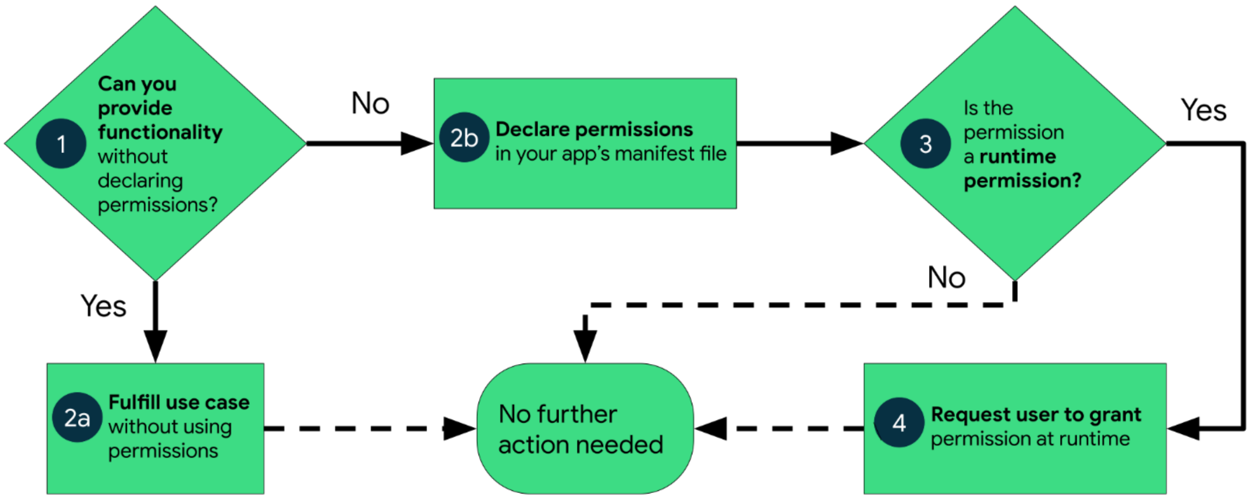
Permissions in Android

- Every Android app runs in a limited-access sandbox.
- If your app needs to use resources or information outside of its own sandbox, you can declare a permission and set up a permission request that provides this access.
- App permissions help support user privacy by protecting access to the following:
    - Restricted data, such as system state and a user's contact information.
    - Restricted actions, such as connecting to a paired device and recording audio.

General Flow of App Permission


1 Can you provide functionality without declaring permissions?
No
2b Declare permissions in your app's manifest file
3 Is the permission a runtime permission?
Yes
Yes
No
2a Fulfill use case without using permissions
No further action needed
4 Request user to grant permission at runtime

Types of permissions

1. Install time permissions
    - Install-time permissions give your app limited access to restricted data, and they allow your app to perform restricted actions that minimally affect the system or other apps.

- When you declare install-time permissions in your app, the system automatically grants your app the permissions when the user installs your app.
- Android includes several sub-types of install-time permissions, including normal permissions and signature permissions.

1.1 Normal Permissions

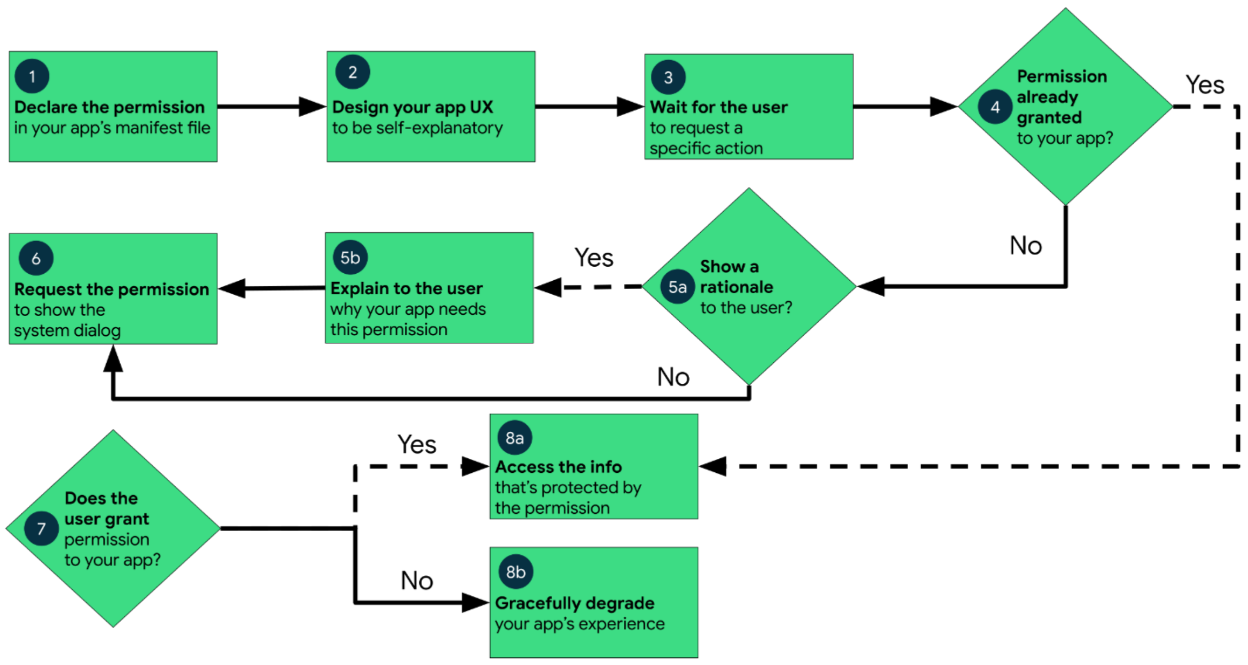
- These permissions allow access to data and actions that extend beyond your app's sandbox.
- However, the data and actions present very little risk to the user's privacy, and the operation of other apps.

1.2 Signature Permissions

- If the app declares a signature permission that another app has defined, and if the two apps are signed by the same certificate, then the system grants the permission to the first app at install time.
- Otherwise, that first app cannot be granted the permission.

2. Run-time Permissions
   - Runtime permissions, also known as dangerous permissions.
   - It gives your app additional access to restricted data, and they allow your app to perform restricted actions that more substantially affect the system and other apps.
   - Therefore, you need to request runtime permissions in your app before you can access the restricted data or perform restricted actions.

Work Flow for Android Run time Permission


1 Declare the permission in your app's manifest file
2 Design your app UX to be self-explanatory
3 Wait for the user to request a specific action
4 Permission already granted to your app?
Yes
No
5a Show a rationale to the user?
Yes
No
5b Explain to the user why your app needs this permission
6 Request the permission to show the system dialog
7 Does the user grant permission to your app?
Yes
No
8a Access the info that's protected by the permission
8b Gracefully degrade your app's experience

1. In your app's manifest file, declare the permissions that your app might need to request.
2. Design your app's UX so that specific actions in your app are associated with specific runtime permission.
3. Wait for the user to invoke the task or action in your app that requires access to specific private user data. At that time, your app can request the runtime permission that's required for accessing that data.
4. Check whether the user has already granted the runtime permission that your app requires. If so, your app can access the private user data. If not, continue to the next step.
5. Check whether your app should show a rationale to the user, explaining why your app needs the user to grant a particular runtime permission. If the system determines that your app shouldn't show a rationale, continue to the next step directly, without showing a UI element.
6. Request the runtime permission that your app requires in order to access the private user data. The system displays a runtime permission prompt, such as the one shown on the permissions overview page.

7. Check the user's response, whether they chose to grant or deny the runtime permission.
8. If the user granted the permission to your app, you can access the private user data. If the user denied the permission instead, gracefully degrade your app experience so that it provides functionality to the user, even without the information that's protected by that permission.

How to Handle permission denial

- If the user denies a permission request, your app should help users understand the implications of denying the permission. In particular, your app should make users aware of the features that don't work because of the missing permission. When you do so, keep the following best practices in mind:

- Guide the user's attention. Highlight a specific part of your app's UI where there's limited functionality because your app doesn't have the necessary permission. Several examples of what you could do include the following:
    - Show a message where the feature's results or data would have appeared.
    - Display a different button that contains an error icon and color.

- Be specific. Don't display a generic message; instead, mention which features are unavailable because your app doesn't have the necessary permission.

- Don't block the user interface. In other words, don't display a full-screen warning message that prevents users from continuing to use your app at all.

How can we determine that App was already grant the permission

- To check if the user has already granted your app a particular permission, pass that permission into the **ContextCompat.checkSelfPermission()** method.
- This method returns either PERMISSION_GRANTED or **PERMISSION**_DENIED, depending on whether your app has the permission.

Use Rationale to explain the need of permission

- If the ContextCompat.checkSelfPermission() method returns PERMISSION_DENIED, call **shouldShowRequestPermissionRationale()**.
- If this method returns true, show an educational UI to the user. In this UI, describe why the feature, which the user wants to enable, needs a particular permission.

Ask For the Permission

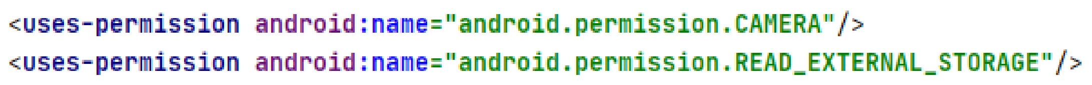
ActivityCompat.requestPermissions(MainActivity.this,new String[]{permission},requestcode);

Steps for Requesting permissions at run time

Step 1: Declare the permission in the **Android Manifest file**:

In Android, permissions are declared in the AndroidManifest.xml file using the uses-permission tag.

<uses-permission android:name=”android.permission.PERMISSION_NAME”/>



```
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

Step 2: Design activity_main.xml file for UI to add Button

Step 3: Add Code in MainActivity.java file to check whether permission is already granted or not.

```
if(ContextCompat.checkSelfPermission(thisActivity,
Manifest.permission.WRITE_CALENDAR)
  != PackageManager.PERMISSION_GRANTED)
{
   // Permission is not granted
}
```

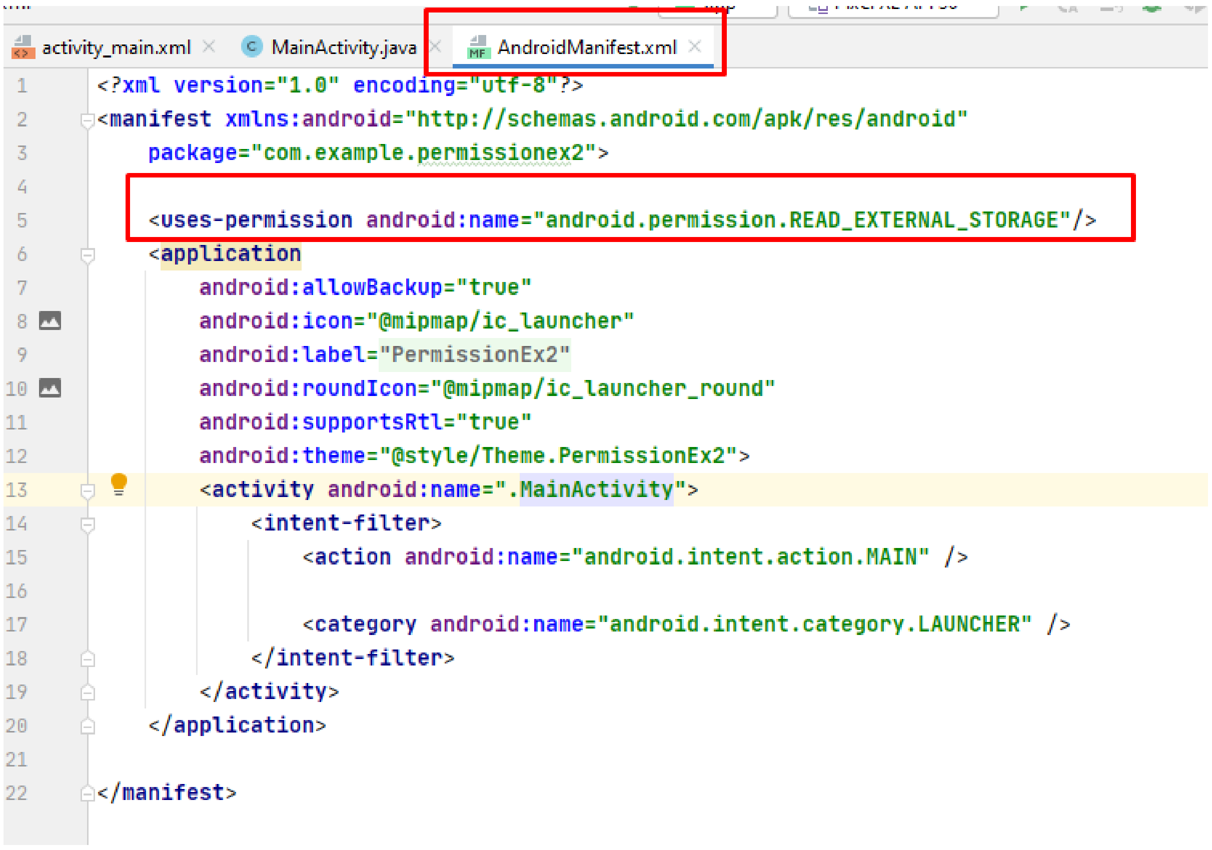
Step 4: Override onRequestPermissionsResult() method:
onRequestPermissionsResult() is called when user grant or decline the permission. RequestCode is one of the parameters of this function which is used to check user action for the corresponding requests. Here a toast message is shown

Example of Permissions – 1

AIM :

- Design UI with a single button Request Permission
- Following code should be in MainActivity.java file
- On Click event of Button – It will ask for the permission (On button click event set in MainActivity.java file)
- Somehow, if user Denies to permission then **(ALERT BOX)** should be popped up.
- And Show user that WHY we need the permission **(Show Rationale)** function is called
- If user will ask for the Positive response then we go for asking request again
- Else we Dismiss the alert box
- Then If we Allow or Deny then onRequestPermissionsResult() is called…..

1. Manifest.xml file



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.permissionex2">

    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="PermissionEx2"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.PermissionEx2">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

2. Activity_main.xml file

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Request Button"
        android:id="@+id/requestButton"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

3. MainActivity.java file

```
package com.example.permissionex2;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

import android.Manifest;
import android.content.DialogInterface;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    private int STORAGE_PERMISSION_CODE = 100;
    Button requestPermissionBtn;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        requestPermissionBtn = findViewById(R.id.requestButton);

        requestPermissionBtn.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (ContextCompat.checkSelfPermission(MainActivity.this,
Manifest.permission.READ_EXTERNAL_STORAGE) ==
PackageManager.PERMISSION_GRANTED) {
                    Toast.makeText(MainActivity.this, "You have already
permission", Toast.LENGTH_SHORT).show();
                } else {
                    requestpermission();

                }
            }
        });
    }

    private void requestpermission(){

if(ActivityCompat.shouldShowRequestPermissionRationale(this,Manifest.per
mission.READ_EXTERNAL_STORAGE))
        {
            new AlertDialog.Builder(this)
                    .setTitle("Perission is needed")
                    .setMessage("This permission is " +
                            "needed because of some data access")
                    .setPositiveButton("Ok", new
DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int
which) {

ActivityCompat.requestPermissions(MainActivity.this,
                                    new
String[]{Manifest.permission.READ_EXTERNAL_STORAGE},STORAGE_PERMISSION_C
ODE);
                        }
                    })
                    .setNegativeButton("Cancel", new
DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int
which) {
                            dialog.dismiss();
                        }
                    })
                    .create().show();
        }
        else{
            ActivityCompat.requestPermissions(this,new String[]
{Manifest.permission.READ_EXTERNAL_STORAGE},STORAGE_PERMISSION_CODE);
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode,  String[]
permissions, int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
        if (requestCode == STORAGE_PERMISSION_CODE) {
            if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                Toast.makeText(this, "Permission Granted",
Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText(this, "Permission Not Granted",
Toast.LENGTH_SHORT).show();

            }

        }
    }
}
```

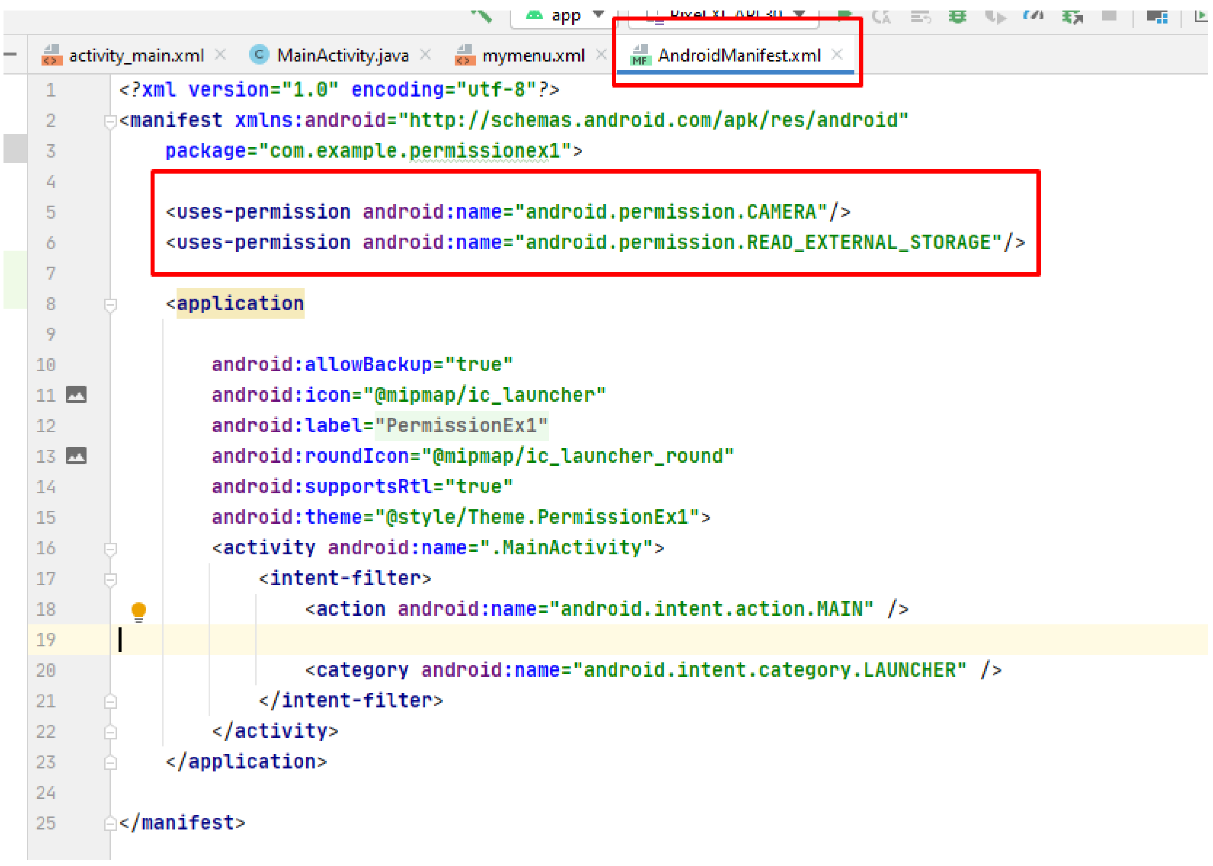
Example of Permissions – 2

AIM :

- Design UI with a Two buttons to Request Camera & Storage Permission
- Add Permission to manifestfile.xml
- Following code should be in MainActivity.java file
- On Click event of Camera Button – It will ask for the permission (On button click event set in MainActivity.java file)
- On Click event of Storage Button – It will ask for the permission (On button click event set in MainActivity.java file)

- If user allows then text of button will change and shows a toast message granted permission
- If permission is already granted then will show toast message PERMISSION is already granted

1. Manifest.xml file



activity_main.xml MainActivity.java mymenu.xml AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.permissionex1">

    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>

    <application

        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="PermissionEx1"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.PermissionEx1">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

2. Activity_main.xml file

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center"
    tools:context=".MainActivity">


    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Get Camera Permission"
        android:id="@+id/cameraButton"
        />


    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Get Storage Permission"
        android:id="@+id/storageButton"
        />

</LinearLayout>
```

3. MainActiviy.java File

```
package com.example.permissionex1;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

import android.Manifest;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    Button cameraBtn,storageBtn;

    private static final int CAMERA_PERMISSION_CODE =112;
    private static final int STORAGE_PERMISSION_CODE =113;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        cameraBtn = findViewById(R.id.cameraButton);
        storageBtn = findViewById(R.id.storageButton);

        cameraBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

checkPermission(Manifest.permission.CAMERA,CAMERA_PERMISSION_CODE);
            }
        });

        storageBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

checkPermission(Manifest.permission.READ_EXTERNAL_STORAGE,STORAGE_PER
MISSION_CODE);
            }
        });
        // we are going to make a global function to check
permissions
    }


    public void checkPermission(String permission, int requestcode){


if(ContextCompat.checkSelfPermission(MainActivity.this,permission)==
PackageManager.PERMISSION_DENIED)
        {
            ActivityCompat.requestPermissions(MainActivity.this,new
String[]{permission},requestcode);
        }else{


            Toast.makeText(MainActivity.this,"Permission Already
Granted",Toast.LENGTH_SHORT).show();
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode,  String[]
permissions,  int[] grantResults) {

        super.onRequestPermissionsResult(requestCode, permissions,
grantResults);

        if(requestCode == CAMERA_PERMISSION_CODE){
            if(grantResults.length>0 &&
grantResults[0]==PackageManager.PERMISSION_GRANTED)
            {
                cameraBtn.setText("Permission Granted For Camera");
                Toast.makeText(MainActivity.this,"CAMERA Permission
Granted",Toast.LENGTH_SHORT).show();
            }
            else{
                Toast.makeText(MainActivity.this,"CAMERA Permission
Not Granted",Toast.LENGTH_SHORT).show();
            }
        }
        else if(requestCode==STORAGE_PERMISSION_CODE)
        {
            if(grantResults.length>0 &&
grantResults[0]==PackageManager.PERMISSION_GRANTED)
            {
                storageBtn.setText("Permission Granted For Storage");
                Toast.makeText(MainActivity.this,"Storage Permission
Granted",Toast.LENGTH_SHORT).show();
            }
            else{
                Toast.makeText(MainActivity.this,"Storage Permission
Not Granted",Toast.LENGTH_SHORT).show();
            }
        }

    }
}
```

Storage in Android

Android provides several options for you to save your app data. The solution you choose depends on your specific needs, such as how much space your data requires, what kind of data you need to store, and whether the data should be private to your app or accessible to other apps and the user.

