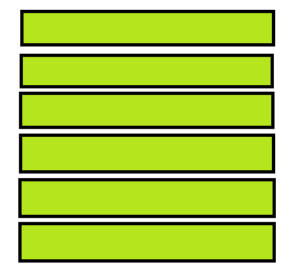
**ListView**

Do you want to display a list in your app? Android has its solution by providing list view layout. It is a layout which displays items in a vertical scroll-able list. Each item in the list is positioned below to the previous item of the list. List items are stored in an array and inserted to the list by using adapter which pulls items from array. List view is sub class of AdapterView.  Custom lists are very common in mobile application and list view provides a very easy way to create custom lists. It is one of the most used layouts, for example when you want to display a group of date, nothing could be more suitable than a list. Following figure shows how list view looks

[](https://javatutorial.net/wp-content/uploads/2017/08/list-view.png)

list view

**Attributes of List View**

Here are some important XML attributes of list view.

**android: divider,**it is used as a draw able or color to draw between list items.

**android: entries,**it is used to reference an array resource to populate the list view.

**android: headerDividersEnabled,**used to draw divider after each header views.

**android: footerDividersEanabled,**used to draw divider before each footer views.

**Methods of List View**

List view has many public methods some of them are explained below.

* **getAdapter( ):**It returns adapter used in list view.
* **addHeaderView( ):**It is used to add a header view at the top of list.
* **getAccessibilityClassName( ):**It returns the class name of object.
* **getDivider( ):**returns the divider between each item of the list.
* **getDividerHeight( ):**returns divider height.
* **isOpaque( ):**Shows whether list is opaque or not.
* **removeFooterView( View view):**It is used to remove previously added footer view in the list.
* **removeHeaderView( View view):**It is used to remove previously added header view in the list.

**Constructors of List View**

List view has four different public constructors as written below:

* ListView( Context context )
* ListView( Context context, AttributeSet attrs )
* ListView( Context context, AttributeSet attrs, int defStyleAttr )
* ListView( Context context, AttributeSet attrs, AttributeSet attrs, int defStyleRes )

(adsbygoogle = window.adsbygoogle || []).push({});

**Example of List View**

Here is an example of using list view in Android. It shows list of languages in computer science. Let’s start it by creating an activity, open activity \_main.xml and paste this code

1. **<?xml version="1.0" encoding="utf-8"?>**
2. **<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"**
3. **xmlns:tools="http://schemas.android.com/tools"**
4. **android:layout\_width="match\_parent"**
5. **android:layout\_height="match\_parent"**
6. **android:orientation="vertical"**
7. **>**
8. **<ListView**
9. **android:id="@+id/mobile\_list"**
10. **android:layout\_width="match\_parent"**
11. **android:layout\_height="wrap\_content" >**
12. **</ListView>**
13. **</LinearLayout>**

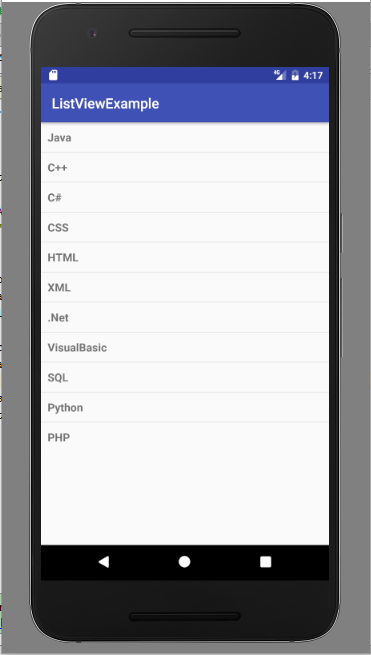
Create another layout as activity\_listview.xml and paste this code

