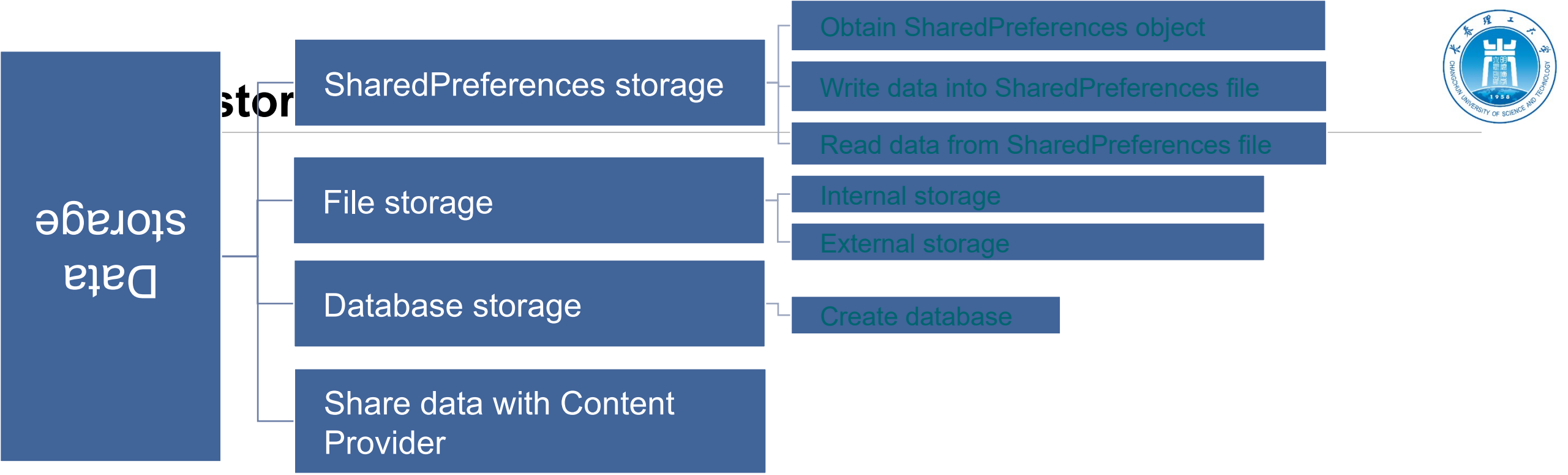
2019 级 - 本科

移动应用开发

张昕

zhangxin@cust.edu.cn

2022


Data storage
stor
SharedPreferences storage
Obtain SharedPreferences object
Write data into SharedPreferences file
Read data from SharedPreferences file
File storage
Internal storage
External storage
Database storage
Create database
Share data with Content Provider

SharedPreferences storage

- **Light-weight solution with simple APIs to manage data**
  - Such as configuration information of app, themes, score of game, etc.
  - Key-value pattern to store data, supporting to deal with Boolean, float, int, long, String, etc.
  - Data is stored in XML file, /data/data/<app module name>/shared_prefs
- **To obtain SharedPreferences object**
  - **Use getSharedPreferences() method**

    getSharedPreferences(String name, int mode)

    name: the file name, in xml format

    mode: specifies the access permission,

    MODE_PRIVATE: only within the same app,

    MODE_MULTI_PROCESS: across multiple processes or apps
  - **Use getPreferences() method**

    getPreferences(int mode)

SharedPreferences storage

- **Write data into SharedPreferences file**
  - **Steps**
    - **Call edit() method of SharedPererence class to obtain SharedPreference.Editor object**

      SharedPreferences.Editor editor = getSharedPreferences(“sp”, MODE_PRIVATE).edit();

    - **Add data into SharedPreferences.Editor object with putBoolean(), putString(), putInt()**

      editor.putString(“username”, userName);
      editor.putBoolean(“status”, false);
      editor.putInt(“level”, 5);

