# External Storage in Android with Example

Android gives various options for storing apps data which uses a file system similar to the disk-based system on computer platforms

- **App-Specific storage:** Store data files within internal volume directories or external. These data files are meant only for the app's use. It uses internal storage directories to save sensitive information such as a username and password that other app should not access.
- **Shared Storage:** Store data files such as images, audio, video, documents, etc. that the app may need to share with other apps.
- **Shared Preferences:** Store primitive data type such as integer, float, boolean, string, long in key-value pairs.
- **Databases:** Store structured data such as user-information(name, age, phone, email, address, etc.) into private databases.

Developers are advised to use the options available to store data depending upon **the space required**, **reliable data access**, and **privacy of data**. The data files saved over external storage devices are publicly accessible on shared external storage using USB mass storage transfer. Data files stored over external storage using a **FileOutputStream** object and can be read using a **FileInputStream** object**.**

## External Storage Availability

In order to avoid crashing the app first, we need to check storage SD Card is available for reading and write operations. The method **getExternalStorageState()** is used to determine the state of mounted storage media such as SD Card is missing, read-only or readable, and writable.

## Methods to Store data in External Storage

- **getExternalStoragePublicDirectory():** This is the present recommended method to keep files public and these files are not deleted even when the app is uninstalled from the system. For eg: Images clicked by the camera are still available even after we uninstall the camera.
- **getExternalFilesDir(String type):** This method is used to store private data that are specific to the app only. And data are removed as we uninstall the app.
- **getExternalStorageDirectory():** This method is not recommended. It is now absolute, and it is used to access external storage in older versions, API Level less than 7.

## Step by Step Implementation

**Step 1: Create a New Project**
**Step 2: Access Permission to External Storage**
To read and write data to external storage, the app required **WRITE_EXTERNAL_STORAGE** and **READ_EXTERNAL_STORAGE** system permission. These permissions are added to the AndroidManifest.xml file. Add these permissions just after the package name.

- XML

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

    <application ... >
        <activity android:name=".MainActivity" ... >
          ...
        </activity>
    </application>
</manifest>
```

**Step 3: Before creating the layout and corresponding java files let's add a few string attributes which we are using in our layout files**

Go to **app > res > values > string.xml** and insert the following code snippet

- XML

```
<resources>
  ...
    <string name="text_view_data">Enter the Text
Data</string>
    <string name="edit_text_data">Enter your
information</string>
    <string name="view_button">View Information</string>
    <string name="save_button_public">Save
Publicly</string>
```

```
    <string name="save_button_private">Save
Privately</string>

    <string name="text_view_saved_data">Saved Text
Data</string>
    <string name="saved_information">Click to view saved
information</string>
    <string name="back_button">Go Back</string>
    <string name="show_button_public">Public Data</string>
    <string name="show_button_private">Private
Data</string>
  ...
</resources>
```

Now go to **app > res > values > colors.xml** and change the color attributes as following in order to make App Bar more attractive.

- XML

```
<resources>
    <color name="colorPrimary">#0F9D58</color>
    <color name="colorPrimaryDark">#16E37F</color>
    <color name="colorAccent">#03DAC5</color>

    <color name="buttonColor">#0F9D58</color>
    <color name="textColor">#FFFFFF</color>
</resources>
```

Again go to **app > res > drawable** and create a new Drawable Resource File and name it as **button_layout**. In this, we are modifying our button style for a better UX/UI.

- XML

```
<?xml version="1.0" encoding="utf-8"?>
<shape
xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">

    <corners
        android:bottomLeftRadius="25dp"
        android:bottomRightRadius="25dp"
        android:radius="50dp"
```

```
        android:topLeftRadius="25dp"
        android:topRightRadius="25dp" />

    <padding
        android:bottom="0dp"
        android:left="0dp"
        android:right="0dp"
        android:top="0dp" />