Android uses a file system that's similar to disk-based file systems on other platforms. The system provides several options for you to save your app data

Storage Options

1. Shared Preferences
2. Internal Storage
3. External Storage
4. SQLite Database
5. Network Storage

1. Shared Preferences

- In android, Shared Preferences are used to save and retrieve the primitive data types (integer, float, boolean, string, long) data in the form of key-value pairs from a file within an apps file structure.

- Generally, the Shared Preferences object will point to a file that contains key-value pairs and provides a simple read and write methods to save and retrieve the key-value pairs from a file.

- The Shared Preferences file is managed by an android framework and it can be accessed anywhere within the app to read or write data into the file, but it’s not possible to access the file from any other app so it’s secured.

- The Shared Preferences are useful to store the small collection of key-values such as user’s login information, app preferences related to users, etc. to maintain the state of the app, next time when they login again to the app.

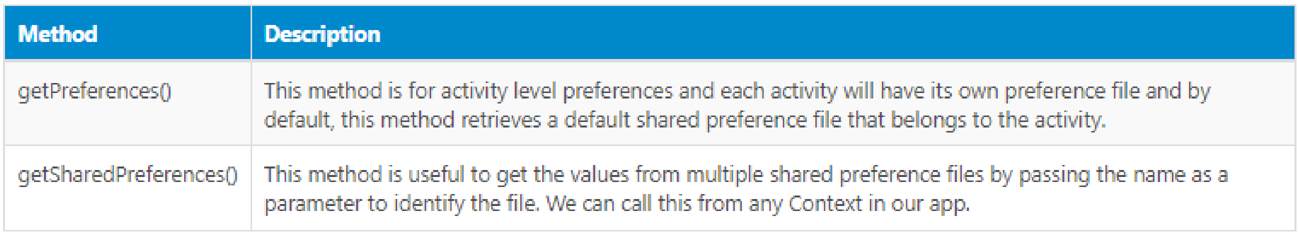
There are three types of Mode in Shared Preference:

- Context.MODE_PRIVATE – default value (Not accessible outside of your application)
- Context.MODE_WORLD_READABLE – readable to other apps
- Context.MODE_WORLD_WRITEABLE – read/write to other apps

Creating Shared Preference:

Need to call : getPreferences(int mode) or getSharedPreferences(String name,int mode) method.

Both of the function return SharedPreferences object to deal with save and get values from preference file.



Method	Description
getPreferences()	This method is for activity level preferences and each activity will have its own preference file and by default, this method retrieves a default shared preference file that belongs to the activity.
getSharedPreferences()	This method is useful to get the values from multiple shared preference files by passing the name as a parameter to identify the file. We can call this from any Context in our app.

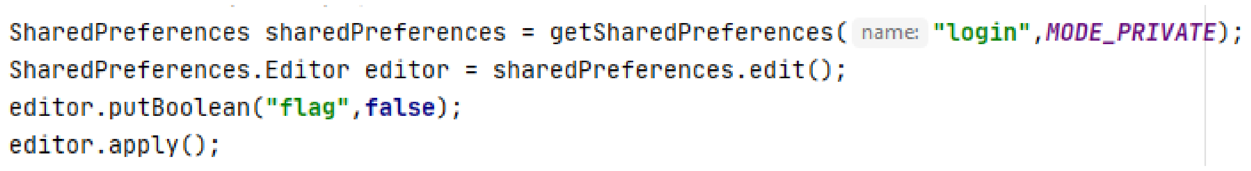
```
SharedPreferences sharedPref =
getSharedPreferences("SharedPrefName",Context.MODE_PRIVATE);
```

How to Write/Save Data in Shared Pref. File

(NOTE: IN following syntax example our Shared Preference Name is “login”)

To write values:

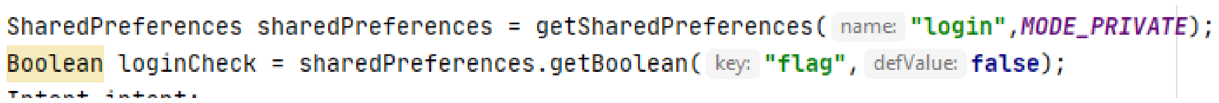
- Call edit()to get a SharedPreferences.Editor.
- Add values with methods such as putBoolean()and putString().
- Commit the new values with commit()
  Ex:



```
SharedPreferences sharedPreferences = getSharedPreferences( name: "login",MODE_PRIVATE);
SharedPreferences.Editor editor = sharedPreferences.edit();
editor.putBoolean("flag",false);
editor.apply();
```

```
SharedPreferences sharedPref = getPreferences(“login”, MODE_PRIVATE);
SharedPreferences.Editor editor = sharedPref.edit();
editor.putBoolean("keyname",true);
editor.putString("keyname","string value");
editor.putInt("keyname",10);
editor.putFloat("keyname","float value");
editor.putLong("keyname","long value");
editor.commit();
```

To Read From Shared Preference



```
SharedPreferences sharedPreferences = getSharedPreferences( name: "login",MODE_PRIVATE);
Boolean loginCheck = sharedPreferences.getBoolean( key: "flag", defValue: false);
Intent intent;
```

```
SharedPreferences pref = getPreferences(“login”, MODE_PRIVATE);
pref.getString("keyname",null);
pref.getInt("keyname",0);
pref.getFloat("keyname",0);
pref.getBoolean("keyname",true);
pref.getLong("keyname",0);
```

Deleting From Shared Preference

```
SharedPreferences pref = getPreferences(“login”, MODE_PRIVATE);
SharedPreferences.Editor editor = pref.edit();
editor.remove("keyname");
editor.commit();
```

Clear From Shared Preference

```
SharedPreferences pref = getPreferences(Context.MODE_PRIVATE);
SharedPreferences.Editor editor = pref.edit();
editor.clear();
editor.commit();
```

Shared Preference Example 1:

AIM: We are going to develop an application in which First we will able to see the splash screen then Login Screen.

By Clicking on login screen, The shared preference Login will be shared in the shared preference

And now User will navigated to the Logout screen .

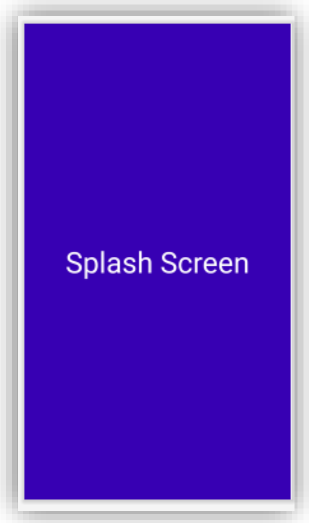
Then Once again If user try to open the app: He will see SPLASH SCREEN – HOME SCREEN that have Logout button

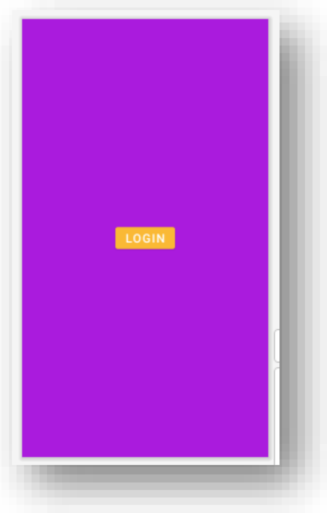
Try to clear your app from phone and try to open it again and check how shared preference works

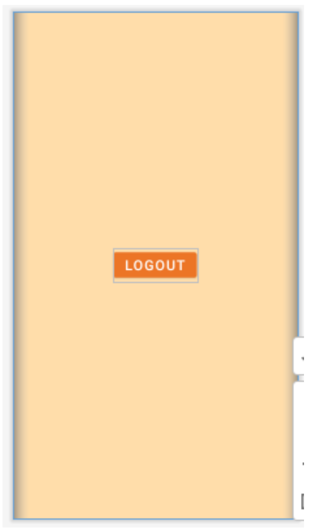
Once user will click on LOGOUT button from home screen then, Shared Preference get cleared.

So, after that If user try to open app again now He must see SPLASH SCREEN – LOGIN SCREEN – HOME SCREEM

- Splash Screen – Login – Home Screen
- SKIP Login Screen When We have already Logged in
- If Not Login then First Shows – Splash Screen – Login – Home Screen







Files for the projects

Activity_main.xml (this contains our Splash Screen)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:background="@color/purple_700"
    android:gravity="center">


    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Splash Screen"
        android:textColor="#FFFFFF"
        android:textSize="45dp"
        />

</LinearLayout>
```

MainActivity.java

```
package com.example.sharedpreferenceexample;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.os.Handler;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                SharedPreferences sharedPreferences =
getSharedPreferences("login",MODE_PRIVATE); // create the sharedpreference
                Boolean loginCheck =
sharedPreferences.getBoolean("flag",false); //


                Intent intent;

                if(loginCheck)
                {
                    intent = new
Intent(MainActivity.this,HomeScreen.class);
                }
                else{

                    intent = new
Intent(MainActivity.this,LoginActivity.class);
                }
                startActivity(intent);
            }
        },4000);
    }
}
```

activity_login.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#AA1BDD"
    android:gravity="center"
    tools:context=".LoginActivity">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Login"
        android:textSize="20dp"
        android:backgroundTint="#FAB935"
        android:id="@+id/loginButton"/>

</LinearLayout>
```

LoginActivity.xml

```
package com.example.sharedpreferenceexample;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class LoginActivity extends AppCompatActivity {

    Button loginButton;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        loginButton = findViewById(R.id.loginButton);

        loginButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                SharedPreferences sharedPreferences =
getSharedPreferences("login",MODE_PRIVATE);
                SharedPreferences.Editor editor = sharedPreferences.edit();
                editor.putBoolean("flag",true);
                editor.putString("Name","LJIET");

                editor.apply();

                Intent intent = new Intent(LoginActivity.this,
HomeScreen.class);
                startActivity(intent);
            }
        });
    }
}
```

activity_home_Screen.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".HomeScreen"
    android:gravity="center"
    android:background="#FFDDAA">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20dp"
        android:text="HELLO"
        android:layout_marginStart="20dp"
        android:layout_marginEnd="20dp"
        android:id="@+id/homeScreenTV"/>


    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Logout"
        android:textSize="20dp"
        android:backgroundTint="#EC7526"
        android:id="@+id/logoutButton"/>


</LinearLayout>
```

HomeScreen.java

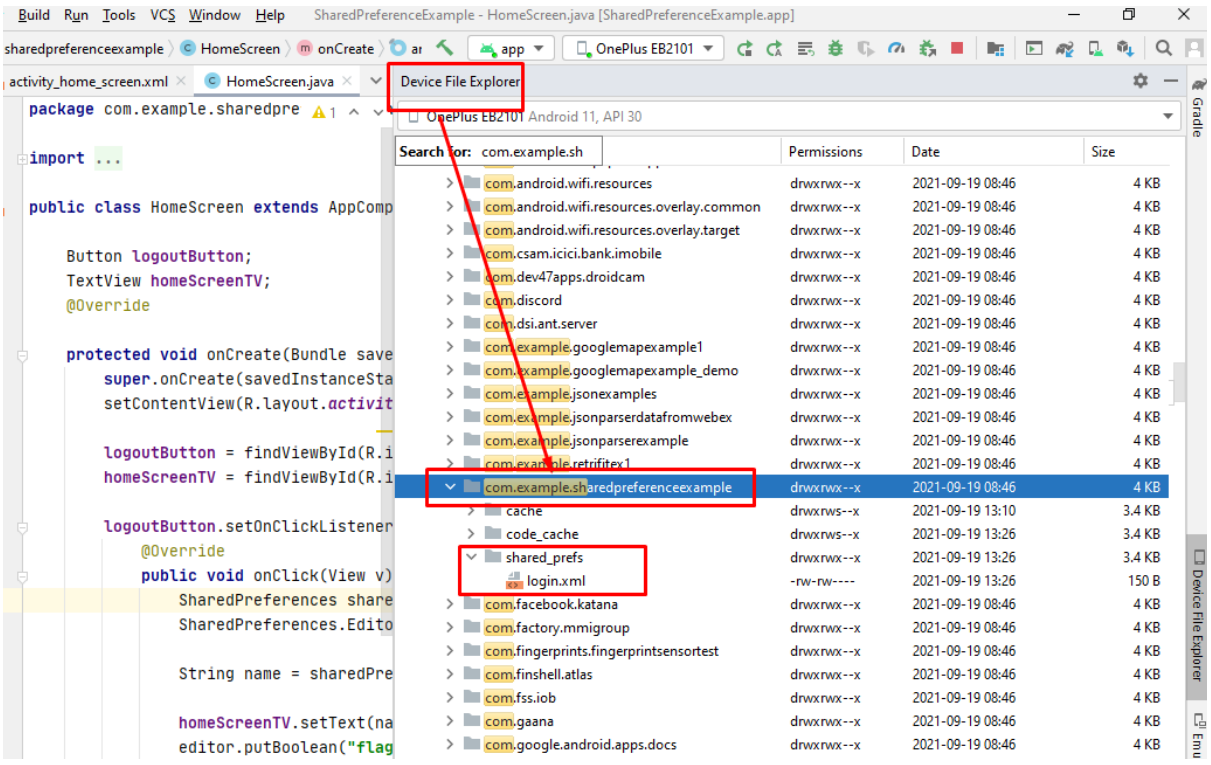
```java
package com.example.sharedpreferenceexample;

import androidx.appcompat.app.AppCompatActivity;

import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class HomeScreen extends AppCompatActivity {

    Button logoutButton;
    TextView homeScreenTV;
    @Override

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home_screen);

        logoutButton = findViewById(R.id.logoutButton);
        homeScreenTV = findViewById(R.id.homeScreenTV);

        logoutButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                SharedPreferences sharedPreferences =
getSharedPreferences("login",MODE_PRIVATE);
                SharedPreferences.Editor editor = sharedPreferences.edit();

                String name = sharedPreferences.getString("Name",null);

                homeScreenTV.setText(name);
                editor.putBoolean("flag",false);
                editor.apply();
            }
        });
    }
}
```

You can check shared preference in Device Explorer window.

Make sure – You have to run app and click on Login button then and only then u can check it

Where to check :

View → Tool Window → Device File Explorer → data → data → Your Package Name


Build Run Tools VCS Window Help
SharedPreferenceExample - HomeScreen.java [SharedPreferenceExample.app]
sharedpreferenceexample 〉 HomeScreen 〉 onCreate 〉 ar
app
OnePlus EB2101
activity_home_screen.xml
HomeScreen.java
Device File Explorer
OnePlus EB2101 Android 11, API 30
Search for: com.example.sh
Gradle
Device File Explorer
Emu

```
package com.example.sharedpre

import ...

public class HomeScreen extends AppComp

    Button logoutButton;
    TextView homeScreenTV;
    @Override

    protected void onCreate(Bundle save
        super.onCreate(savedInstanceSta
        setContentView(R.layout.activit

        logoutButton = findViewById(R.i
        homeScreenTV = findViewById(R.i

        logoutButton.setOnClickListener
            @Override
            public void onClick(View v)
                SharedPreferences share
                SharedPreferences.Edito

                String name = sharedPre

                homeScreenTV.setText(na
                editor.putBoolean("flag
```

	Permissions	Date	Size
com.android.wifi.resources	drwxrwx--x	2021-09-19 08:46	4 KB
com.android.wifi.resources.overlay.common	drwxrwx--x	2021-09-19 08:46	4 KB
com.android.wifi.resources.overlay.target	drwxrwx--x	2021-09-19 08:46	4 KB
com.csam.icici.bank.imobile	drwxrwx--x	2021-09-19 08:46	4 KB
com.dev47apps.droidcam	drwxrwx--x	2021-09-19 08:46	4 KB
com.discord	drwxrwx--x	2021-09-19 08:46	4 KB
com.dsi.ant.server	drwxrwx--x	2021-09-19 08:46	4 KB
com.example.googlemapexample1	drwxrwx--x	2021-09-19 08:46	4 KB
com.example.googlemapexample_demo	drwxrwx--x	2021-09-19 08:46	4 KB
com.example.jsonexamples	drwxrwx--x	2021-09-19 08:46	4 KB
com.example.jsonparserdatafromwebex	drwxrwx--x	2021-09-19 08:46	4 KB
com.example.jsonparserexample	drwxrwx--x	2021-09-19 08:46	4 KB
com.example.retrifitex1	drwxrwx--x	2021-09-19 08:46	4 KB
com.example.sharedpreferenceexample	drwxrwx--x	2021-09-19 08:46	4 KB
cache	drwxrws--x	2021-09-19 13:10	3.4 KB
code_cache	drwxrws--x	2021-09-19 13:26	3.4 KB
shared_prefs	drwxrwx--x	2021-09-19 13:26	3.4 KB
login.xml	-rw-rw----	2021-09-19 13:26	150 B
com.facebook.katana	drwxrwx--x	2021-09-19 08:46	4 KB
com.factory.mmigroup	drwxrwx--x	2021-09-19 08:46	4 KB
com.fingerprints.fingerprintsensortest	drwxrwx--x	2021-09-19 08:46	4 KB
com.finshell.atlas	drwxrwx--x	2021-09-19 08:46	4 KB
com.fss.iob	drwxrwx--x	2021-09-19 08:46	4 KB
com.gaana	drwxrwx--x	2021-09-19 08:46	4 KB
com.google.android.apps.docs	drwxrwx--x	2021-09-19 08:46	4 KB

<u>Program 2:</u>

<u>Shared Preference Example 2</u>

AIM: We are going to develop an application in which First we will able to see the splash screen then Login Screen.

This login screen contains name & Age as well as Remember Me check box.

If you clicked on remember me then and only the data will be appeared in the next Activity. Else after login you can not see Name & Age in the next activity.

What we need to create for this ?

Create: MainActivity – with two edit text, one check box of remember me & one Login Button

Next We will create Second Acivity that contain Two Text Views to show name and Age & One Logout Button.

Files are as below

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="15dp"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_centerInParent="true">

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Enter User Details"
            android:textSize="30dp"
            android:gravity="center"
            android:textColor="@color/black"/>

        <EditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/nameEdt"
            android:inputType="textCapSentences"
            android:hint="Enter Your Name Please"
            android:layout_marginTop="4dp"
            android:ems="10"
            android:textSize="15dp"/>

        <EditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/ageEdt"
            android:inputType="number"
            android:hint="Enter Your Age"
            android:layout_marginTop="4dp"
            android:ems="10"
            android:textSize="15dp"/>
        <CheckBox
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/checkBox"
            android:textSize="13dp"
            android:text="Remeber Me"
            android:layout_marginTop="5dp"/>

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/loginButton"
            android:text="Login"
            android:textSize="15dp"
            android:backgroundTint="@color/teal_700"
            android:layout_gravity="center"
            android:layout_marginTop="4dp"/>
    </LinearLayout>
</RelativeLayout>
```