    - **Use commit() method to submit data to complete data storage**

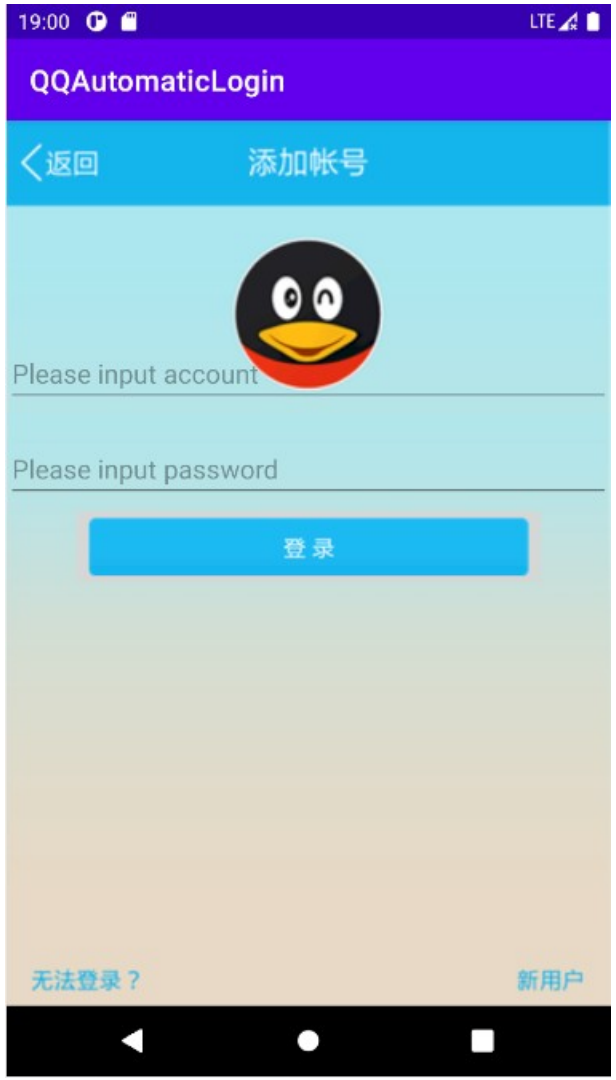
      editor.commit();

SharedPreferences storage

- **Read data from SharedPreferences file**
  - **With getXxx() method of SharedPreferences class**

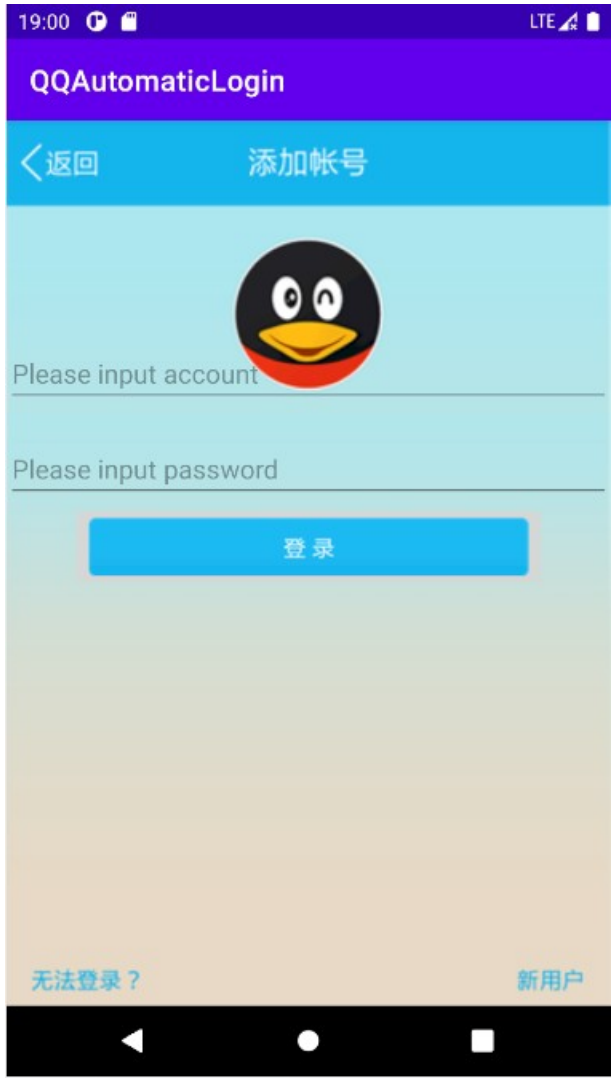
```
SharedPreferences sp = getSharedPreferences(“sp”, MODE_PRIVATE);
String username = sp.getString(“username”, “sp”);
int age = sp.getInt(“age”, 22);
Boolean status = sp.getBoolean(“status”, false);
        // the second parameter is the preset default value in case the data can not be read
```

Small Demo




```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/background"
    android:orientation="vertical"
    tools:context="com.example.qqautomaticlogin.MainActivity">
    <EditText
        android:id="@+id/username"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="@dimen/margin_top_username"
        android:hint="@string/username_hint" />
    <EditText
        android:id="@+id/password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="@dimen/margin_top_password"
        android:hint="@string/password_hint"
        android:inputType="textPassword" />
    <ImageButton
        android:id="@+id/login"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:src="@drawable/login" />
</LinearLayout>
```