    <size
        android:width="64dp"
        android:height="16dp" />

    <solid android:color="@color/buttonColor" />

</shape>
```

**Step 4: Working with the activity_main.xml file**

Go to **res > layout > activity_main.xml** and write down the following code. In this layout file, we are creating a multiline EditText View for getting the data from users and Buttons to save that data over internal and external storage media.

- 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">

    <Button
        android:id="@+id/saveButton_public"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:layout_below="@+id/editText_data"
        android:layout_marginEnd="48dp"
        android:layout_marginTop="8dp"
        android:background="@drawable/button_layout"
        android:onClick="savePublicly"
```

```
        android:padding="8dp"
        android:text="@string/save_button_public"
        android:textAllCaps="false"
        android:textColor="@color/textColor" />

    <Button
        android:id="@+id/saveButton_private"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_below="@+id/editText_data"
        android:layout_marginStart="48dp"
        android:layout_marginTop="8dp"
        android:layout_toEndOf="@+id/saveButton_public"
        android:background="@drawable/button_layout"
        android:onClick="savePrivately"
        android:padding="8dp"
        android:text="@string/save_button_private"
        android:textAllCaps="false"
        android:textColor="@color/textColor" />

    <Button
        android:id="@+id/viewButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/saveButton_public"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="16dp"
        android:background="@drawable/button_layout"
        android:onClick="viewInformation"
        android:padding="8dp"
        android:text="@string/view_button"
        android:textAllCaps="false"
        android:textColor="@color/textColor" />

    <EditText
        android:id="@+id/editText_data"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView_data"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="8dp"
        android:autofillHints=""
```

```
        android:hint="@string/edit_text_data"
        android:inputType="textMultiLine" />

    <TextView
        android:id="@+id/textView_data"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="8dp"
        android:text="@string/text_view_data"
        android:textAllCaps="true"
        android:textStyle="bold" />

</RelativeLayout>
```

### Step 5: Working with the MainActivity.java file

In MainActivity we define the functions that handled the onClick behavior of the buttons. And fetch the data from the EditText and save it in external storage publicly and privately. We also display a Toast message of the path where the data is stored.

- Java

```
import android.Manifest;
import android.content.Intent;
import android.os.Bundle;
import android.os.Environment;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;

public class MainActivity extends AppCompatActivity {

    // After API 23 the permission request for accessing external storage
is changed
```

```
    // Before API 23 permission request is asked by the user during
installation of app
    // After API 23 permission request is asked at runtime
    private int EXTERNAL_STORAGE_PERMISSION_CODE = 23;
    EditText editText;

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

        // findViewById return a view, we need to cast it to EditText View
        editText = (EditText) findViewById(R.id.editText_data);
    }

    public void savePublicly(View view) {
        // Requesting Permission to access External Storage
        ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.READ_EXTERNAL_STORAGE},
                EXTERNAL_STORAGE_PERMISSION_CODE);
        String editTextData = editText.getText().toString();

        // getExternalStoragePublicDirectory() represents root of external
storage, we are using DOWNLOADS
        // We can use following directories: MUSIC, PODCASTS, ALARMS,
RINGTONES, NOTIFICATIONS, PICTURES, MOVIES
        File folder =
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLO
ADS);

        // Storing the data in file with name as geeksData.txt
        File file = new File(folder, "geeksData.txt");
        writeTextData(file, editTextData);
        editText.setText("");
    }

    public void savePrivately(View view) {
        String editTextData = editText.getText().toString();

        // Creating folder with name GeekForGeeks
        File folder = getExternalFilesDir("Hello World !!");

        // Creating file with name gfg.txt
```

```
        File file = new File(folder, "file.txt");
        writeTextData(file, editTextData);
        editText.setText("");
    }

    public void viewInformation(View view) {
        // Creating an intent to start a new activity
        Intent intent = new Intent(MainActivity.this,
ViewInformationActivity.class);
        startActivity(intent);
    }

    // writeTextData() method save the data into the file in byte format
    // It also toast a message "Done/filepath_where_the_file_is_saved"
    private void writeTextData(File file, String data) {
        FileOutputStream fileOutputStream = null;
        try {
            fileOutputStream = new FileOutputStream(file);
            fileOutputStream.write(data.getBytes());
            Toast.makeText(this, "Done" + file.getAbsolutePath(),
Toast.LENGTH_SHORT).show();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (fileOutputStream != null) {
                try {
                    fileOutputStream.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

**Step 6: Create a new Empty Activity**

We will create a new activity and name it **ViewInformationActivity**. We use this activity to display the saved data from the external storage. So, first, we create a layout for this activity similar to the MainActivity layout. **activity_view_information.xml** layout code snippet:

- XML