MainActivity.java

```
package com.example.sharedpreferenceexample2;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    EditText nameEdt, ageEdt;
    Button loginBtn;
    CheckBox rememberMe;
    SharedPreferences sharedPreferences;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        nameEdt = findViewById(R.id.nameEdt);
        loginBtn = findViewById(R.id.loginButton);
        rememberMe = findViewById(R.id.checkBox);
        ageEdt = findViewById(R.id.ageEdt);


        boolean isRemembered;
        sharedPreferences =
getSharedPreferences("Shared_Pref",MODE_PRIVATE);

        isRemembered = sharedPreferences.getBoolean("CHECKBOX",false);
        if(isRemembered)
        {
            Toast.makeText(MainActivity.this,"Data Has Been
Saved",Toast.LENGTH_SHORT).show();
            Intent intent = new
Intent(MainActivity.this,SecondActivity.class);
            startActivity(intent);

        }
        loginBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String name = nameEdt.getText().toString();
                int age =
Integer.parseInt(ageEdt.getText().toString().trim());
                boolean islogin = rememberMe.isChecked();


                SharedPreferences.Editor editor = sharedPreferences.edit();
                editor.putString("NAME",name);
                editor.putInt("Age",age);
                editor.putBoolean("CHECKBOX", islogin);
                editor.apply();

                Toast.makeText(MainActivity.this,"Data Has Been
Saved",Toast.LENGTH_SHORT).show();
                Intent intent = new
Intent(MainActivity.this,SecondActivity.class);
                startActivity(intent);

            }
        });
    }
}
```

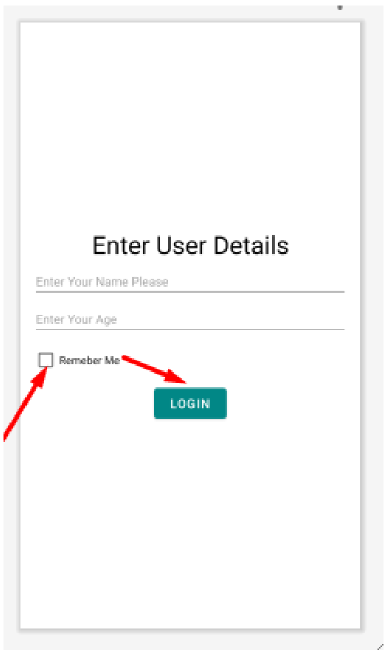
activity_Second.xml

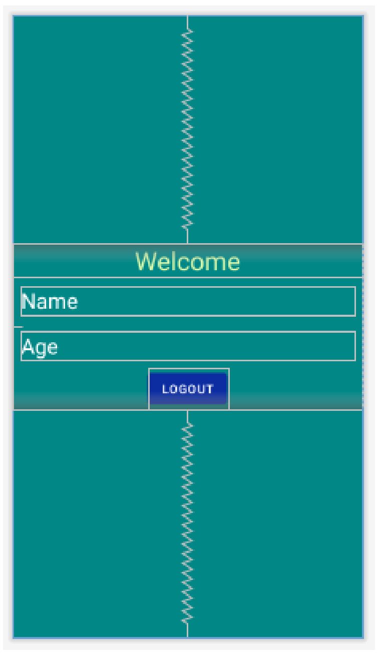
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SecondActivity"
    android:background="@color/teal_700">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_centerInParent="true">

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Welcome"
            android:textSize="30dp"
            android:gravity="center"
            android:textColor="#D9F8AA"
            />
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textColor="@color/white"
            android:textSize="25dp"
            android:text="Name"
            android:layout_margin="10dp"
            android:id="@+id/nameTv"
             />
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textColor="@color/white"
            android:textSize="25dp"
            android:layout_margin="10dp"
            android:text="Age"
            android:id="@+id/ageTv"
            />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/logoutButton"
            android:layout_gravity="center"
            android:text="LOGOUT"
            android:backgroundTint="#0C2CA1"/>



    </LinearLayout>

</RelativeLayout>
```

SecondActivity.java

```
package com.example.sharedpreferenceexample2;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class SecondActivity extends AppCompatActivity {

    Button logoutbtn;
    TextView nameTv, ageTv;
    SharedPreferences sharedPreferences;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);
        logoutbtn = findViewById(R.id.logoutButton);
        nameTv = findViewById(R.id.nameTv);
        ageTv = findViewById(R.id.ageTv);


        sharedPreferences =
getSharedPreferences("Shared_Pref",MODE_PRIVATE);
        String name = sharedPreferences.getString("NAME","");
        nameTv.setText(name);
        int age = sharedPreferences.getInt("Age",0);
        ageTv.setText(""+age);

        logoutbtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                SharedPreferences.Editor  editor =
sharedPreferences.edit();
                editor.clear();
                editor.apply();
                Intent intent = new
Intent(SecondActivity.this,MainActivity.class);
                startActivity(intent);
                finish();
            }
        });

    }
}
```

Output

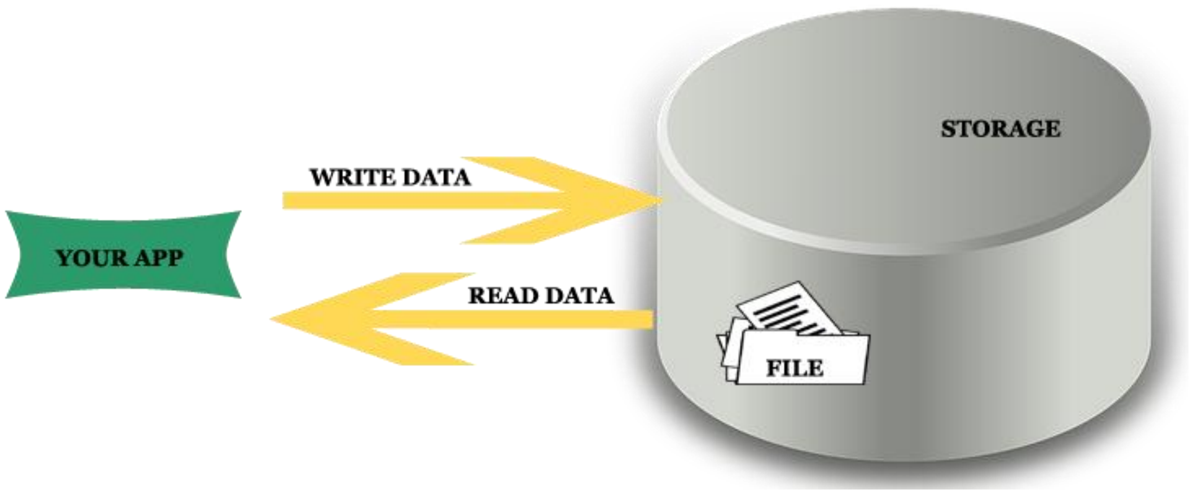




2. Internal Storage:

In android, Internal Storage is useful to store the data files locally on the device's internal memory using a FileOutputStream object. After storing the data files in device internal storage, we can read the data file from the device using a FileInputStream object.

The data files saved in the internal are managed by an android framework and it can be accessed anywhere within the app to read or write data into the file, but it's not possible to access the file from any other app so it's secured. When the user uninstalls the app, automatically these data files will be removed from the device internal storage.



- The stored data in memory is allowed to read and write files.
- When files are stored in internal storage these file can only be accessed by the application itself not by other applications.
- These files in storage exist till the application stays over the device, as you uninstall associated files get removed automatically.
- The files are stored in directory data/data which is followed by the application package name.
- User can explicitly grant the permission to other apps to access files.
- To make the data private i.e you can use MODE_PRIVATE as discussed below about the modes.
- Technique is best suited when data can only be access by the application neither by the user nor by other apps.

- The data is stored in a file which contains data in bit format so it's required to convert data before adding it to a file or before extracting from a file.

How to write in File of internal storage:

```
String File_Name= "Demo.txt"; //gives file name
String Data="Hello!!"; //define data

FileOutputStream fileobj = openFileOutput( File_Name, Context.MODE_PRIVATE);
byte[] ByteArray = Data.getBytes(); //Converts into bytes stream
fileobj.write(ByteArray); //writing to file
fileobj.close(); //File closed
```

How to read from file

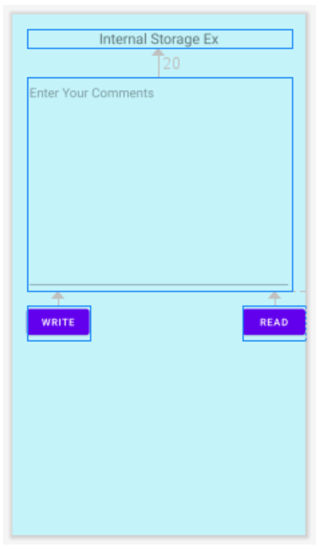
```
String FILENAME = "user_details";
FileInputStream fstream = openFileInput(FILENAME);
StringBuffer sbuffer = new StringBuffer();
int i;
while ((i = fstream.read())!= -1){
    sbuffer.append((char)i);
}
fstream.close();
```

<u>Android Intenral Storage Example</u>

Aim: Here, we are going to make an activity with EDIT TEXT and TWO button write and read.

- We will Enter Data in Edit Text
- Write it – it will create a file and stored in internal storage

• Read it- Read the data from internal storage



Program Files are as below

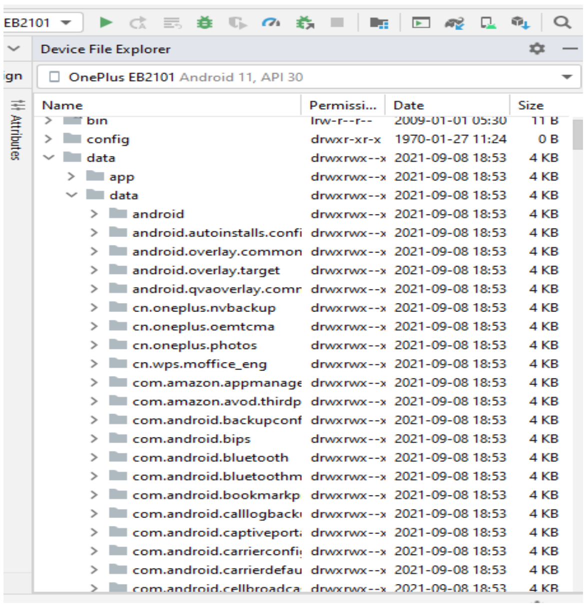
Activity For Internal Storage.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".InternalStorageEx"
    android:background="#C4F4FA">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="20dp"
        android:text="Internal Storage Ex"
        android:gravity="center"
        android:id="@+id/internalStorageTextView"
        android:layout_margin="20dp"
        />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="300dp"
        android:id="@+id/internalStorageEditText"
        android:layout_below="@+id/internalStorageTextView"
        android:layout_margin="20dp"
        android:hint="Enter Your Comments"
        android:inputType="text"
        android:gravity="fill_horizontal"
        />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Write"
        android:id="@+id/internalStorageWriteDataBtn"
        android:layout_below="@+id/internalStorageEditText"
        android:layout_marginStart="20dp"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Read"
        android:id="@+id/internalStorageReadDataBtn"
        android:layout_below="@+id/internalStorageEditText"
        android:layout_marginStart="20dp"
        android:layout_alignParentEnd="true"/>
</RelativeLayout>
```

InternalStorage.Java

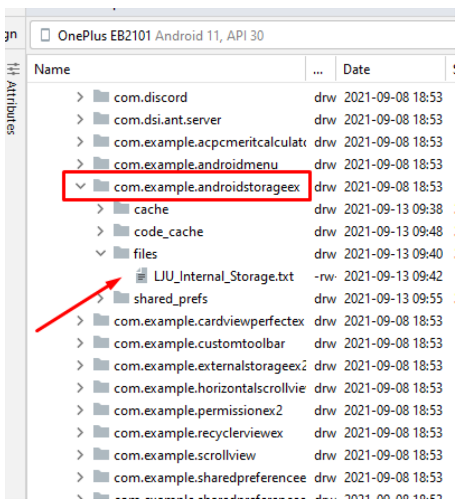
```
package com.example.androidstorageex;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;

public class InternalStorageEx extends AppCompatActivity {

    EditText internalEdt;
    Button writeData, readData;
    String filename = "LJU_Internal_Storage_Live_Prac.txt";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_internal_storage_ex);

        internalEdt = findViewById(R.id.internalStorageEditText);
        writeData = findViewById(R.id.internalStorageWriteDataBtn);
        readData = findViewById(R.id.internalStorageReadDataBtn);

        writeData.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
               try{
                    FileOutputStream fos =
openFileOutput(filename,MODE_PRIVATE);
                    fos.write(internalEdt.getText().toString().getBytes());
                    fos.close();
                    internalEdt.setText("");

                }catch (FileNotFoundException e){
                    e.printStackTrace();
                }
                catch (IOException e)
                {
                    e.printStackTrace();
                }
            }
        });

        readData.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                StringBuffer stringBuffer = new StringBuffer();
                try {
                    FileInputStream fis = openFileInput(filename);

                    int i = 0;
                    while((i = fis.read())!=-1)
                    {
                        stringBuffer.append((char)i);
                    }
                    fis.close();
                } catch (FileNotFoundException e) {
                    e.printStackTrace();
                } catch (IOException e) {
                    e.printStackTrace();
                }

                internalEdt.setText(stringBuffer);
            }
        });

    }
}
```

How To Check Output Click on View→ Tool Window →Device File Explorer →data→data →Your Package Name →Get File →Your Created File or Folder over there



OnePlus EB2101 Android 11, API 30

Name	Permissi...	Date	Size
bin	lrw-r--r--	2009-01-01 05:30	11 B
config	drwxr-xr-x	1970-01-27 11:24	0 B
data	drwxrwx--x	2021-09-08 18:53	4 KB
app	drwxrwx--x	2021-09-08 18:53	4 KB
data	drwxrwx--x	2021-09-08 18:53	4 KB
android	drwxrwx--x	2021-09-08 18:53	4 KB
android.autoinstalls.confi	drwxrwx--x	2021-09-08 18:53	4 KB
android.overlay.common	drwxrwx--x	2021-09-08 18:53	4 KB
android.overlay.target	drwxrwx--x	2021-09-08 18:53	4 KB
android.qvaoverlay.comr	drwxrwx--x	2021-09-08 18:53	4 KB
cn.oneplus.nvbackup	drwxrwx--x	2021-09-08 18:53	4 KB
cn.oneplus.oemtcma	drwxrwx--x	2021-09-08 18:53	4 KB
cn.oneplus.photos	drwxrwx--x	2021-09-08 18:53	4 KB
cn.wps.moffice_eng	drwxrwx--x	2021-09-08 18:53	4 KB
com.amazon.appmanage	drwxrwx--x	2021-09-08 18:53	4 KB
com.amazon.avod.thirdp	drwxrwx--x	2021-09-08 18:53	4 KB
com.android.backupconf	drwxrwx--x	2021-09-08 18:53	4 KB
com.android.bips	drwxrwx--x	2021-09-08 18:53	4 KB
com.android.bluetooth	drwxrwx--x	2021-09-08 18:53	4 KB
com.android.bluetoothm	drwxrwx--x	2021-09-08 18:53	4 KB
com.android.bookmarkp	drwxrwx--x	2021-09-08 18:53	4 KB
com.android.calllogback	drwxrwx--x	2021-09-08 18:53	4 KB
com.android.captiveport	drwxrwx--x	2021-09-08 18:53	4 KB
com.android.carrierconfi	drwxrwx--x	2021-09-08 18:53	4 KB
com.android.carrierdefau	drwxrwx--x	2021-09-08 18:53	4 KB
com.android.cellbroadca	drwxrwx--x	2021-09-08 18:53	4 KB

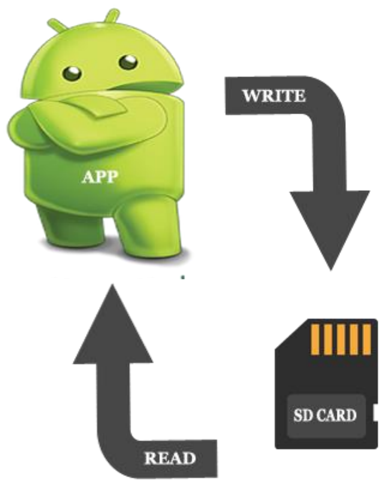


OnePlus EB2101 Android 11, API 30

Name	...	Date
com.discord	drw	2021-09-08 18:53
com.dsi.ant.server	drw	2021-09-08 18:53
com.example.acpcmeritcalculat	drw	2021-09-08 18:53
com.example.androidmenu	drw	2021-09-08 18:53
com.example.androidstorageex	drw	2021-09-08 18:53
cache	drw	2021-09-13 09:38
code_cache	drw	2021-09-13 09:48
files	drw	2021-09-13 09:40
LJU_Internal_Storage.txt	-rw-	2021-09-13 09:42
shared_prefs	drw	2021-09-13 09:55
com.example.cardviewperfectex	drw	2021-09-08 18:53
com.example.customtoolbar	drw	2021-09-08 18:53
com.example.externalstorageex2	drw	2021-09-08 18:53
com.example.horizontalscrollvie	drw	2021-09-08 18:53
com.example.permissionex2	drw	2021-09-08 18:53
com.example.recyclerviewex	drw	2021-09-08 18:53
com.example.scrollview	drw	2021-09-08 18:53
com.example.sharedpreferencee	drw	2021-09-08 18:53

3. External Storage in Android

- In android, External Storage is useful to store the data files publically on the shared external storage using the FileOutputStream object. After storing the data files on external storage, we can read the data file from external storage media using a FileInputStream object.
- The data is stored in a file specified by the user itself and user can access these file. These files are only accessible till the application exits or you have SD card mounted on your device.



External Storage Availability In Android Studio

- To avoid crashing app it is required to check before whether storage SD card is available for read & write operations. getExternalStorageState() method is used to determine the state of the storage media i.e SD card is mounted, is it readable , it is writable etc.. all this kind of information.

```
boolean Available= false;
boolean Readable= false;
String state = Environment.getExternalStorageState();
if(Environment.MEDIA_MOUNTED.equals(state)){
    // Both Read and write operations available
    Available= true;
} else if (Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)){
    // Only Read operation available
    Available= true;
    Readable= true;
} else {
    // SD card not mounted
    Available = false;
}
```

<u>Methods to Store Data In Android</u>:

- getExternalStorageDirectory() – Older way to access external storage in API Level less than 7. It is absolute now and not recommended. It directly get the reference to the root directory of your external storage or SD Card.
- getExternalFilesDir(String type) – It is recommended way to enable us to create private files specific to app and files are removed as app is uninstalled. Example is app private data.
- getExternalStoragePublicDirectory() : This is current recommended way that enable us to keep files public and are not deleted with the app uninstallation. Example images clicked by the camera exists even we uninstall the camera app.