1. **<?xml version="1.0" encoding="utf-8"?>**
2. **<!-- Single List Item Design -->**
3. **<TextView xmlns:android="http://schemas.android.com/apk/res/android"**
4. **android:id="@+id/label"**
5. **android:layout\_width="fill\_parent"**
6. **android:layout\_height="fill\_parent"**
7. **android:padding="10dip"**
8. **android:textSize="16dip"**
9. **android:textStyle="bold" >**
10. **</TextView>**

open your MainActivity.java and use this code

1. **package com.example.admin.listviewexample;**
2. **import android.support.v7.app.AppCompatActivity;**
3. **import android.os.Bundle;**
4. **import android.widget.ArrayAdapter;**
5. **import android.widget.ListView;**
6. **public class MainActivity extends AppCompatActivity {**
7. **String[] mobileArray = {"Java","C++","C#","CSS",**
8. **"HTML","XML",".Net","VisualBasic", "SQL", "Python", "PHP"};**
9. **@Override**
10. **protected void onCreate(Bundle savedInstanceState) {**
11. **super.onCreate(savedInstanceState);**
12. **setContentView(R.layout.activity\_main);**
13. **ArrayAdapter adapter = new ArrayAdapter<String>(this,**
14. **R.layout.activity\_listview, mobileArray);**
15. **ListView listView = (ListView) findViewById(R.id.mobile\_list);**
16. **listView.setAdapter(adapter);**
17. **}**
18. **}**

Here is the output how it looks like

[](https://javatutorial.net/wp-content/uploads/2017/08/list-view-example.png)

## ****Android ListAdapter****

List view is very simple, do you want to customize your list? ListAdapter is used to customize list view layout. It behaves like a bridge between data source and a list view. Its parent class is BaseAdapter. It is used when you want to specify layout for individual rows in the list.  Don’t confuse it with ArrayAdapter, ListAdapter is an interface while ArrayAdapter is a class which work with array of data. Following figure shows a custom list view by using  ListAdapter.

[](https://javatutorial.net/wp-content/uploads/2017/08/List-Adapter.jpg)

List Adapter

**ListAdapter Constructor**

Constructor of ListAdapter takes a parameter that specifies a layout resource for each row. Other than this it also has two more parameters (these two parameters are typically parallel arrays) which enable us to specify which data field to associate with which object in the row layout resource.

**Methods of ListAadapter**

ListAdapter has following public methods.

* **areAllItemsEnabled( ):**It tells if all items in list adapter are enabled or not, by returning a Boolean value. If it returns true, it means all items are selectable and clickable. If it returns different values (if value changes) there is no guarantee it will take effect.
* **isEnabled( int position ):**returns true if the item at a specific position is clickable and selectable. If position given is invalid then result can not be specified.

**BaseAdapter**

public abstract class BaseAdapter   
extends Object implements ListAdapter, SpinnerAdapter

What is Context?

As the name suggests, it's the context of current state of the application/object. It lets newly-created objects understand what has been going on. Typically you call it to get information regarding another part of your program (activity and package/application).

You can get the context by invoking getApplicationContext(), getContext(), getBaseContext()or this (when in a class that extends from Context, such as the Application, Activity, Service and IntentService classes).

Typical uses of context:

Creating new objects: Creating new views, adapters, listeners:

TextView tv = new TextView(getContext());

ListAdapter adapter = new SimpleCursorAdapter(getApplicationContext(), ...);

Accessing standard common resources: Services like LAYOUT\_INFLATER\_SERVICE, SharedPreferences:

context.getSystemService(LAYOUT\_INFLATER\_SERVICE)

getApplicationContext().getSharedPreferences(\*name\*, \*mode\*);

Accessing components implicitly: Regarding content providers, broadcasts, intent

getApplicationContext().getContentResolver().query(uri, ...);

**LayoutInflater** is used to manipulate **Android** screen using predefined XML layouts. This class is used to instantiate layout XML file into its corresponding View objects. It is never used directly.

**What is Layoutinflater ?**

LayoutInflater is a class (wrapper of some implementation or service), you can get one:

LayoutInflater li = LayoutInflater.from(context);

**How to use Layoutinflater ?**

You feed it an XML layout file. You need not give full file address, just its resource id, generated for you automatically in R class. For example, a layout file which look like:

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent">

<TextView

android:id="@+id/text\_view"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"/>

</LinearLayout>

saved as /res/layout/my\_layout.xml.

You give it to LayoutInflater like:

View v = li.inflate(R.layout.my\_layout,null,false);

**What did Layout Inflater do ?**

That v is now a LinearLayout object (LinearLayoutextendsView) , and contains a TextViewobject, arranged in exact order and with all properties set, as we *described* in the XML above.