```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=".ViewInformationActivity">

    <Button
        android:id="@+id/showButton_public"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:layout_below="@+id/textView_get_saved_data"
        android:layout_marginEnd="48dp"
        android:layout_marginTop="8dp"
        android:background="@drawable/button_layout"
        android:onClick="showPublicData"
        android:padding="8dp"
        android:text="@string/show_button_public"
        android:textAllCaps="false"
        android:textColor="@color/textColor" />

    <Button
        android:id="@+id/showButton_private"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_below="@+id/textView_get_saved_data"
        android:layout_marginStart="48dp"
        android:layout_marginTop="8dp"
        android:layout_toEndOf="@+id/showButton_public"
        android:background="@drawable/button_layout"
        android:onClick="showPrivateData"
        android:padding="8dp"
        android:text="@string/show_button_private"
        android:textAllCaps="false"
        android:textColor="@color/textColor" />

    <Button
        android:id="@+id/goBackButton"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_below="@+id/showButton_public"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="16dp"
        android:background="@drawable/button_layout"
        android:onClick="back"
        android:padding="8dp"
        android:text="@string/back_button"
        android:textAllCaps="false"
        android:textColor="@color/textColor" />

    <TextView
        android:id="@+id/textView_get_saved_data"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView_saved_data"
        android:layout_marginTop="8dp"
        android:gravity="center"
        android:hint="@string/saved_information" />

    <TextView
        android:id="@+id/textView_saved_data"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:gravity="center"
        android:text="@string/text_view_saved_data"
        android:textAllCaps="true"
        android:textStyle="bold" />

</RelativeLayout>
```

Now, we will create a corresponding java code for **ViewInformationActivity**. In this, we define methods **showPublicData()** and **showPrivateData()** over buttons that will fetch the data from the files saved to external storage and add the data to buffer, and then populate the data to TextView in order to show them. **ViewInformationAcitity.java** code snippet:

- Java

```
import android.content.Intent;
import android.os.Bundle;
import android.os.Environment;
```

```
import android.view.View;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;

public class ViewInformationActivity extends AppCompatActivity {

    TextView textView;

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

        // findViewById returns a view, we need to cast it into TextView
        textView = (TextView) findViewById(R.id.textView_get_saved_data);
    }

    public void showPublicData(View view) {
        // Accessing the saved data from the downloads folder
        File folder =
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLO
ADS);

        // file1 represent the file data that is saved publicly
        File file = new File(folder, "file1.txt");
        String data = getdata(file);
        if (data != null) {
            textView.setText(data);
        } else {
            textView.setText("No Data Found");
        }
    }

    public void showPrivateData(View view) {

        // GeeksForGeeks represent the folder name to access privately
saved data
        File folder = getExternalFilesDir("hello android !!");

        // gft.txt is the file that is saved privately
```

```
        File file = new File(folder, "file.txt");
        String data = getdata(file);
        if (data != null) {
            textView.setText(data);
        } else {
            textView.setText("No Data Found");
        }
    }

    public void back(View view) {
        Intent intent = new Intent(ViewInformationActivity.this,
MainActivity.class);
        startActivity(intent);
    }

    // getdata() is the method which reads the data
    // the data that is saved in byte format in the file
    private String getdata(File myfile) {
        FileInputStream fileInputStream = null;
        try {
            fileInputStream = new FileInputStream(myfile);
            int i = -1;
            StringBuffer buffer = new StringBuffer();
            while ((i = fileInputStream.read()) != -1) {
                buffer.append((char) i);
            }
            return buffer.toString();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (fileInputStream != null) {
                try {
                    fileInputStream.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
        return null;
    }
}
```

## Output: Run on Emulator