(Do not forget give permission in Manifest File)

Permissions:

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
  <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

How to write a File to external Storage?

```
String FILENAME = "user_details";
String name = "suresh";

File folder = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLOADS);
File myFile = new File(folder, FILENAME);
FileOutputStream fstream = new FileOutputStream(myFile);
fstream.write(name.getBytes());
fstream.close();
```

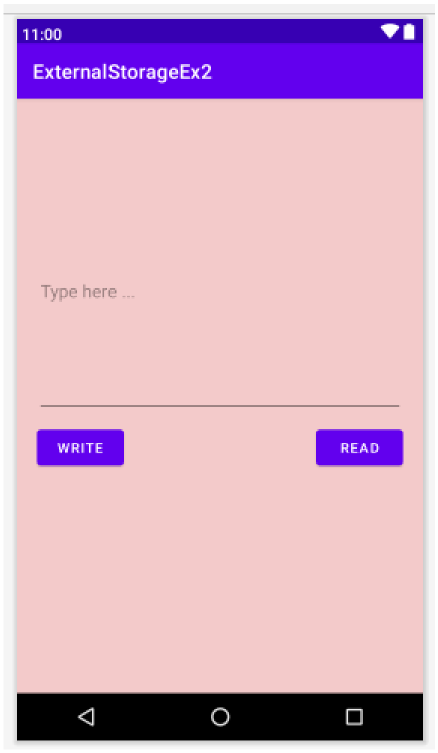
How to read a file from External Storage ?

```
String FILENAME = "user_details";
File folder = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLOADS);
File myFile = new File(folder, FILENAME);
FileInputStream fstream = new FileInputStream(myFile);
StringBuffer sbuffer = new StringBuffer();
int i;
while ((i = fstream.read())!= -1){
    sbuffer.append((char)i);
}
fstream.close();
```

Practical For External Storage

AIM:

- We will Enter Data in Edit Text
- Write it – it will create a file and stored in External storage
- Read it- Read the data from External storage



Activity_main.xml File

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#F3CACA"
    android:padding="20dp"
    tools:context=".MainActivity">


    <EditText
        android:layout_width="match_parent"
        android:layout_height="250dp"
        android:id="@+id/externalStorageEdt"
        android:hint="Type here ... "
        android:layout_marginTop="50dp"
        android:layout_gravity="fill_horizontal"
        />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Write"
        android:layout_below="@+id/externalStorageEdt"
        android:layout_marginTop="10dp"
        android:id="@+id/writeDataButton"
        />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Read"
        android:layout_below="@+id/externalStorageEdt"
        android:layout_alignParentEnd="true"
        android:layout_marginTop="10dp"
        android:id="@+id/readDataButton"
        />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/showDatatextView"
        android:layout_below="@+id/readDataButton"/>
</RelativeLayout>
```

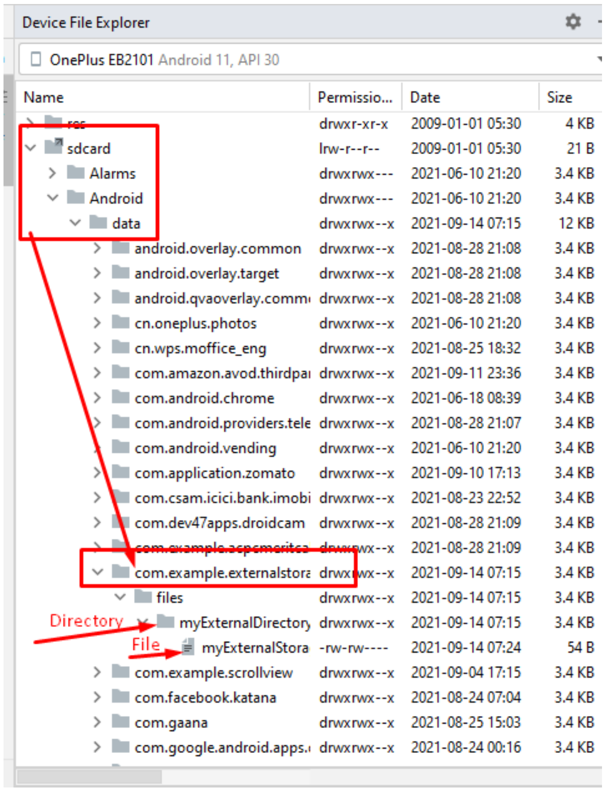
MainActivity.java

```java
package com.example.externalstorageex2;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.os.Environment;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.IOException;

public class MainActivity extends AppCompatActivity {

    Button readbtn, writeBtn;
    EditText externalDataEdt;
    TextView showData;
    String filename="";
    String filepath="";
    String filecontent = "";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        readbtn = findViewById(R.id.readDataButton);
        writeBtn = findViewById(R.id.writeDataButton);
        externalDataEdt = findViewById(R.id.externalStorageEdt);
        showData = findViewById(R.id.showDatatextView);

        filename = "myExternalStorage_Live.txt";
        filepath = "myExternalDirectory_Live";
        if(!isExternalStorageAvailableForRW())
        {
            writeBtn.setEnabled(false);
        }
        writeBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                showData.setText("");
                filecontent = externalDataEdt.getText().toString().trim();
                if(!filecontent.equals(""))
                {
                    File myExternalFile = new
File(getExternalFilesDir(filepath),filename);
                    FileOutputStream fos = null;
                    try {
                        fos = new FileOutputStream(myExternalFile);
                        fos.write(filecontent.getBytes());
                    } catch (FileNotFoundException e) {
                        e.printStackTrace();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                    externalDataEdt.setText("");
                    Toast.makeText(MainActivity.this, "Information Saved
Succesfully", Toast.LENGTH_SHORT).show();

                }
                else {
                    Toast.makeText(MainActivity.this,"Text FIled Can not be
empty",Toast.LENGTH_SHORT).show();
                }
            }
        });

        readbtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                FileReader fr = null;
                File myExternalFile = new
File(getExternalFilesDir(filepath),filename);
                StringBuilder stringBuilder = new StringBuilder();
                try {
                    fr = new FileReader(myExternalFile);
                    BufferedReader br = new BufferedReader(fr);
                    String line = br.readLine();
                    while(line != null)
                    {
                        stringBuilder.append(line).append("\n");
                        line = br.readLine();
                    }
                } catch (FileNotFoundException e) {
                    e.printStackTrace();
                } catch (IOException e) {
                    e.printStackTrace();
                }
                finally {
                    String fileContents =  stringBuilder.toString();
                    showData.setText(fileContents);
                }
            }
        });
```

```
    }

    private boolean isExternalStorageAvailableForRW() {

        String externalStorageState =
Environment.getExternalStorageState();
        if(externalStorageState.equals(Environment.MEDIA_MOUNTED))
        {
            return true;

        }
        else{
            return  false;
        }
    }
}
```

Where to find the External Storage File

- Storage path:
- Click on View → Tool Window → Device File Explorer → SD Card → Android → data → Your Package Name → Files → Your Created File or Folder over there

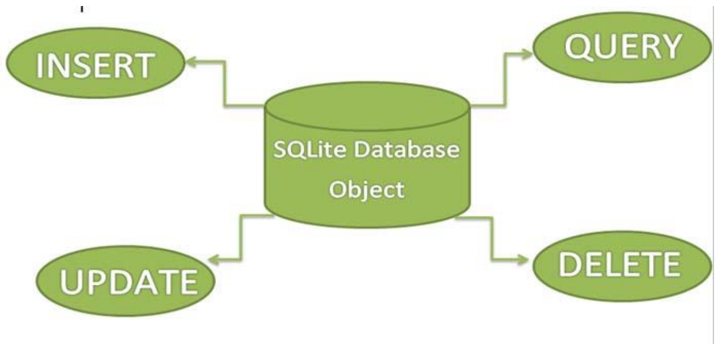


OnePlus EB2101 Android 11, API 30

Name	Permissio...	Date	Size
res	drwxr-xr-x	2009-01-01 05:30	4 KB
sdcard	lrw-r--r--	2009-01-01 05:30	21 B
Alarms	drwxrwx---	2021-06-10 21:20	3.4 KB
Android	drwxrwx---	2021-06-10 21:20	3.4 KB
data	drwxrwx--x	2021-09-14 07:15	12 KB
android.overlay.common	drwxrwx--x	2021-08-28 21:08	3.4 KB
android.overlay.target	drwxrwx--x	2021-08-28 21:08	3.4 KB
android.qvaoverlay.comm	drwxrwx--x	2021-08-28 21:08	3.4 KB
cn.oneplus.photos	drwxrwx--x	2021-06-10 21:20	3.4 KB
cn.wps.moffice_eng	drwxrwx--x	2021-08-25 18:32	3.4 KB
com.amazon.avod.thirdpa	drwxrwx--x	2021-09-11 23:36	3.4 KB
com.android.chrome	drwxrwx--x	2021-06-18 08:39	3.4 KB
com.android.providers.tele	drwxrwx--x	2021-08-28 21:07	3.4 KB
com.android.vending	drwxrwx--x	2021-06-10 21:20	3.4 KB
com.application.zomato	drwxrwx--x	2021-09-10 17:13	3.4 KB
com.csam.icici.bank.imobi	drwxrwx--x	2021-08-23 22:52	3.4 KB
com.dev47apps.droidcam	drwxrwx--x	2021-08-28 21:09	3.4 KB
com.example.aspcmeritca	drwxrwx--x	2021-08-28 21:09	3.4 KB
com.example.externalstora	drwxrwx--x	2021-09-14 07:15	3.4 KB
files	drwxrwx--x	2021-09-14 07:15	3.4 KB
myExternalDirectory	drwxrwx--x	2021-09-14 07:15	3.4 KB
myExternalStora	-rw-rw----	2021-09-14 07:24	54 B
com.example.scrollview	drwxrwx--x	2021-09-04 17:15	3.4 KB
com.facebook.katana	drwxrwx--x	2021-08-24 07:04	3.4 KB
com.gaana	drwxrwx--x	2021-08-25 15:03	3.4 KB
com.google.android.apps.	drwxrwx--x	2021-08-24 00:16	3.4 KB

SQLite Storage in Android

- SQLite is a Structure query base database, open source, light weight, no network access and standalone database. It support embedded relational database features.
- Generally, in our android applications Shared Preferences, Internal Storage and External Storage options are useful to store and maintain a small amount of data.
- In case, if we want to deal with large amounts of data, then SQLite database is the preferable option to store and maintain the data in a structured format.
- By default, Android comes with built-in SQLite Database support so we don't need to do any configurations.
- Just like we save the files on the device's internal storage, Android stores our database in a private disk space that's associated with our application and the data is secure, because by default this area is not accessible to other applications.
- The package android.database.sqlite contains all the required APIs to use an SQLite database in our android applications.



SQLiteHelper Class

- For creating, updating and other operations you need to create a subclass or SQLiteOpenHelper class.
- SQLiteOpenHelper is a helper class to manage database creation and version management.
- It provides two methods onCreate(SQLiteDatabase db), onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion).

- The SQLiteOpenHelper is responsible for opening database if exist, creating database if it does not exists and upgrading if required.
- The SQLiteOpenHelper only require the DATABASE_NAME to create database.
- After extending SQLiteOpenHelper you will need to implement its methods onCreate, onUpgrade and constructor.

onCreate(SQLiteDatabase sqLiteDatabase)

- method is called only once throughout the application lifecycle. It will be called whenever there is a first call to getReadableDatabase() or getWritableDatabase() function available in super SQLiteOpenHelper class.
- So SQLiteOpenHelper class call the onCreate() method after creating database and instantiate SQLiteDatabase object.

onUpgrade(SQLiteDatabase db,int oldVersion, int newVersion)

- Method is only called whenever there is a updation in existing version.
- So to update a version we have to increment the value of version variable passed in the superclass constructor.

How to create Database & Table

Make a class that extends SQLiteHelper

Then over ride methods onCreate & onUpgrade

```
public class DbHandler extends SQLiteOpenHelper {
    private static final int DB_VERSION = 1;
    private static final String DB_NAME = "usersdb";
    private static final String TABLE_Users = "userdetails";
    private static final String KEY_ID = "id";
    private static final String KEY_NAME = "name";
    private static final String KEY_LOC = "location";
    private static final String KEY_DESG = "designation";
    public DbHandler(Context context){
        super(context,DB_NAME, null, DB_VERSION);
    }
    @Override
    public void onCreate(SQLiteDatabase db){
        String CREATE_TABLE = "CREATE TABLE " + TABLE_Users + "("
                + KEY_ID + " INTEGER PRIMARY KEY
AUTOINCREMENT," + KEY_NAME + " TEXT,"
```

```
                + KEY_LOC + " TEXT,"
                + KEY_DESG + " TEXT"+ ")";
        db.execSQL(CREATE_TABLE);
    }
    @Override
    public void onUpgrade(SQLiteDatabase
db, int oldVersion, int newVersion){
        // Drop older table if exist
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_Users);
        // Create tables again
        onCreate(db);
    }
}
```

How to Insert Data into Table

```
//Get the Data Repository in write mode
SQLiteDatabase db = this.getWritableDatabase();
//Create a new map of values, where column names are the keys
ContentValues cValues = new ContentValues();
cValues.put(KEY_NAME, name);
cValues.put(KEY_LOC, location);
cValues.put(KEY_DESG, designation);
// Insert the new row, returning the primary key value of the new row
long newRowId = db.insert(TABLE_Users,null, cValues);
```

How to Read Data from Table

```
//Get the Data Repository in write mode
SQLiteDatabase db = this.getWritableDatabase();
Cursor cursor =
db.query(TABLE_Users, new String[]{KEY_NAME, KEY_LOC, KEY_DESG}, KEY_ID
+ "=?",new String[]{String.valueOf(userid)},null, null, null, null);
```

How to Update record from Table

```
//Get the Data Repository in write mode
SQLiteDatabase db = this.getWritableDatabase();
ContentValues cVals = new ContentValues();
cVals.put(KEY_LOC, location);
cVals.put(KEY_DESG, designation);
```

```
int count = db.update(TABLE_Users, cVals, KEY_ID+" = 
?",new String[]{String.valueOf(id)});
```

How to Delete record from Table

```
//Get the Data Repository in write mode
SQLiteDatabase db = this.getWritableDatabase();
db.delete(TABLE_Users, KEY_ID+" = 
?",new String[]{String.valueOf(userid)});
```

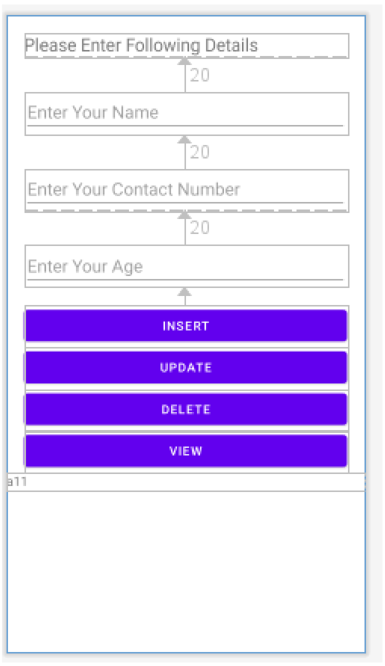
How to Fetch/View record from Table

```
SQLiteDatabase db = this.getWritableDatabase();

Cursor cursor = db.rawQuery("Select * from Users_live", null);
return cursor;
```

Practical For SQLite Database

AIM:

We are going to make a Layout that contains 4 Edit Text & 4 buttons for Insert Update Delete & View. We have one TextView (to show data) to view the final inserted data.



Files are as below.

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Please Enter Following Details"
        android:id="@+id/titleTextView"
        android:textSize="20dp"
        android:layout_margin="20dp" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="20dp"
        android:hint="Enter Your Name"
        android:id="@+id/nameEdt"
        android:inputType="textPersonName"
        android:layout_below="@+id/titleTextView"
        android:layout_margin="20dp"/>

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="20dp"
        android:hint="Enter Your Contact Number"
        android:id="@+id/contactEdt"
        android:inputType="number"
        android:layout_below="@+id/nameEdt"
        android:layout_margin="20dp"/>

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="20dp"
        android:hint="Enter Your Age"
        android:id="@+id/ageEdt"
        android:inputType="textPersonName"
        android:layout_below="@+id/contactEdt"
        android:layout_margin="20dp"
        android:layout_marginEnd="20dp"/>

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/insertButton"
        android:layout_below="@+id/ageEdt"
        android:layout_marginStart="20dp"
        android:layout_marginEnd="20dp"
        android:text="Insert"/>

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/updateButton"
        android:layout_below="@+id/insertButton"
        android:layout_marginStart="20dp"
        android:layout_marginEnd="20dp"
        android:text="Update"/>
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/deleteButton"
        android:layout_below="@+id/updateButton"
        android:layout_marginStart="20dp"
        android:layout_marginEnd="20dp"
        android:text="Delete"/>
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/viewButton"
        android:layout_below="@+id/deleteButton"
        android:layout_marginStart="20dp"
        android:layout_marginEnd="20dp"
        android:text="View"/>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="all"
        android:textSize="15dp"
        android:id="@+id/showData"
        android:layout_below="@+id/viewButton"
        />
</RelativeLayout>
```

DatabaseHelper.java

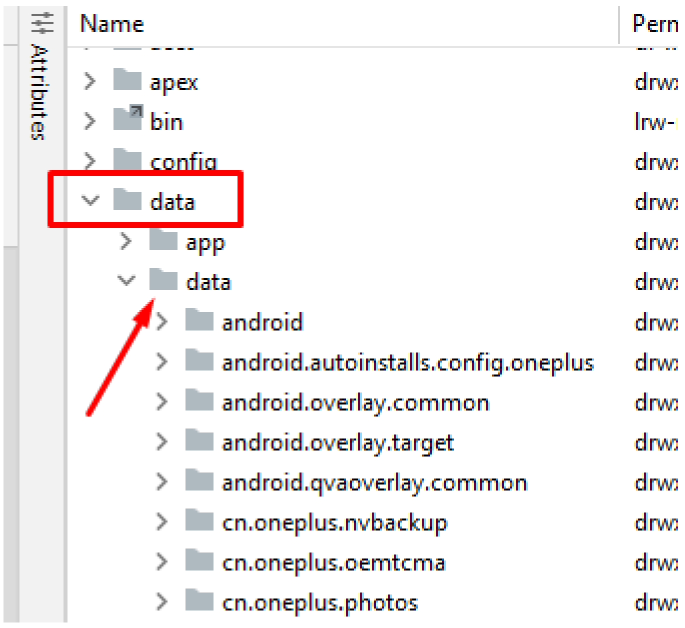
```
package com.example.sqlitedbstorageex2;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

import androidx.annotation.Nullable;

public class DatabaseHelper extends SQLiteOpenHelper {
    private static final String DATABASE_NAME= "db_user_data_LIVE";
    private static final String TABLE_User= "Users_live";
    private static final String KEY_ID= "id";
    private static final String KEY_NAME= "name";
    private static final String KEY_CONTACT = "contact";
    private static final String KEY_AGE = "age";
    private static final int DATABASE_VERSION = 1;

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("create Table Users_live (name TEXT primary key, contact
TEXT, age TEXT )");

    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {

        db.execSQL("drop Table if exists Users_live");
    }
    public boolean insertUserData(String name, String contact, String age)
    {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put("name",name);
        contentValues.put("contact",contact);
        contentValues.put("age",age);
        long result = db.insert("Users_live",null,contentValues);
        if(result==-1)
        {
            return false;
        }
        else
        {
            return true;
        }
    }
    public boolean updateUserData(String name, String contact, String age)
    {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();

        contentValues.put("contact",contact);
        contentValues.put("age",age);

        Cursor cursor = db.rawQuery("Select * from Users_live where name =
?", new String[]{name});

        if(cursor.getCount()>0)
        {

            long result = db.update("Users_live",contentValues, "name = ?",
new String[]{name});
            if(result==-1)
            {
                return false;
            }
            else
            {
                return true;
            }
        }
        else{
            return false;
        }

    }


    public boolean deleteUserData(String name)
    {
        SQLiteDatabase db = this.getWritableDatabase();

        Cursor cursor = db.rawQuery("Select * from Users_live where name =
?", new String[]{name});

        if(cursor.getCount()>0)
        {

            long result = db.delete("Users_live", "name = ?", new
String[]{name});
            if(result==-1)
            {
                return false;
            }
            else
            {
                return true;
            }
        }
        else{
            return false;
        }

    }

    public Cursor getUserData()
    {
        SQLiteDatabase db = this.getWritableDatabase();

        Cursor cursor = db.rawQuery("Select * from Users_live", null);
        return cursor;
    }

}
```

