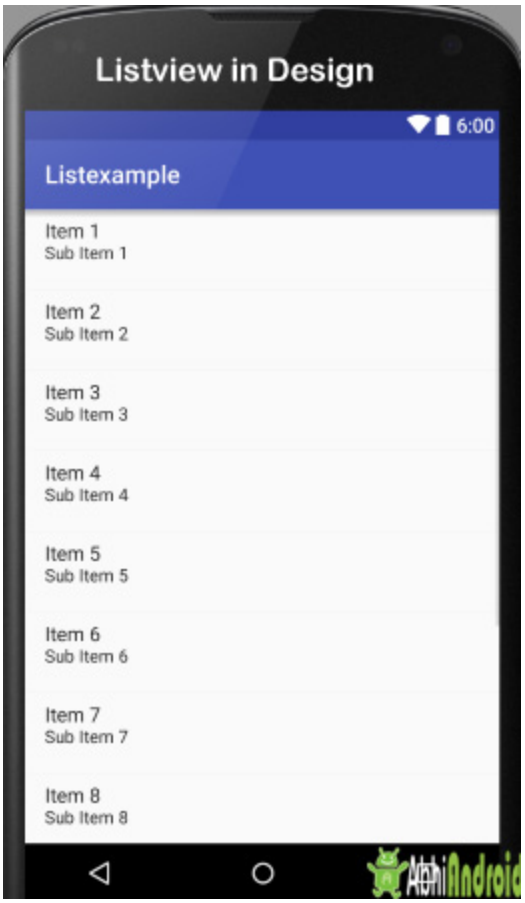
ListView in Android Studio: Listview is present inside Containers. From there you can drag and drop on virtual mobile screen to create it. Alternatively you can also XML code to create it.



Here is Android ListView XML Code:

```
<ListView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/simpleListView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    tools:context="abhiandroid.com.listexample.MainActivity">
</ListView>
```

Listview look in Design:



Attributes of ListView:

Lets see some different attributes of ListView which will be used while designing a custom list:

1. id: id is used to uniquely identify a ListView.

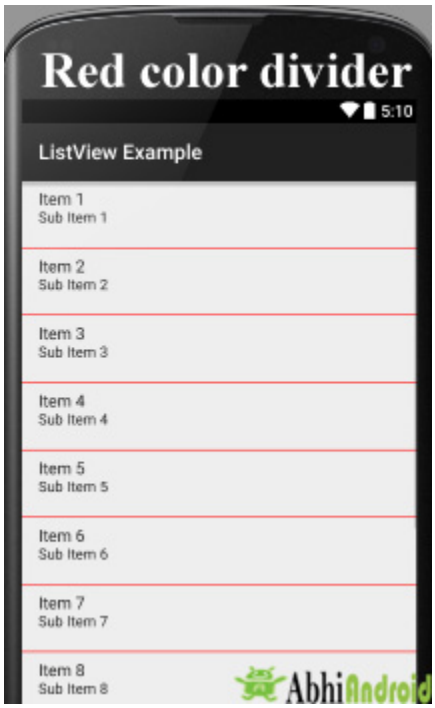
Below is the id attribute's example code with explanation included.

```
<!-- Id of a list view uniquely identify it-->
<ListView
android:id="@+id/simpleListView"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
/>
```

2. divider: This is a drawable or color to draw between different list items.

Below is the divider example code with explanation included, where we draw red color divider between different views.

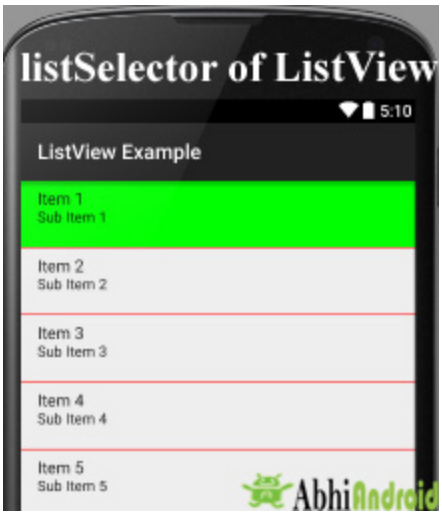
```
<!--Divider code in ListView-->
<ListView
android:id="@+id/simpleListView"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:divider="#f00"
android:dividerHeight="1dp"
/>
```



3. dividerHeight: This specify the height of the divider between list items. This could be in dp(density pixel),sp(scale independent pixel) or px(pixel).

In above example of divider we also set the divider height 1dp between the list items. The height should be in dp,sp or px.

4. listSelector: listSelector property is used to set the selector of the listView. It is generally orange or Sky blue color mostly but you can also define your custom color or an image as a list selector as per your design.



Below is listSelector example code with explanation includes, where list selector color is green, when you select any list item then that item's background color is green .

```
<!-- List Selector Code in ListView -->
<ListView
android:id="@+id/simpleListView"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:divider="#f00"
android:dividerHeight="1dp"
android:listSelector="#0f0"/> <!--list selector in green color-->
```

Adapters Use in ListView:

An adapter is a bridge between UI component and data source that helps us to fill data in UI component. It holds the data and send the data to adapter view then view can takes the data from the adapter view and shows the data on different views like as list view, grid view, spinner etc.

ListView is a subclass of AdapterView and it can be populated by binding to an Adapter, which retrieves the data from an external source and creates a View that represents each data entry.

In android commonly used adapters are:

Array Adapter

Base Adapter

Now we explain these two adapter in detail:

1.Array Adapter:

Whenever you have a list of single items which is backed by an array, you can use ArrayAdapter. For instance, list of phone contacts, countries or names.

Important Note: By default, ArrayAdapter expects a Layout with a single TextView, If you want to use more complex views means more customization in list items, please avoid ArrayAdapter and use custom adapters.

Below is Array Adapter code:

```
ArrayAdapter adapter = new
ArrayAdapter<String>(this,R.layout.ListView,R.id.textView,StringArray);
```

2.Base Adapter:

BaseAdapter is a common base class of a general implementation of an Adapter that can be used in ListView. Whenever you need a customized list you create your own adapter and extend base adapter in that. Base Adapter can be extended to create a custom Adapter for displaying a custom list item. ArrayAdapter is also an implementation of BaseAdapter.