MainActivity.java

```
package com.example.sqlitedbstorageex2;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    //declare the variable
    Button insertBtn, deleteBtn, updateBtn, viewBtn;
    EditText nameEdt, ageEdt, contactEdt;
    TextView showDataTV;
    DatabaseHelper db;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // map the variables
        insertBtn = findViewById(R.id.insertButton);
        deleteBtn = findViewById(R.id.deleteButton);
        updateBtn = findViewById(R.id.updateButton);
        viewBtn = findViewById(R.id.viewButton);
        nameEdt = findViewById(R.id.nameEdt);
        ageEdt = findViewById(R.id.ageEdt);
        contactEdt = findViewById(R.id.contactEdt);
        showDataTV = findViewById(R.id.showData);


        db = new DatabaseHelper(this);

        insertBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String nameTxt = nameEdt.getText().toString();
                String contactTxt = contactEdt.getText().toString();
                String ageTxt = ageEdt.getText().toString();


                Boolean checkInsertData =
db.insertUserData(nameTxt,contactTxt,ageTxt);
                if(checkInsertData)
                {
                    Toast.makeText(MainActivity.this,"Insertion Succesfully
Done",Toast.LENGTH_LONG).show();
                }
                else
                {
                    Toast.makeText(MainActivity.this,"Insertion
Failed",Toast.LENGTH_LONG).show();
                }
            }
        });

        updateBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String nameTxt = nameEdt.getText().toString();
                String contactTxt = contactEdt.getText().toString();
                String ageTxt = ageEdt.getText().toString();


                Boolean checkUpdateData =
db.updateUserData(nameTxt,contactTxt,ageTxt);
                if(checkUpdateData)
                {
                    Toast.makeText(MainActivity.this,"Update Succesfully
Done",Toast.LENGTH_LONG).show();
                }
                else
                {
                    Toast.makeText(MainActivity.this,"Update
Failed",Toast.LENGTH_LONG).show();
                }
            }
        });

        deleteBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String nameTxt = nameEdt.getText().toString();
                Boolean checkDeleteData = db.deleteUserData(nameTxt);
                if(checkDeleteData)
                {
                    Toast.makeText(MainActivity.this,"Deletion Succesfully
Done",Toast.LENGTH_LONG).show();
                }
                else
                {
                    Toast.makeText(MainActivity.this,"Deletion
Failed",Toast.LENGTH_LONG).show();
                }
            }
        });

        viewBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Cursor cursor = db.getUserData();
                if(cursor.getCount()==0)
                {
                    Toast.makeText(MainActivity.this,"No Data
Exists",Toast.LENGTH_LONG).show();
                    return;
                }
                else{
                    StringBuffer stringBuffer = new StringBuffer();
                    while(cursor.moveToNext())
                    {
                        stringBuffer.append("Name :
"+cursor.getString(0)+"\n");
                        stringBuffer.append("Number :
"+cursor.getString(1)+"\n");
                        stringBuffer.append("Age :
"+cursor.getString(2)+"\n\n");
                    }

                    showDataTV.setText(stringBuffer.toString());
                }
            }
        });

    }
}
```

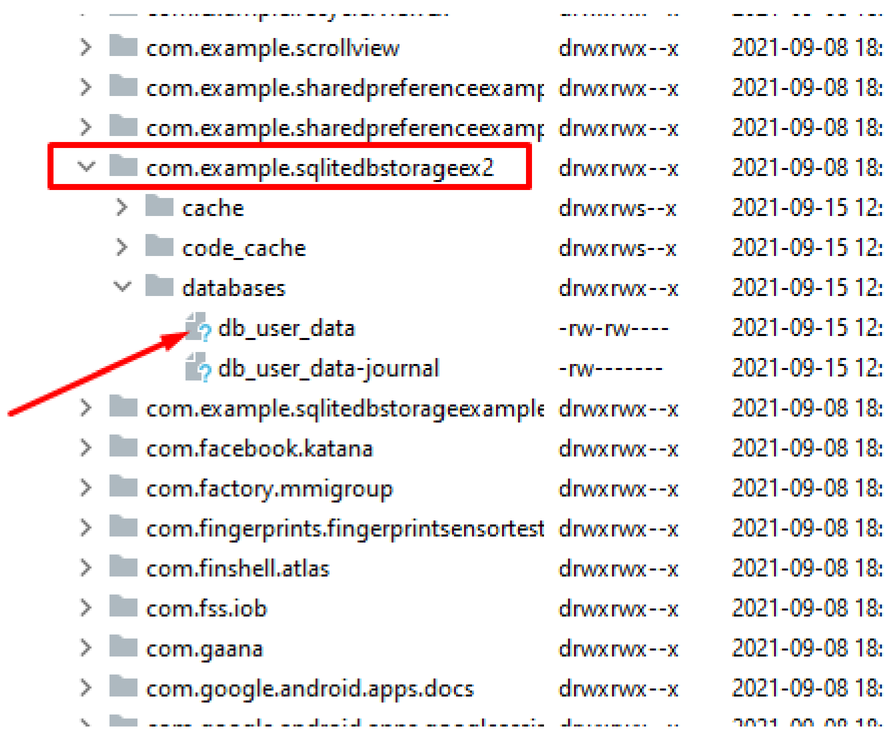
How To Check Database ?

Storage Path:



```
Attributes   Name                                        Pern
  > apex                                                  drw
  > bin                                                   lrw-
  > config                                                drw
  v data                                                  drw
      > app                                               drw
      v data                                              drw
          > android                                       drw
          > android.autoinstalls.config.oneplus           drw
          > android.overlay.common                        drw
          > android.overlay.target                        drw
          > android.qvaoverlay.common                     drw
          > cn.oneplus.nvbackup                           drw
          > cn.oneplus.oemtcma                            drw
          > cn.oneplus.photos                             drw
```

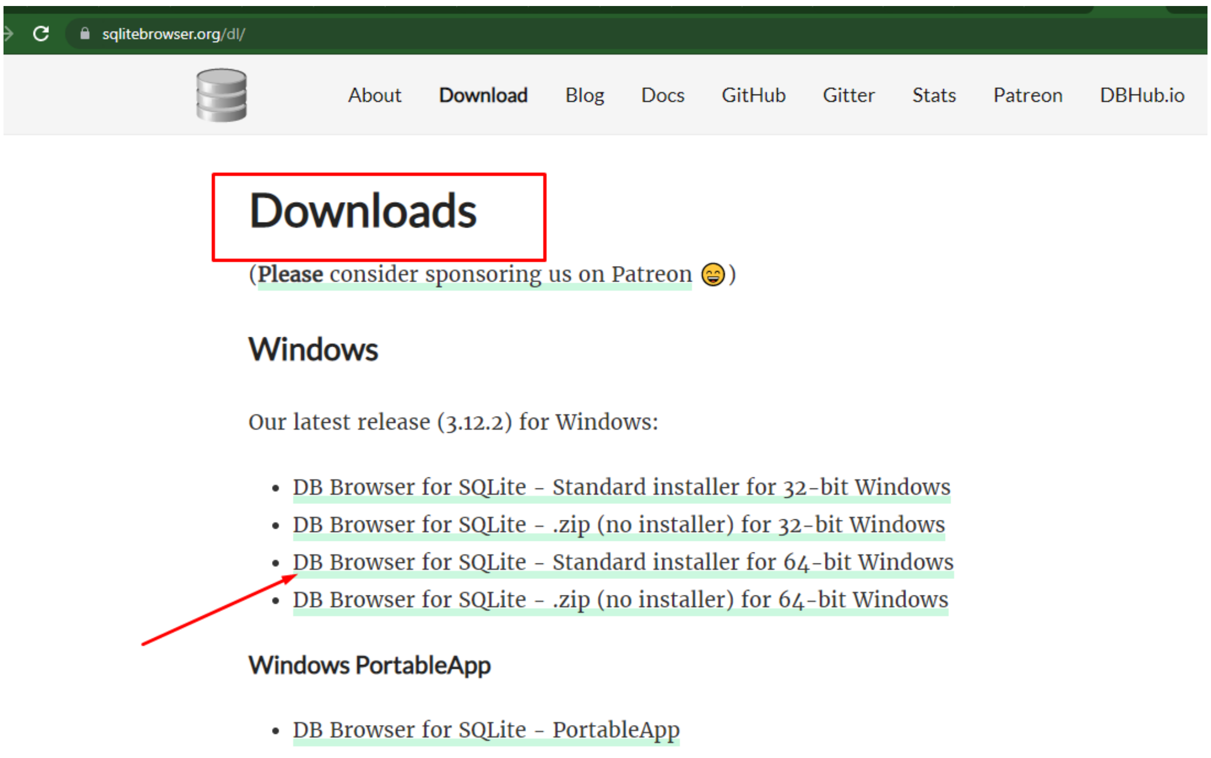
Then find your package name



```
> com.example.scrollview                     drwxrwx--x   2021-09-08 18:
> com.example.sharedpreferenceexamp          drwxrwx--x   2021-09-08 18:
> com.example.sharedpreferenceexamp          drwxrwx--x   2021-09-08 18:
v com.example.sqlitedbstorageex2             drwxrwx--x   2021-09-08 18:
    > cache                                  drwxrws--x   2021-09-15 12:
    > code_cache                             drwxrws--x   2021-09-15 12:
    v databases                              drwxrwx--x   2021-09-15 12:
        db_user_data                         -rw-rw----   2021-09-15 12:
        db_user_data-journal                 -rw-------   2021-09-15 12:
> com.example.sqlitedbstorageexample         drwxrwx--x   2021-09-08 18:
> com.facebook.katana                        drwxrwx--x   2021-09-08 18:
> com.factory.mmigroup                       drwxrwx--x   2021-09-08 18:
> com.fingerprints.fingerprintsensortest     drwxrwx--x   2021-09-08 18:
> com.finshell.atlas                         drwxrwx--x   2021-09-08 18:
> com.fss.iob                                drwxrwx--x   2021-09-08 18:
> com.gaana                                  drwxrwx--x   2021-09-08 18:
> com.google.android.apps.docs               drwxrwx--x   2021-09-08 18:
```

How to visualize the Database Using SQLite Browser

Go to website : https://sqlitebrowser.org/



Downloads

(Please consider sponsoring us on Patreon 😄)

Windows

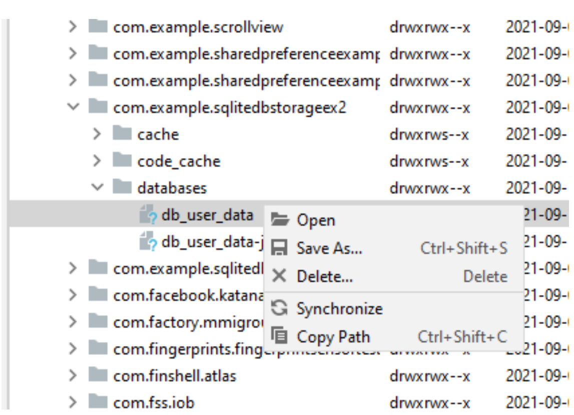
Our latest release (3.12.2) for Windows:

- DB Browser for SQLite – Standard installer for 32-bit Windows
- DB Browser for SQLite – .zip (no installer) for 32-bit Windows
- DB Browser for SQLite – Standard installer for 64-bit Windows
- DB Browser for SQLite – .zip (no installer) for 64-bit Windows

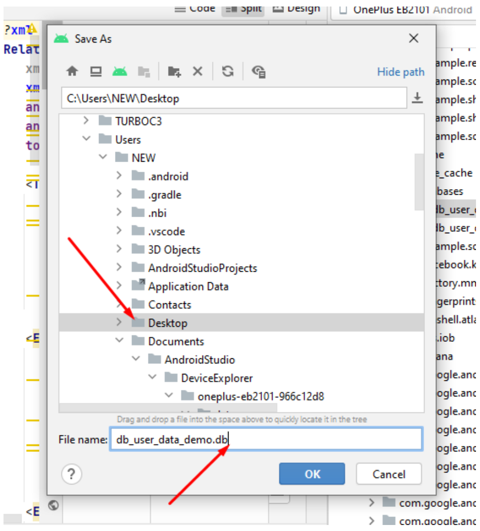
Windows PortableApp

- DB Browser for SQLite – PortableApp

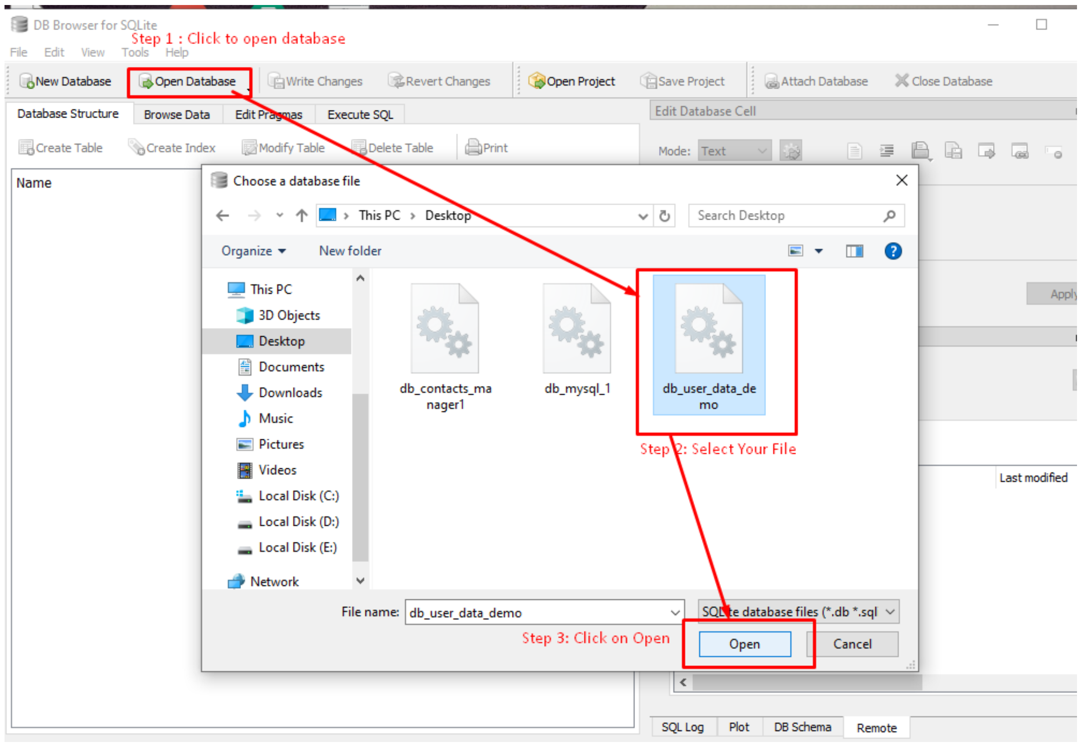
Now Right Click on Database – Save on Particular Location

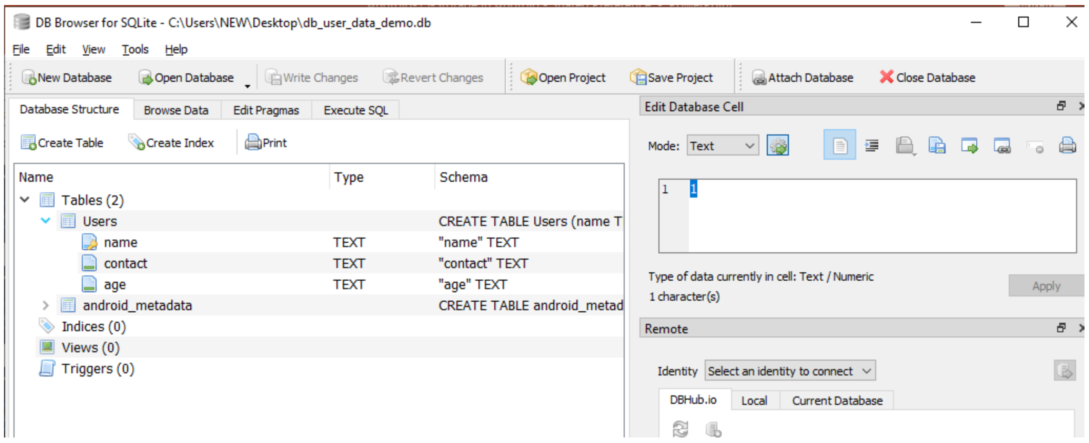


```
> com.example.scrollview                  drwxrwx--x  2021-09-
> com.example.sharedpreferenceexamp       drwxrwx--x  2021-09-
> com.example.sharedpreferenceexamp       drwxrwx--x  2021-09-
v com.example.sqlitedbstorageex2          drwxrwx--x  2021-09-
  > cache                                 drwxrws--x  2021-09-
  > code_cache                            drwxrws--x  2021-09-
  v databases                             drwxrwx--x  2021-09-
      db_user_data                                    21-09-
      db_user_data-j                                  21-09-
> com.example.sqlitedl                                21-09-
> com.facebook.katana                                 21-09-
> com.factory.mmigro                                  21-09-
> com.fingerprints.fing                               21-09-
> com.finshell.atlas                      drwxrwx--x  2021-09-
> com.fss.iob                             drwxrwx--x  2021-09-
```

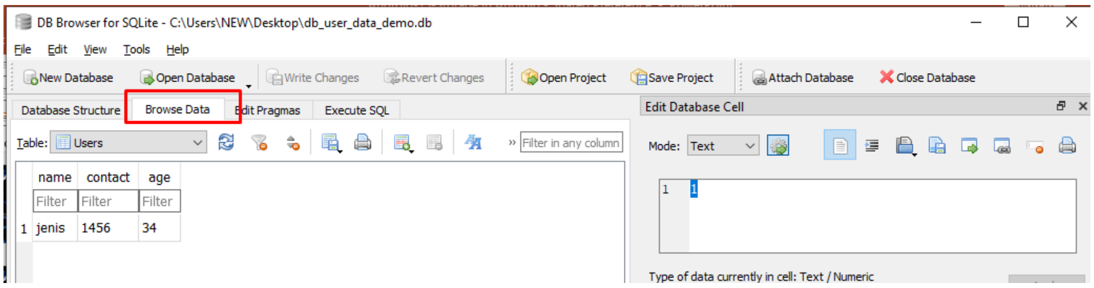


Click to open Database & Check in SQLite Browser





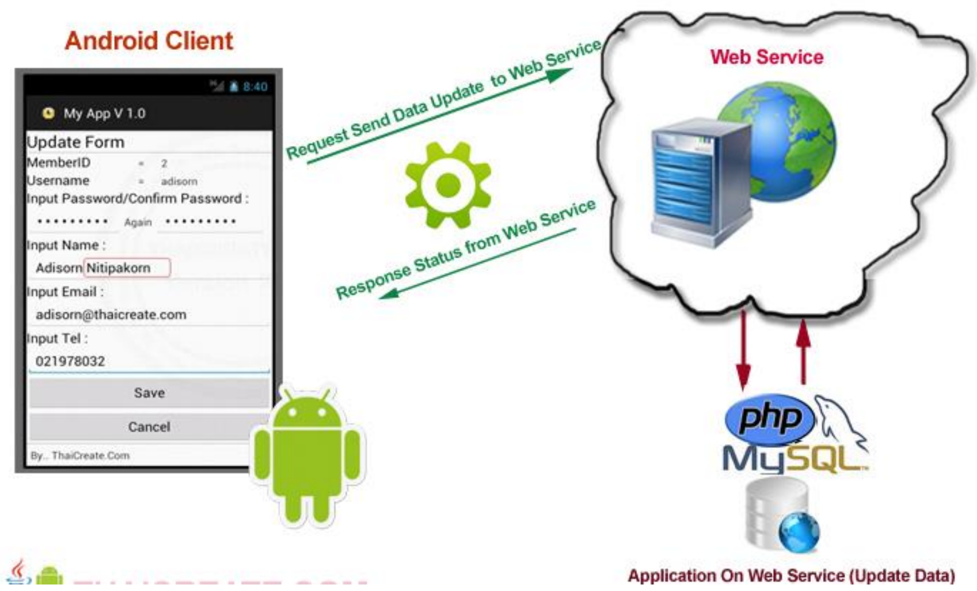
Name	Type	Schema
Tables (2)		
Users		CREATE TABLE Users (name T
name	TEXT	"name" TEXT
contact	TEXT	"contact" TEXT
age	TEXT	"age" TEXT
android_metadata		CREATE TABLE android_metad
Indices (0)		
Views (0)		
Triggers (0)		


DB Browser for SQLite - C:\Users\NEW\Desktop\db_user_data_demo.db
Browse Data
Table: Users

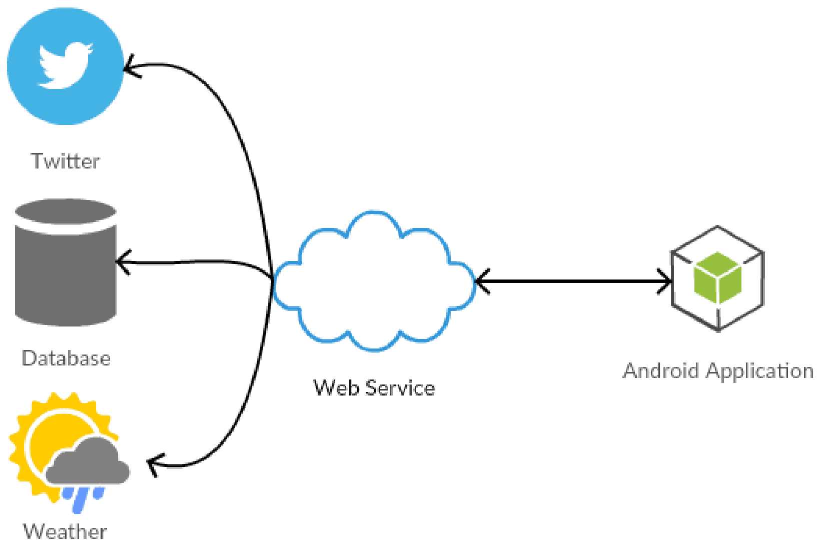
	name	contact	age
1	jenis	1456	34

Web Services In Android

- A web service is a standard for exchanging information between different types of applications irrespective of language and platform.
- Web services are a collection of open-source protocols and standards(xml, http, etc.) that are useful for the exchange of data between systems or applications.

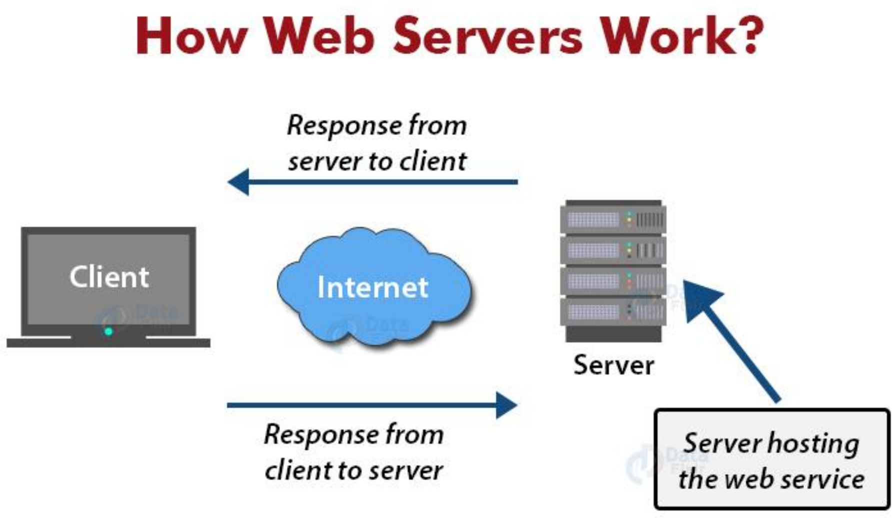


Example of Process of Webservice



Example of Process of Webservice

- A web service is basically required to provide interoperability, i.e. connecting various applications.
- It allows different apps to communicate with each other and share the data and services among themselves.
- Web services provide a standard for all the types of client applications to invoke functions on every type of app server.
- For example, you can consider an android application interacting with a .NET app using a web service.
- Provided below is the simple depiction of how a Web Server actually works. It has two important things that are the Client and the Server.
- Here first the Client makes a request from the Server and then, the Server makes a response to the Client


How Web Servers Work?
Response from server to client
Client
Internet
Server
Response from client to server
Server hosting the web service

Process of Web Service

How to understand the Web Service

Situation

- Two Person are there. One understands marathi and another gujarati. Now Problem : How can they understand each other?

Solution :

- They need common Language. Like English.
- Need language with specific rules that matches their accent.
- With Language they need Medium to communicate with each other. Like air or telephone

NOW Considering Above Scenario – & Matching with Web Services.

Problem:

- Mobile App – language is Android -- Wants to Access Database.
- Can not be possible TO Access Data From Database.

Solution

- So, We need Web Technology languages Like PHP- JAVA - .net to access the data from Database.
- But The Languages formats are different. So We need common format to communicate between APP & SERVER. That format can be XML or JSON.
- The protocol or rules can be HTTP – HTTPS – FTP – SMTP
- The medium can be LAN AND WAN

That means: Language is JSON – Medium is LAN or Server/Network

PROTOCOL is – HTTP – HTTPS – FTP – SMTP

Web Server Architecture

1. Publisher
   The publisher can be understood as a Service provider. The publisher is responsible for creating the web service and making it available for the Clients.

2. Subscriber
   The Subscriber is nothing but, the service requester. The Service requester is the one that needs to contact the web service. The client application will contact through a client application. This Client application can be based on .Net or any language based language.

3. Broker
   The broker here is the application that provides access to the UDDI. The UDDI stands for User descriptive, discovery and integration. It enables the client application to locate the web service exactly.

Services are as Follows

1. Publish
   Publishers Publishing the web services means informing the broker about its existence. It is done using the Broker's interface to make is easily accessible to the subscribers

2. Subscribe
   The Subscriber will consult the broker to locate the published web service easily .

3. Bind
   Once the information regarding the web services is gained from the broker, the subscriber can bind the web service.

Characteristics of Web Services

1. Web services are XML – based. They use it at its data representational layer and its transportational layer as it removes networking, operating system or even the platform binding. These services are highly interoperable at their core level.

2. Web services are loosely coupled. That means the consumer web services and providers of web service are not tied together directly.

3. Web services have the ability to be either Synchronous or Asynchronous. Here Synchronous can be understood as binding the client to the execution of the service. On the other hand, Asynchronous refers to allowing the client to invoke a service first and later executing the other functions.

4. Web Services supports Remote Procedure Calls. Remote Procedure calls can often be referred to as RPCs. These RPCs let the clients invoke various functions, methods, and services on remote objects using XML.

5. There is support to Document exchange in Web Services. In fact, XML has a very generic way to represent data as well as complex documents. Along with that, it has got various ways to represent these documents.

Types of Web Services

1. XML-RPC

In XML-RPC, RPC stands for remote procedure calls. It is an XML based protocol for the exchange of data between a huge range of devices over the internet.

2. UDDI

UDDI stands for Universal Descriptive, discovery, and integration. It is an XML- based standard used for detailing, publishing and discovering new web services.

3. SOAP

SOAP here stands for Simple object access protocol. It is an XML based web service protocol used for the exchange of data or documents over HTTP(Hypertext transfer protocol) or SMTP(Simple Message Transfer Protocol). It allows the communication of independent processes that operate on disparate systems.

4. REST

Here, REST is Representational State Transfer. It provides communication and connectivity between devices and the internet.

Advantages of Web Services

Following are the advantages of Web services-

1. Web services enable interoperability among different Applications.
2. One of the very important advantages of using web services is Reusability.
3. Web services offer faster communications within and across applications and organizations.
4. They use a quality industry-standard protocol to enable communication between different applications.
5. They use SOAP over HTTP to enable the use of low-cost internet for implementing web services.
6. Web Services are deployed over the standard internet technologies.
7. They allow us to expose the functions of the existing code over the internet

Dis Advantages of Web Services

1. Web services do not access from the browser.
2. They don’t leverage emerging Web developments
3. The HTTP protocol used by web services is not reliable and is insecure.

JSON Parsing

- JSON stands for JavaScript Object Notation.
- This an independent data exchange format and is the best alternative for XML.
- Android provides four different classes to manipulate JSON data. These classes are JSONArray,JSONObject,JSONStringer and JSONTokenizer.
- The main advantage of JSON is, it’s a language independent and the JSON object will contain data like key/value pair.
- Generally, the JSON nodes will start with a square bracket ([) or with a curly bracket ({).
- The difference between square bracket and curly bracket is, the square bracket ([) represents the starting of a JSONArray node, whereas curly bracket ({) represents a JSONObject so we need to call appropriate method to get the data.
- If JSON data starts with [, then we need to use getJSONArray() method to get the data, same way if it starts with {, then we need to use getJSONObject() method.
- To parse the JSON data in android, we need to create an instance of JSONObject and JSONArray objects with a string that contains JSON data in it.

JSON Object:

```
{
   "employee": {
      "name":      "sachin",
      "salary":    56000,
      "married":   true
   }
}
```

JSON Array Ex:

```
{ "Employee" :
   [
    {"id":"101","name":"Sonoo Jaiswal","salary":"50000"},
    {"id":"102","name":"Vimal Jaiswal","salary":"60000"}
   ]
}
```

Example of Both Array & Object:

Array of users

```
{
      "users": [
                  {
                  "name": "JENIS SHAH",
                  "designation": "Asst. Professor",
                  "location": "LJIET"
                  },
                  {
                  "name": "MAHESH SHAH",
                  "designation": "LAB Assistant",
                  "location": "LJMBA"
                  }
                  {
                  "name": "Suresh SHAH",
                  "designation": "LAB Assistant",
                  "location": "LJMCA"
                  }
                  {
                  "name": "Ramesh SHAH",
                  "designation": "Asst. Professor",
                  "location": "LJMCA"
                  }

                  {
                  "name": "Naresh Shah",
                  "designation": "Asst. Professor",
                  "location": "LJIET"
                  }


          ]
 }
```

JSON ELEMNTS

1. Array([)
   In a JSON file , square bracket ([) represents a JSON array

2. Objects({)
   In a JSON file, curly bracket ({) represents a JSON object

3. Key
   A JSON object contains a key that is just a string. Pairs of key/value make up a JSON object

4. Value
   Each key has a value that could be string , integer or double e.t.

How to parse JSON Data

String in;

JSONObject reader = new JSONObject(in);

JSONObject sys = reader.getJSONObject("sys");

country = sys.getString("country");

JSONObject main = reader.getJSONObject("main");

temperature = main.getString("temp");

Below is Another Code Snippet to explain JSON Data in android using JSON Object and Array

```
JSONObject jObj = new JSONObject(jsonStr);
JSONArray jsonArry = jObj.getJSONArray("users");
for(int i=0;i<jsonArry.length();i++){
    HashMap<String,String> user = new HashMap<>();
    JSONObject obj = jsonArry.getJSONObject(i);
    user.put("name",obj.getString("name"));
    user.put("designation",obj.getString("designation"));
    user.put("location",obj.getString("location"));
    userList.add(user);
}
```

Prog:1

SIMPLE JSON PARSING EXAMPLE – 1 USING GSON

(Convert JAVA object to JSON & JSON to JAVA Object)

Dependency

Add Dependency in build.gradle (Module File in dependencies { ... }):

```
implementation 'com.google.code.gson:gson:2.8.6'
```

Permissions: No Extra permission are required

Only We are focusing on Java Class Files.

Steps : Write code in Main Activity.Java File for Creating Employee Class with basic Data. & Constructor.

Employee.java

```
package com.example.jsonexamples;

import com.google.gson.annotations.SerializedName;

public class Employee {

    private String firstname;
    private int number;
    private int age;


    public Employee(String firstname, int number, int age)

      {
              this.firstname = firstname;
              this.number= number;
              this.age = age;

      }
}
```

MainActivity.java

```
Gson gson = new Gson();;
Employee emp = new Employee("JENIS",23,28);

String json = gson.toJson(emp); // this will convert java object to json
```

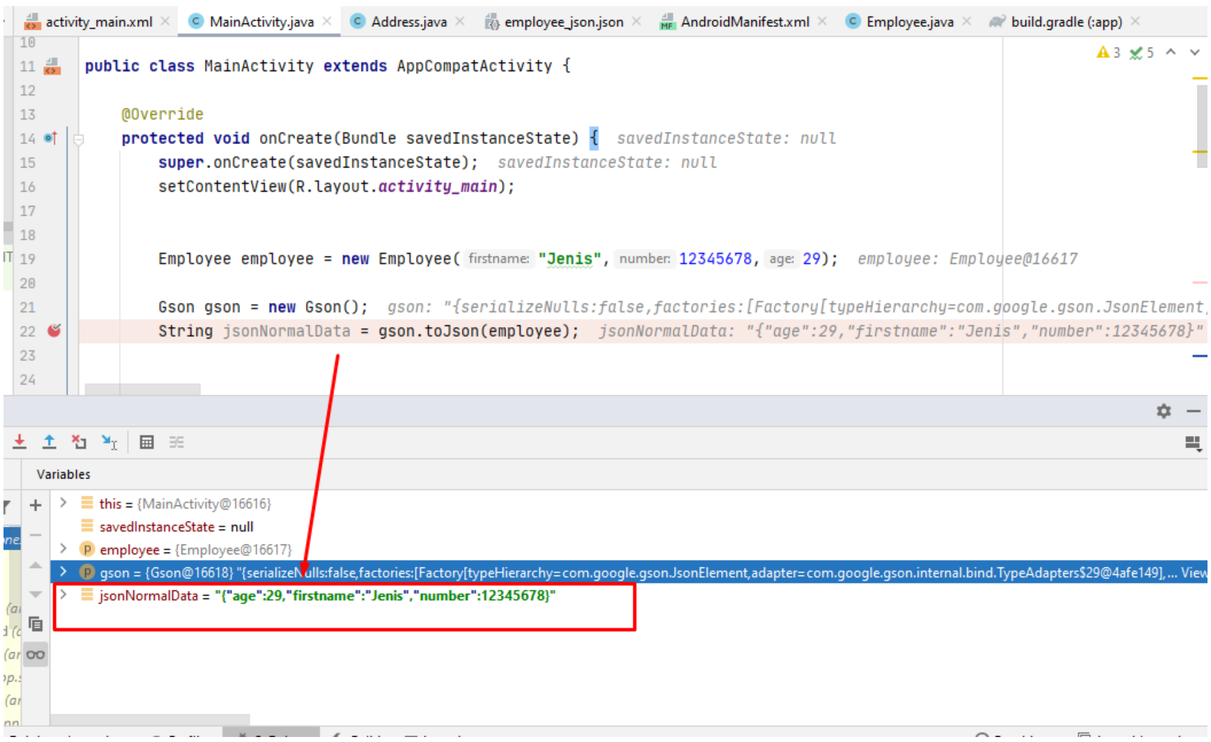
Now Debut the code

<u>HOW TO DEBUG THE CODE & OPEN DEBUG WINDOW</u>

Open Debug Window : Shift+F9
Check Output in Bottom of the Screen

To Run The Debug the app from the Run menu → Debug App

You can see the output in the bottom DEBUG WINDOW



```
public class MainActivity extends AppCompatActivity {

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


        Employee employee = new Employee( firstname: "Jenis", number: 12345678, age: 29);  employee: Employee@16617

        Gson gson = new Gson();  gson: "{serializeNulls:false,factories:[Factory[typeHierarchy=com.google.gson.JsonElement,
        String jsonNormalData = gson.toJson(employee);  jsonNormalData: "{"age":29,"firstname":"Jenis","number":12345678}"
```

```
this = {MainActivity@16616}
savedInstanceState = null
employee = {Employee@16617}
gson = {Gson@16618} "{serializeNulls:false,factories:[Factory[typeHierarchy=com.google.gson.JsonElement,adapter=com.google.gson.internal.bind.TypeAdapters$29@4afe149],... View
jsonNormalData = "{"age":29,"firstname":"Jenis","number":12345678}"
```

Output in Debug Window : Press F8 to check more debug data
jsonNormalData = {"age":29,"firstname":"Jenis","number":12345678}

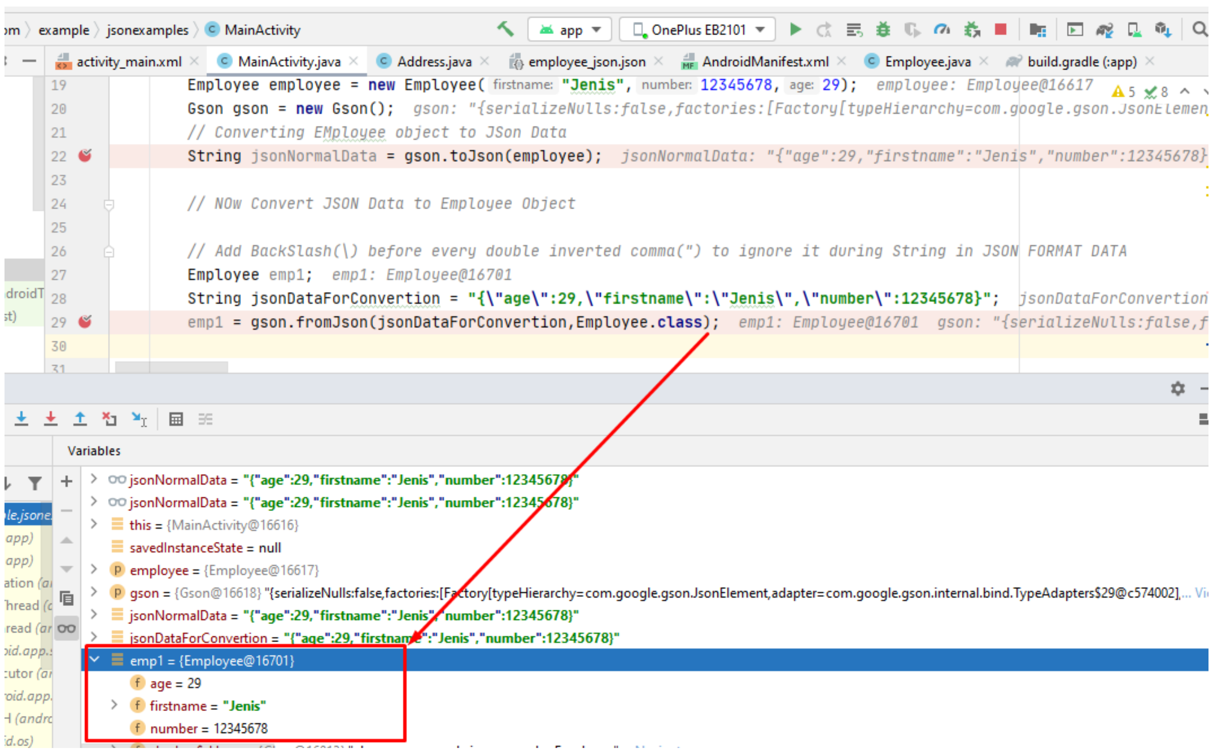
Now Converting JSON Data to Java Object

```
// Add BackSlash(\) before every double inverted comma(") to ignore it
during String in JSON FORMAT DATA

Employee emp1;
String jsonDataForConvertion =
"{\"age\":29,\"firstname\":\"Jenis\",\"number\":12345678}";
emp1 = gson.fromJson(jsonDataForConvertion,Employee.class);
```

To Run The code: Again Debug the app from the Run menu → Debug App

You can see the output in the bottom DEBUG WINDOW

Output: (Do not forget to click on **F8 Button to** check more debug output or click on **Step Over** Button in Debug Window.



```
Employee employee = new Employee( firstname: "Jenis", number: 12345678, age: 29);   employee: Employee@16617
Gson gson = new Gson();   gson: "{serializeNulls:false,factories:[Factory[typeHierarchy=com.google.gson.JsonElemen
// Converting EMployee object to JSon Data
String jsonNormalData = gson.toJson(employee);   jsonNormalData: "{"age":29,"firstname":"Jenis","number":12345678}

// NOw Convert JSON Data to Employee Object

// Add BackSlash(\) before every double inverted comma(") to ignore it during String in JSON FORMAT DATA
Employee emp1;   emp1: Employee@16701
String jsonDataForConvertion = "{\"age\":29,\"firstname\":\"Jenis\",\"number\":12345678}";   jsonDataForConvertion
emp1 = gson.fromJson(jsonDataForConvertion,Employee.class);   emp1: Employee@16701   gson: "{serializeNulls:false,f
```

```
jsonNormalData = "{"age":29,"firstname":"Jenis","number":12345678}"
jsonNormalData = "{"age":29,"firstname":"Jenis","number":12345678}"
this = {MainActivity@16616}
savedInstanceState = null
employee = {Employee@16617}
gson = {Gson@16618} "{serializeNulls:false,factories:[Factory[typeHierarchy=com.google.gson.JsonElement,adapter=com.google.gson.internal.bind.TypeAdapters$29@c574002],...
jsonNormalData = "{"age":29,"firstname":"Jenis","number":12345678}"
jsonDataForConvertion = "{"age":29,"firstname":"Jenis","number":12345678}"
emp1 = {Employee@16701}
    age = 29
    firstname = "Jenis"
    number = 12345678
```

Now Nested JSON Object Examples

Add a Address class in the Above code.

```
package com.example.jsonexamples;

public class Address {

    private String city;
    private String state;

    public Address(String city, String state) {
        this.city = city;
        this.state = state;
    }
}
```

- Now we are going to add the Address object as the argument of Employee class.

Updated Employee Class

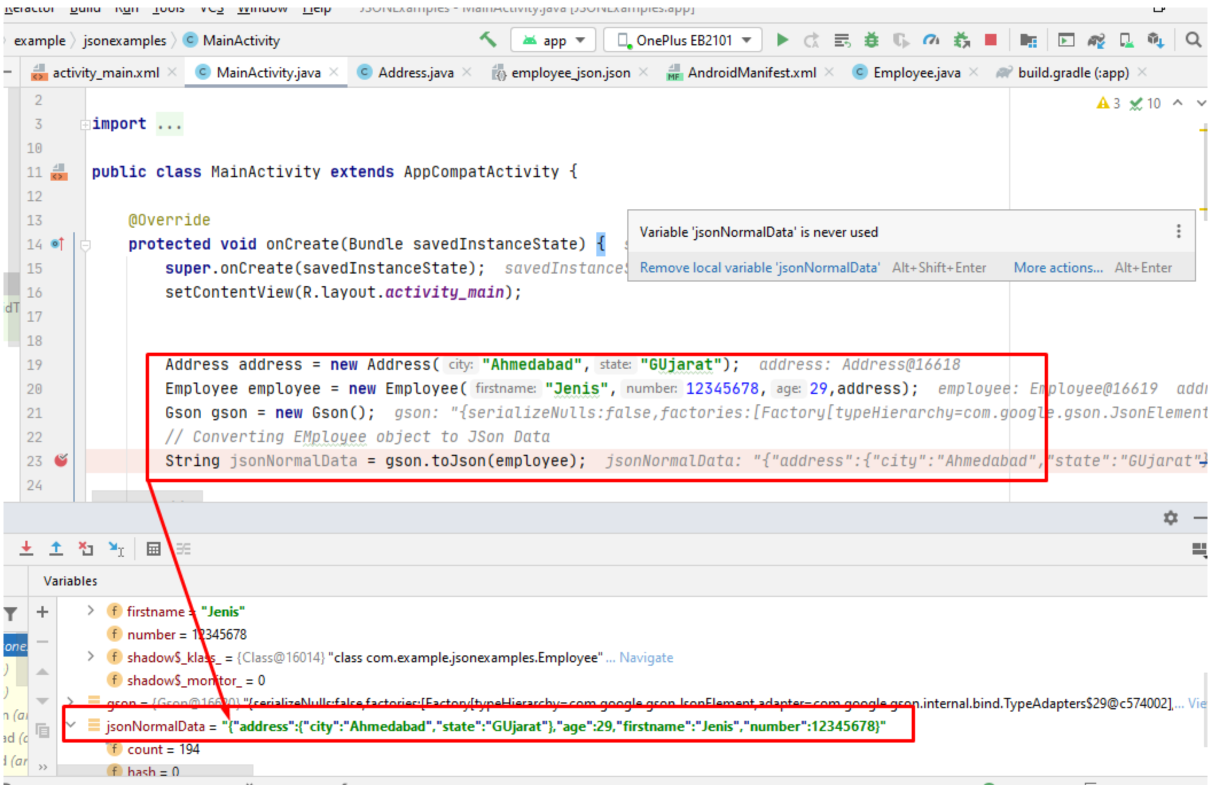
```
package com.example.jsonexamples;

import com.google.gson.annotations.SerializedName;

public class Employee {
    private String firstname;
    private int number;
    private int age;
    private Address address;

    public Employee(String firstname, int number, int age, Address address)
{
        this.firstname = firstname;
        this.number = number;
        this.age = age;
    }
}
```

Updated MainActivity.Java File

```
package com.example.jsonexamples;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.util.Log;
import android.widget.Toast;

import com.google.gson.Gson;

public class MainActivity extends AppCompatActivity {

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


        Address address = new Address("Ahmedabad","GUjarat");
        Employee employee = new Employee("Jenis",12345678,29,address);
        Gson gson = new Gson();
        // Converting EMployee object to JSon Data
        String jsonNormalData = gson.toJson(employee);
        }
}
```

Output :

JSONNormalData =
{"address":{"city":"Ahmedabad","state":"GUjarat"},"age":29,"firstname":"Jenis","number":12345678}

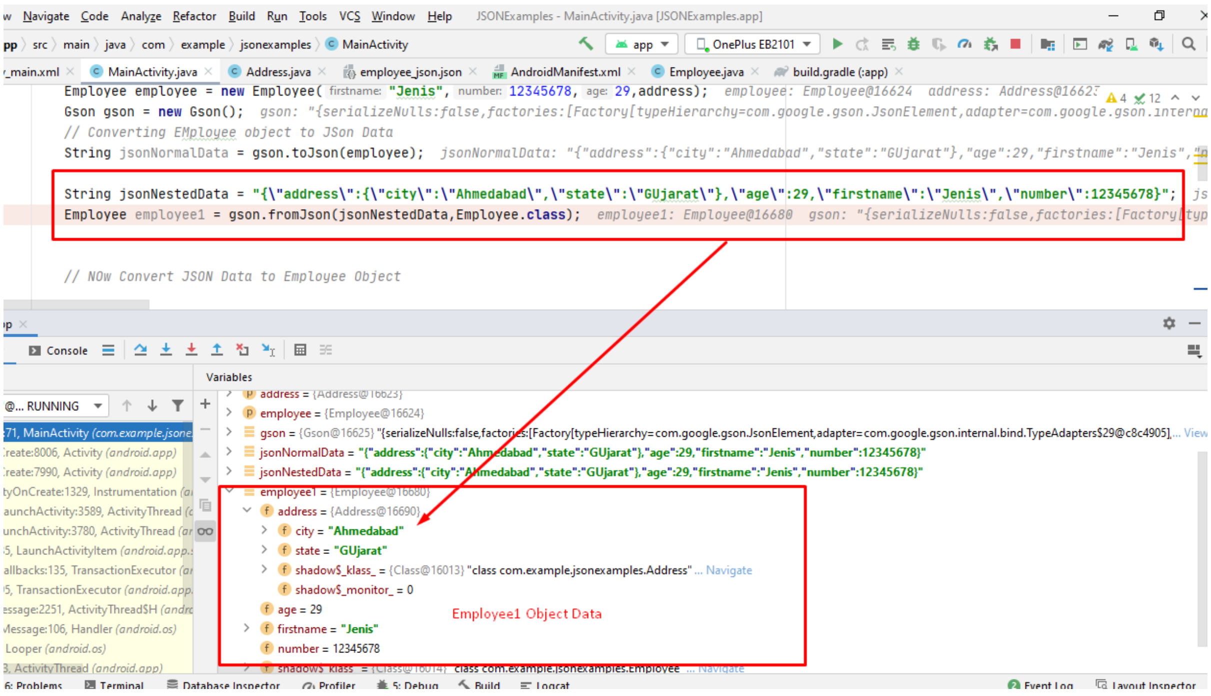


```
import ...

public class MainActivity extends AppCompatActivity {

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


        Address address = new Address( city: "Ahmedabad", state: "GUjarat");   address: Address@16618
        Employee employee = new Employee( firstname: "Jenis", number: 12345678, age: 29,address);   employee: Employee@16619   addr
        Gson gson = new Gson();   gson: "{serializeNulls:false,factories:[Factory[typeHierarchy=com.google.gson.JsonElement
        // Converting EMployee object to JSon Data
        String jsonNormalData = gson.toJson(employee);   jsonNormalData: "{"address":{"city":"Ahmedabad","state":"GUjarat"}
```

```
Variable 'jsonNormalData' is never used
Remove local variable 'jsonNormalData'  Alt+Shift+Enter   More actions...  Alt+Enter
```

```
Variables
  firstname = "Jenis"
  number = 12345678
  shadow$_klass_ = {Class@16014} "class com.example.jsonexamples.Employee" ... Navigate
  shadow$_monitor_ = 0
gson = {Gson@16620} "{serializeNulls:false,factories:[Factory[typeHierarchy=com.google.gson.JsonElement,adapter=com.google.gson.internal.bind.TypeAdapters$29@c574002],... View
jsonNormalData = "{"address":{"city":"Ahmedabad","state":"GUjarat"},"age":29,"firstname":"Jenis","number":12345678}"
  count = 194
  hash = 0
```

Now convert back to the Employee Object: (Code & Output)



```
Employee employee = new Employee( firstname: "Jenis", number: 12345678, age: 29,address);   employee: Employee@16624   address: Address@16623
Gson gson = new Gson();   gson: "{serializeNulls:false,factories:[Factory[typeHierarchy=com.google.gson.JsonElement,adapter=com.google.gson.interna
// Converting EMployee object to JSon Data
String jsonNormalData = gson.toJson(employee);   jsonNormalData: "{"address":{"city":"Ahmedabad","state":"GUjarat"},"age":29,"firstname":"Jenis","n

String jsonNestedData = "{\"address\":{\"city\":\"Ahmedabad\",\"state\":\"GUjarat\"},\"age\":29,\"firstname\":\"Jenis\",\"number\":12345678}";
Employee employee1 = gson.fromJson(jsonNestedData,Employee.class);   employee1: Employee@16680   gson: "{serializeNulls:false,factories:[Factory[typ


// NOw Convert JSON Data to Employee Object
```

```
Variables
address = {Address@16623}
employee = {Employee@16624}
gson = {Gson@16625} "{serializeNulls:false,factories:[Factory[typeHierarchy=com.google.gson.JsonElement,adapter=com.google.gson.internal.bind.TypeAdapters$29@c8c4905],... View
jsonNormalData = "{"address":{"city":"Ahmedabad","state":"GUjarat"},"age":29,"firstname":"Jenis","number":12345678}"
jsonNestedData = "{"address":{"city":"Ahmedabad","state":"GUjarat"},"age":29,"firstname":"Jenis","number":12345678}"
employee1 = {Employee@16680}
  address = {Address@16690}
    city = "Ahmedabad"
    state = "GUjarat"
    shadow$_klass_ = {Class@16013} "class com.example.jsonexamples.Address" ... Navigate
    shadow$_monitor_ = 0
  age = 29
  firstname = "Jenis"
  number = 12345678
  shadow$_klass_ = {Class@16014} "class com.example.jsonexamples.Employee" ... Navigate
```

Employee1 Object Data

Prog:2

SIMPLE JSON PARSING EXAMPLE – 2

(Parsing Static JSON Data & bind to List View)

AIM: Our aim is to parse the json string to the listview in the android to showcase our data.

Remember: Here we are using inbuild classes like JSONArray and JSONObject to get the data. We are using static JSON data in the project.

Steps:

- Create a activity_main.xml file with List View
- Create a List View Layout File (list_row.xml) with 3 text views to show data.
- Bind the JSON data in to main Activity file & bind the data with List view.

Activity_Main.xml File

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <ListView
        android:id="@+id/user_list"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:dividerHeight="1dp" />

</LinearLayout>
```

List_row.xml

```
<?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:orientation="horizontal"
    android:padding="4dp">
    <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>
```

MainActivity.java

```
package com.example.jsonparserexample;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.util.Log;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.widget.SimpleAdapter;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.HashMap;

public class MainActivity extends AppCompatActivity {


    ListView lv;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        String jsondata = getListData();

        ArrayList<HashMap<String, String>> userlist = new ArrayList();
        lv = findViewById(R.id.user_list);

        try {
            // create json object
            JSONObject jsonObject = new JSONObject(jsondata);

            // get users array from the json object
            JSONArray jsonArray = jsonObject.getJSONArray("users");

            for(int i=0; i<jsonArray.length();i++)
            {
                HashMap<String,String> user = new HashMap<>();
                JSONObject obj  = jsonArray.getJSONObject(i);
                user.put("name",obj.getString("name"));
                user.put("designation",obj.getString("designation"));
                user.put("location",obj.getString("location"));
                userlist.add(user);
            }

            ListAdapter listAdapter = new
SimpleAdapter(MainActivity.this,userlist,R.layout.list_row,new
String[]{"name","designation","location"},new
int[]{R.id.name,R.id.designation,R.id.location});
            lv.setAdapter(listAdapter);

        } catch (JSONException e) {
            Log.e("JsonParser Example","unexpected JSON exception", e);
        }


        // create json array
    }

    private String getListData() {
        String jsonStr = "{ \"users\" :[" +
                "{\"name\":\"Jenis Shah\",\"designation\":\"Asst.
Prof\",\"location\":\"LJIET\"}" +
                ",{\"name\":\"Ramesh Shah\",\"designation\":\"Placement
Officer\",\"location\":\"LJMCA\"}" +
                ",{\"name\":\"Mahesh Shah\",\"designation\":\"
Accountant\",\"location\":\"LJCOM\"}] }";
        return jsonStr;
    }
}
```

Prog:3

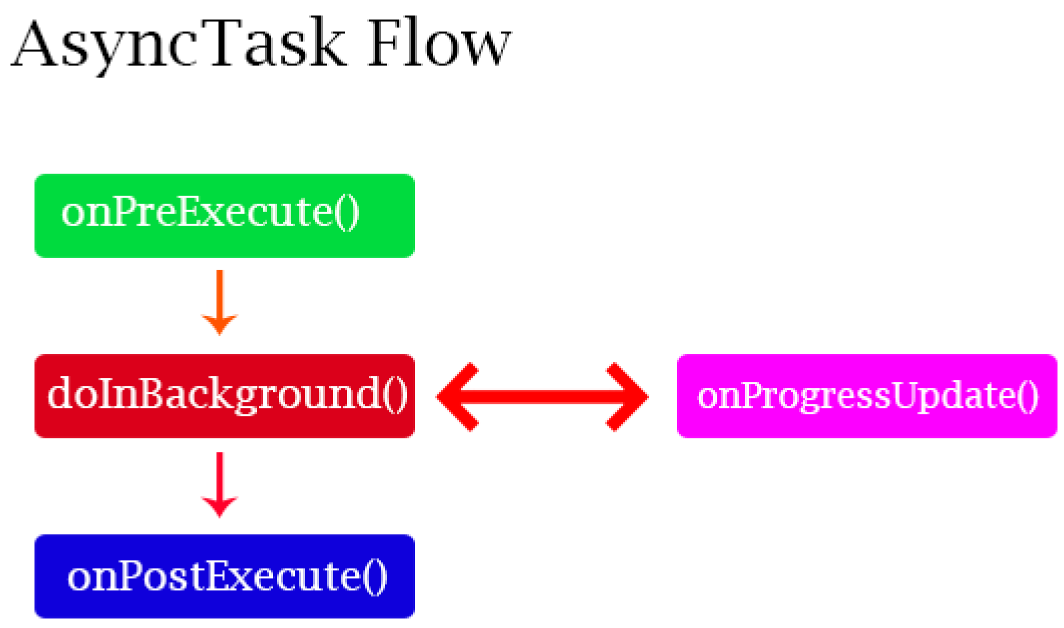
JSON PARSING WEB DATA FROM WEB API

ASYNC Task

In Android, AsyncTask (Asynchronous Task) allows us to run the instruction in the background and then synchronize again with our main thread. This class will override at least one method i.e doInBackground(Params) and most often will override second method onPostExecute(Result).

AsyncTask class is used to do background operations that will update the UI(user interface). Mainly we used it for short operations that will not effect on our main thread.

AsyncTask class is firstly executed using execute() method. In the first step AsyncTask is called onPreExecute() then onPreExecute() calls doInBackground() for background processes and then doInBackground() calls onPostExecute() method to update the UI.



Syntax: AsyncTask <TypeOfVarArgParams, ProgressValue, ResultValue>

1. TypeOfVarArgParams: Params is the type of the parameters sent to the task upon execution.

2. ProgressValue: Progress is the type of the progress units published during the background computation.

3. ResultValue: ResultValue is the type of the result of the background computation.

Method of AsyncTask In Android:

1. **onPreExecute()** – It invoked on the main UI thread before the task is executed. This method is mainly used to setup the task for instance by showing a ProgressBar or ProgressDialog in the UI(user interface).

2. **doInBackground(Params)** – This method is invoked on the background thread immediately after onPreExecute() finishes its execution. Main purpose of this method is to perform the background operations that can take a long time.

   The parameters of the Asynchronous task are passed to this step for execution. The result of the operations must be returned by this step and it will be passed back to the last step/method i.e onPostExecutes().

   This method can also use publishProgress(Progress…) to publish one or more units of progress. These values will be published on the main UI thread in the onProgressUpdate(Progress…) method.

3. **onProgressUpdate(Progress…)** – This method is invoked on the main UI thread after a call to publishProgress(Progress…). Timing of the execution is undefined.

   This method is used to display any form of progress in the user interface while the background operations are executing. We can also update our progress status for good user experience.

4. **onPostExecute(Result)** – This method is invoked on the main UI thread after the background operation finishes in the doInBackground method.

   The result of the background operation is passed to this step as a parameter and then we can easily update our UI to show the results.

(Here we are taking data from the Web API: Where API URL =

Main Activity.java

```java
package com.example.tutorialspoint7.myapplication;

import android.os.AsyncTask;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import android.widget.Toast;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.HashMap;

public class MainActivity extends AppCompatActivity {

   private String TAG = MainActivity.class.getSimpleName();
   private ListView lv;

   ArrayList<HashMap<String, String>> contactList;

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

      contactList = new ArrayList<>();
      lv = (ListView) findViewById(R.id.list);

      new GetContacts().execute();
   }

   private class GetContacts extends AsyncTask<Void, Void, Void>
{
      @Override
      protected void onPreExecute() {
         super.onPreExecute();
         Toast.makeText(MainActivity.this,"Json Data is
            downloading",Toast.LENGTH_LONG).show();

      }

      @Override
      protected Void doInBackground(Void... arg0) {
         HttpHandler sh = new HttpHandler();
         // Making a request to url and getting response
         String url = "http://api.androidhive.info/contacts/";
         String jsonStr = sh.makeServiceCall(url);

         Log.e(TAG, "Response from url: " + jsonStr);
         if (jsonStr != null) {
            try {
               JSONObject jsonObj = new JSONObject(jsonStr);

               // Getting JSON Array node
               JSONArray contacts =
jsonObj.getJSONArray("contacts");

               // looping through All Contacts
               for (int i = 0; i < contacts.length(); i++) {
                  JSONObject c = contacts.getJSONObject(i);
                  String id = c.getString("id");
                  String name = c.getString("name");
                  String email = c.getString("email");
                  String address = c.getString("address");
                  String gender = c.getString("gender");

                  // Phone node is JSON Object
                  JSONObject phone = c.getJSONObject("phone");
                  String mobile = phone.getString("mobile");
                  String home = phone.getString("home");
                  String office = phone.getString("office");

                  // tmp hash map for single contact
                  HashMap<String, String> contact = new
HashMap<>();

                  // adding each child node to HashMap key =>
value
                  contact.put("id", id);
                  contact.put("name", name);
                  contact.put("email", email);
                  contact.put("mobile", mobile);

                  // adding contact to contact list
                  contactList.add(contact);
               }
            } catch (final JSONException e) {
               Log.e(TAG, "Json parsing error: " +
e.getMessage());
               runOnUiThread(new Runnable() {
                  @Override
                  public void run() {
                     Toast.makeText(getApplicationContext(),
                     "Json parsing error: " + e.getMessage(),
                        Toast.LENGTH_LONG).show();
```

```
                    }
                 });

              }

           } else {
              Log.e(TAG, "Couldn't get json from server.");
              runOnUiThread(new Runnable() {
                 @Override
                 public void run() {
                    Toast.makeText(getApplicationContext(),
                       "Couldn't get json from server. Check LogCat
for possible errors!",
                       Toast.LENGTH_LONG).show();
                 }
              });
           }

           return null;
        }

        @Override
        protected void onPostExecute(Void result) {
           super.onPostExecute(result);
           ListAdapter adapter = new
SimpleAdapter(MainActivity.this, contactList,
              R.layout.list_item, new String[]{ "email","mobile"},
                 new int[]{R.id.email, R.id.mobile});
           lv.setAdapter(adapter);
        }
     }
}
```

HTTPHandler.java

```
package com.example.tutorialspoint7.myapplication;

import android.util.Log;

import java.io.BufferedInputStream;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.ProtocolException;
import java.net.URL;

public class HttpHandler {

   private static final String TAG =
HttpHandler.class.getSimpleName();

   public HttpHandler() {
   }

   public String makeServiceCall(String reqUrl) {
      String response = null;
      try {
         URL url = new URL(reqUrl);
         HttpURLConnection conn = (HttpURLConnection)
url.openConnection();
         conn.setRequestMethod("GET");
         // read the response
         InputStream in = new
BufferedInputStream(conn.getInputStream());
         response = convertStreamToString(in);
      } catch (MalformedURLException e) {
         Log.e(TAG, "MalformedURLException: " + e.getMessage());
      } catch (ProtocolException e) {
         Log.e(TAG, "ProtocolException: " + e.getMessage());
      } catch (IOException e) {
         Log.e(TAG, "IOException: " + e.getMessage());
      } catch (Exception e) {
         Log.e(TAG, "Exception: " + e.getMessage());
      }
      return response;
   }

   private String convertStreamToString(InputStream is) {
      BufferedReader reader = new BufferedReader(new
InputStreamReader(is));
      StringBuilder sb = new StringBuilder();

      String line;
      try {
         while ((line = reader.readLine()) != null) {
            sb.append(line).append('\n');
         }
      } catch (IOException e) {
         e.printStackTrace();
      } finally {
         try {
            is.close();
         } catch (IOException e) {
            e.printStackTrace();
         }
      }
      return sb.toString();
   }
}
```

Acitivty_main.xml

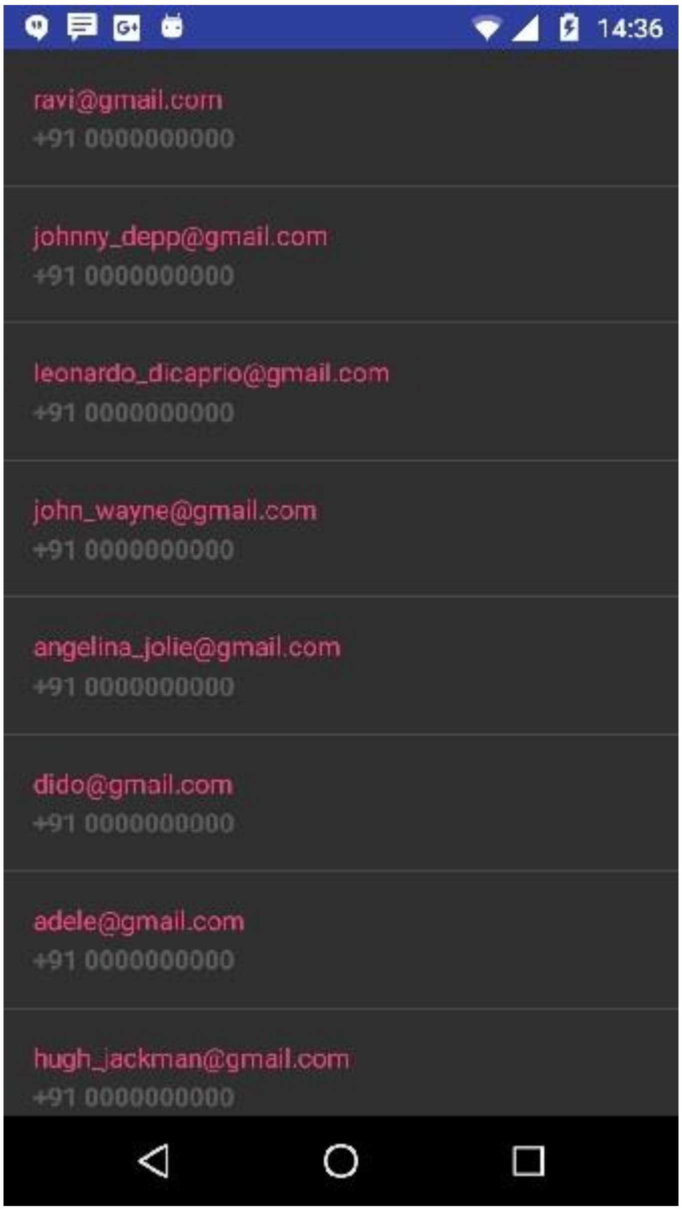
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

tools:context="com.example.tutorialspoint7.myapplication.MainActi
vity">

   <ListView
      android:id="@+id/list"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content" />
</RelativeLayout>
```

ListItem.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="wrap_content"
   android:orientation="vertical"
   android:padding="@dimen/activity_horizontal_margin">
   <TextView
      android:id="@+id/email"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:paddingBottom="2dip"
      android:textColor="@color/colorAccent" />

   <TextView
      android:id="@+id/mobile"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textColor="#5d5d5d"
      android:textStyle="bold" />
</LinearLayout>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.tutorialspoint7.myapplication">

   <uses-permission android:name="android.permission.INTERNET"/>
   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:supportsRtl="true"
      android:theme="@style/AppTheme">
         <activity android:name=".MainActivity">
            <intent-filter>
               <action android:name="android.intent.action.MAIN"
/>
               <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```



14:36

ravi@gmail.com
+91 0000000000

johnny_depp@gmail.com
+91 0000000000

leonardo_dicaprio@gmail.com
+91 0000000000

john_wayne@gmail.com
+91 0000000000

angelina_jolie@gmail.com
+91 0000000000

dido@gmail.com
+91 0000000000

adele@gmail.com
+91 0000000000

hugh_jackman@gmail.com
+91 0000000000

PROG 5:

Retrofit Library

- Retrofit is a type-safe REST client for Android, Java and Kotlin developed by Square.
- The library provides a powerful framework for authenticating and interacting with APIs and sending network requests with OkHttp.

Retrofit is used to perform the following tasks:

- It manages the process of receiving, sending, and creating HTTP requests and responses.
- It alternates IP addresses if there is a connection to a web service failure.
- It caches responses to avoid sending duplicate requests.
- Retrofit pools connections to reduce latency.
- Retrofit resolves issues before sending an error and crashing the app.

Advantages of retrofit

- It is very fast.
- It enables direct communication with the web service.
- It is easy to use and understand.
- It supports request cancellation.
- It supports post requests and multipart uploads.
- It supports both synchronous and asynchronous network requests.
- Supports dynamic URLs.
- Supports convertors.

Disadvantages of retrofit

- It does not support image loading. It requires other libraries such as Glide and Picasso.
- It does not support setting priorities.

Prerequisites to Setup Retrofit Library In Your PROJECT

1. Add permission of Internet in your manifest file.

```
<uses-permission
android:name="android.permission.INTERNET"/>
```

2. Add dependencies in you're app/build.gradle file

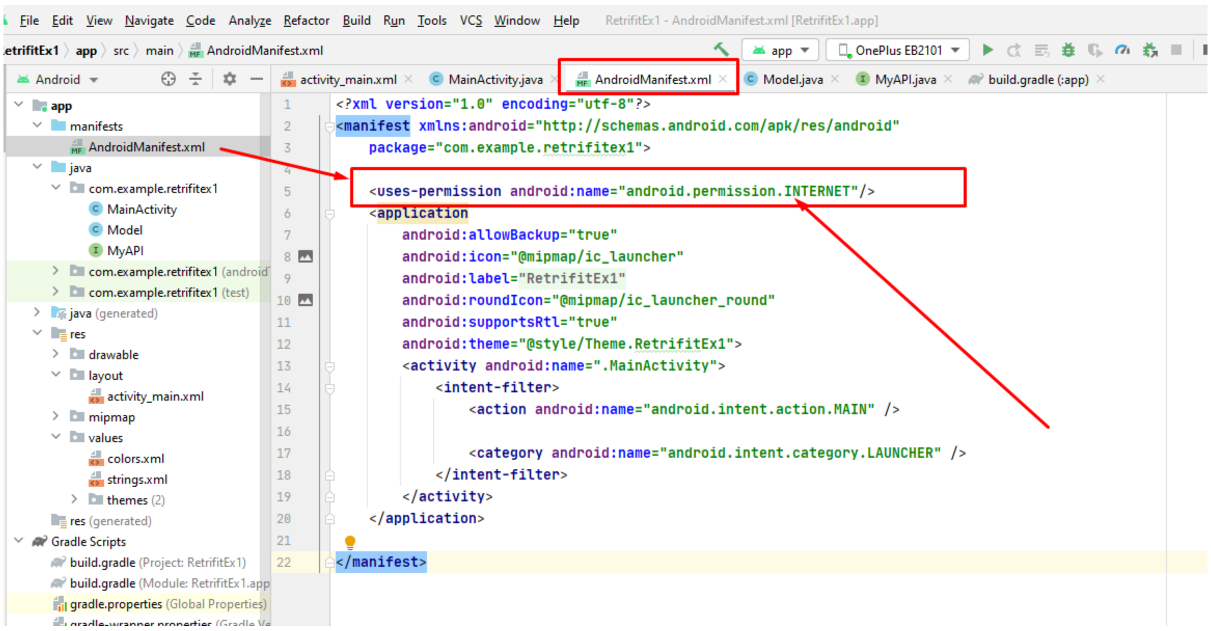
```
implementation 'com.squareup.retrofit2:retrofit:2.3.0'
implementation 'com.squareup.retrofit2:converter-
gson:2.3.0'
```

What are the Steps for Retrofit Data Fetching Project

1. Add Pre requisites in your project
2. Create a Simple POJO - Model Class as per your requirements of data.
3. Create an Interface (API Interface) With return Retrofit call of model class type
4. Create Retrofit Object in MainActivity.java File
5. Convert JSON Data to Model Class Object
6. Create Call of Model Class & Enqueue it For Processing
7. Receive Response Data in Simple Model Type List

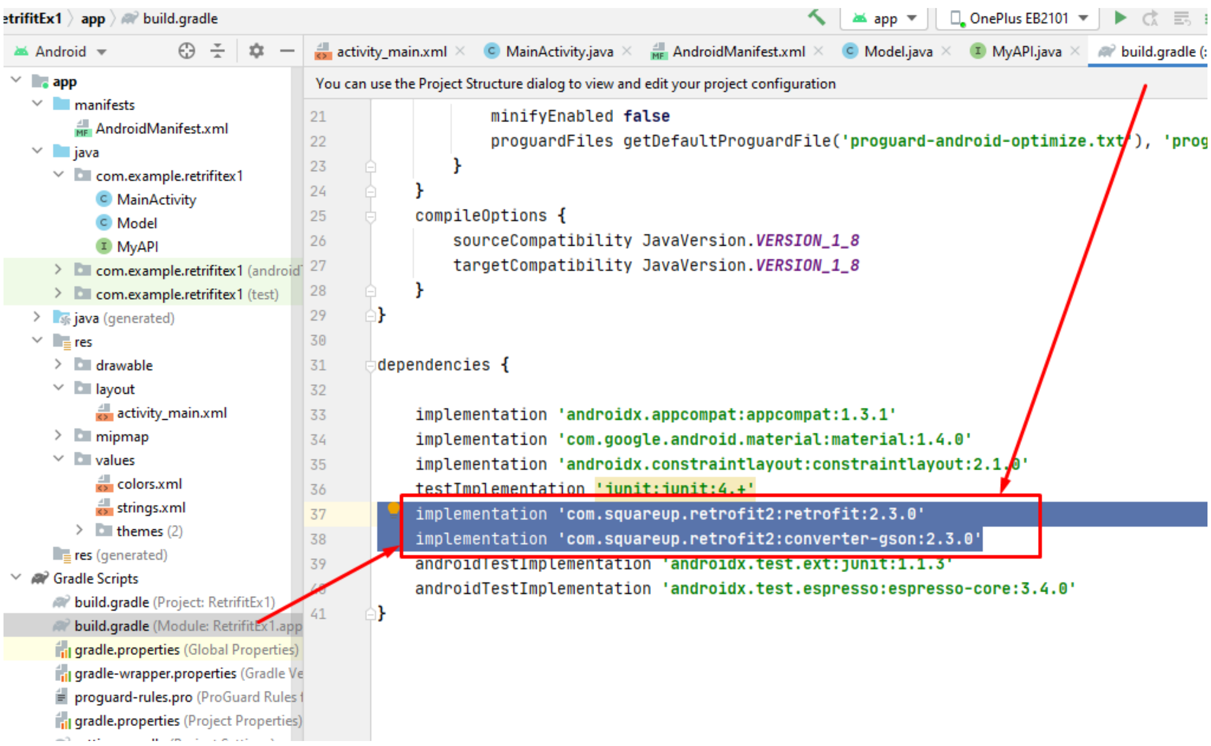
Now We are going to follow above mentioned steps & Create Project for it.

Firstly Create project & add the prerequisites as below



```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.retrifitex1">

    <uses-permission android:name="android.permission.INTERNET"/>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="RetrifitEx1"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.RetrifitEx1">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Then Add Dependencies as Below



You can use the Project Structure dialog to view and edit your project configuration

```
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'prog
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}

dependencies {

    implementation 'androidx.appcompat:appcompat:1.3.1'
    implementation 'com.google.android.material:material:1.4.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.0'
    testImplementation 'junit:junit:4.+'
    implementation 'com.squareup.retrofit2:retrofit:2.3.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.3.0'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
}
```

Now Create the Activity_main.xml File Where you want to Showcase your Fetched Data.

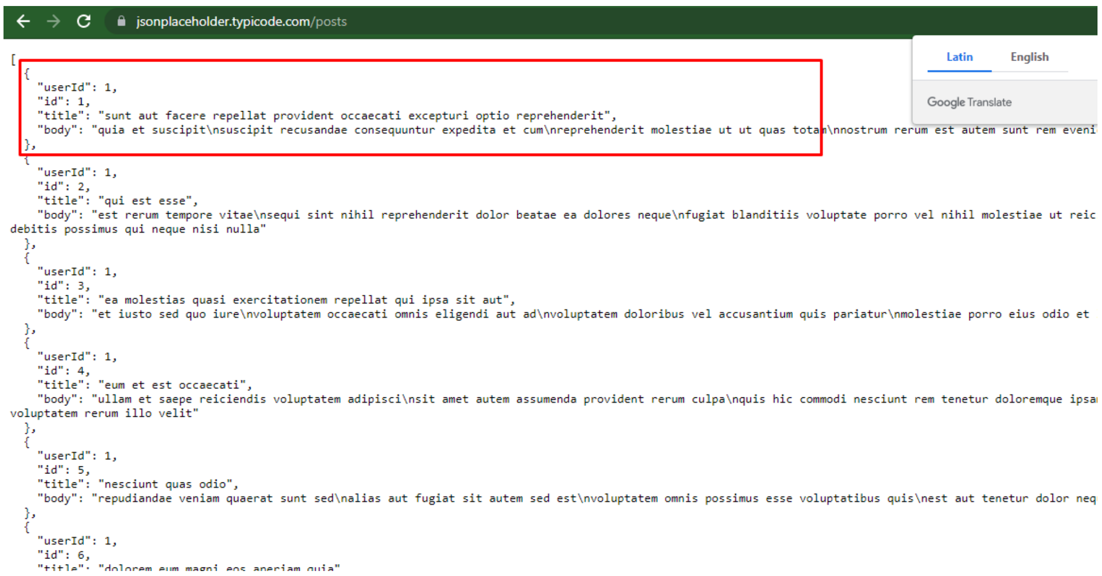
(Note: Here I took Scroll view to apply the scroll in the data & took one text view to show the data in the form of text view)

Activity_main.xml File

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        >


        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/dataTextView"
            android:text="DEMO"
            android:textSize="20dp"
            android:layout_marginTop="20dp"
            android:layout_marginStart="20dp"
            android:layout_marginBottom="30dp"
            />

    </ScrollView>

</LinearLayout>
```

Now Check your API from Where you want to fetch data.

Our API URL is : https://jsonplaceholder.typicode.com/posts



```
[
  {
    "userId": 1,
    "id": 1,
    "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
    "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveni
  },
  {
    "userId": 1,
    "id": 2,
    "title": "qui est esse",
    "body": "est rerum tempore vitae\nsequi sint nihil reprehenderit dolor beatae ea dolores neque\nfugiat blanditiis voluptate porro vel nihil molestiae ut reic
debitis possimus qui neque nisi nulla"
  },
  {
    "userId": 1,
    "id": 3,
    "title": "ea molestias quasi exercitationem repellat qui ipsa sit aut",
    "body": "et iusto sed quo iure\nvoluptatem occaecati omnis eligendi aut ad\nvoluptatem doloribus vel accusantium quis pariatur\nmolestiae porro eius odio et
  },
  {
    "userId": 1,
    "id": 4,
    "title": "eum et est occaecati",
    "body": "ullam et saepe reiciendis voluptatem adipisci\nsit amet autem assumenda provident rerum culpa\nquis hic commodi nesciunt rem tenetur doloremque ipsa
voluptatem rerum illo velit"
  },
  {
    "userId": 1,
    "id": 5,
    "title": "nesciunt quas odio",
    "body": "repudiandae veniam quaerat sunt sed\nalias aut fugiat sit autem sed est\nvoluptatem omnis possimus esse voluptatibus quis\nest aut tenetur dolor neq
  },
  {
    "userId": 1,
    "id": 6,
    "title": "dolorem eum magni eos aperiam quia"
```

Here we have Objects of fields like userId, id, title and body .

Now we are going to make the Model Class As per the fields of API.

Model.java

```
package com.example.retrifitex1;

public class Model {
    private int userId, id;
    private String title, body;

    public Model(int userId, int id, String title, String body) {
        this.userId = userId;
        this.id = id;
        this.title = title;
        this.body = body;
    }

    public int getUserId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getTitle() {
        return title;
    }

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

    public String getBody() {
        return body;
    }

    public void setBody(String body) {
        this.body = body;
    }
}
```

Now create an interface API for fetch data.

MyAPI.interface

```
package com.example.retrifitex1;


import java.util.List;

import retrofit2.Call;
import retrofit2.http.GET;

public interface MyAPI {



    // this is call for get model data. Here posts is the file name or we
can say the URL name

    @GET("posts")
    Call<List<Model>> getModelData();

}
```

MainActivity.Java file for Retrofit object creation & fetching data (Enqueue data)

```
package com.example.retrifitex1;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;
import android.widget.Toast;

import com.google.gson.Gson;

import java.util.List;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;
import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

public class MainActivity extends AppCompatActivity {


    TextView dataTv;
    String jsonURL = "https://jsonplaceholder.typicode.com/";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        dataTv = findViewById(R.id.dataTextView);


        //1.. create retrofit object


        Retrofit retrofit = new Retrofit.Builder()
                .baseUrl(jsonURL)
                .addConverterFactory(GsonConverterFactory.create())
                .build();


        // 2 .. Convert json data to model class object

        MyAPI myAPI = retrofit.create(MyAPI.class); /// json data will be
converted to MyAPI type data



        // 3.. Create a call of model class and enqueue for processing

        Call<List<Model>> call = myAPI.getModelData();

        call.enqueue(new Callback<List<Model>>() {

            @Override
            public void onResponse(Call<List<Model>> call,
Response<List<Model>> response) {

// 4. Add data to the model class object & set to the text view
                List<Model> data = response.body();

                for(int i =0; i<data.size();i++)
                {
                    dataTv.append(" Sr no: "+ data.get(i).getId() + " \n "
+ data.get(i).getTitle() + "\n \n \n ");

                }
            }

            @Override
            public void onFailure(Call<List<Model>> call, Throwable t) {

            }
        });
    }
}
```

Reference Links For Learning Android

1. https://developer.android.com/
2. https://www.javatpoint.com/
3. https://www.tutlane.com/tutorial/android/
4. https://www.tutorialspoint.com/
5. https://abhiandroid.com/