Small Demo




```
public class MainActivity extends AppCompatActivity {
    private String mr = "sp", mrsoft = "cust";
    private String username, password;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final EditText usernameET = (EditText) findViewById(R.id.username);
        final EditText passwordET = (EditText) findViewById(R.id.password);
        ImageButton login = (ImageButton) findViewById(R.id.Login);
        final SharedPreferences sp = getSharedPreferences("sp", MODE_PRIVATE);
        final SharedPreferences.Editor editor = sp.edit();
        if (sp.getString("username", username) != null &&
                sp.getString("password", password) != null) {
            if (sp.getString("username", username).equals(mr) &&
                    sp.getString("password", password).equals(mrsoft)) {
                Intent intent = new Intent(MainActivity.this, MessageActivity.class);
                startActivity(intent);
            }
        } else {
            login.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    username = usernameET.getText().toString();
                    password = passwordET.getText().toString();
                    if (username.equals(mr) && password.equals(mrsoft)) {
                        Toast.makeText(MainActivity.this, "Account and password are correct",
                                Toast.LENGTH_SHORT).show();
                        Intent intent = new Intent(MainActivity.this, MessageActivity.class);
                        startActivity(intent);
                        editor.putString("username", username);
                        editor.putString("password", password);
                        editor.commit();
                        Toast.makeText(MainActivity.this, "Account and password has been saved",
                                Toast.LENGTH_SHORT).show();
                    } else {
                        Toast.makeText(MainActivity.this, "Account or passord wrong",
                                Toast.LENGTH_SHORT).show();
                    }
                }
            });
        }
    }
}
```

Small Demo


```xml
<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"
    android:background="@drawable/background2"
    tools:context="com.example.qqautomaticlogin.MessageActivity">
</RelativeLayout>
```

```java
public class MessageActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_message2);
    }
}
```

File storage

- **Internal storage**
  - FileOutputStream class provides openFileOuput() method
  - FileInputStream class provides openFileInput() method
- **External storage**
  - Environment class provides getExternalStorageDirectory() method to treat data (write and read) at external storage area

Internal storage

- **Position: /data/data/<packagename>/files/**
- **Write file**
  - **Steps**
    - **Obtain FileOutputStream object and call openFileOutput() method of FileOutputStream class**
    - **Call write() method write data into file**
    - **Call flush() method to empty buffer**
    - **Call close() method to close FileOutputStream**
  - **FileOutputStream openFileOutput(String name, int mode) throws FileNotFoundException**
    - **Mode:**
      - **MODE_PRIVATE**
      - **MODE_APPEND**
      - **MODE_WORLD_READABLE**
      - **MODE_WORLD_WRITEABLE**

```
Try{  //The name cannot contain any path information, such "/"
      FileOutputStream fos = openFileOutput("filedata.txt",
MODE_PRIVATE);
      fos.write("cust".getBytes());
      fos.flush();
      fos.close();
}catch(Exception e){
      e.printStackTrace();
}
```

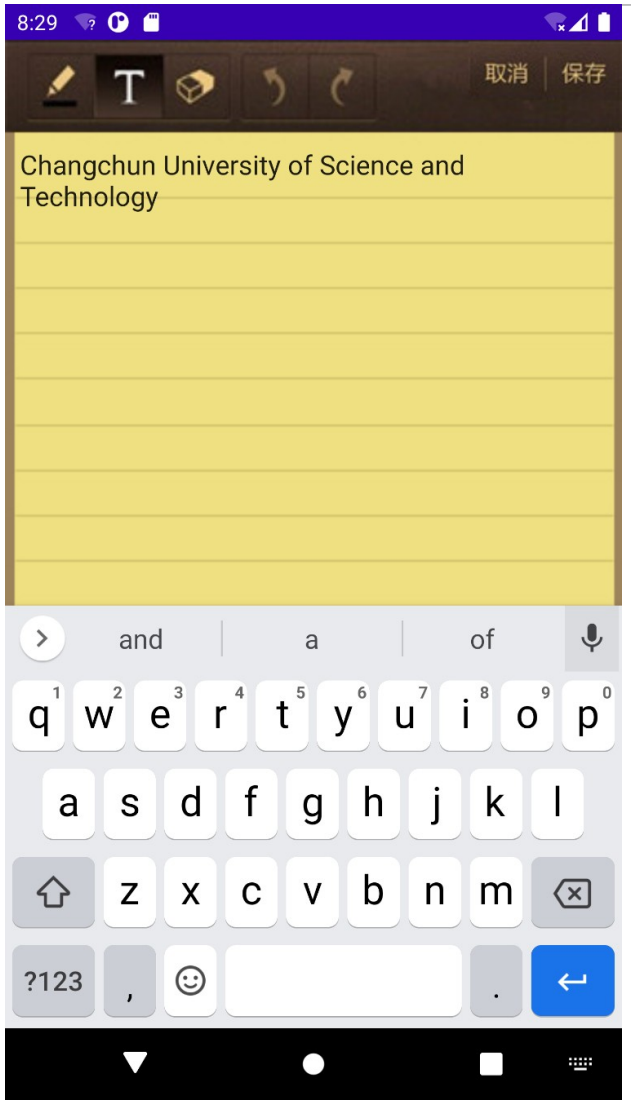
Internal storage

- **Read file**
  - **Steps**
    - **Obtain FileInputStream object and call openFileInput() method of FileOutputStream class**
    - **Call read() method read data from file**
    - **Call close() method to close FileOutputStream**
  - **FileInputStream openFileInput(String name) throws FileNotFoundException**

```
FileInputStream fis = openFileInput("filedata.txt);
byte[] buffer = new byte[fis.available()];
fis.read(buffer);
```

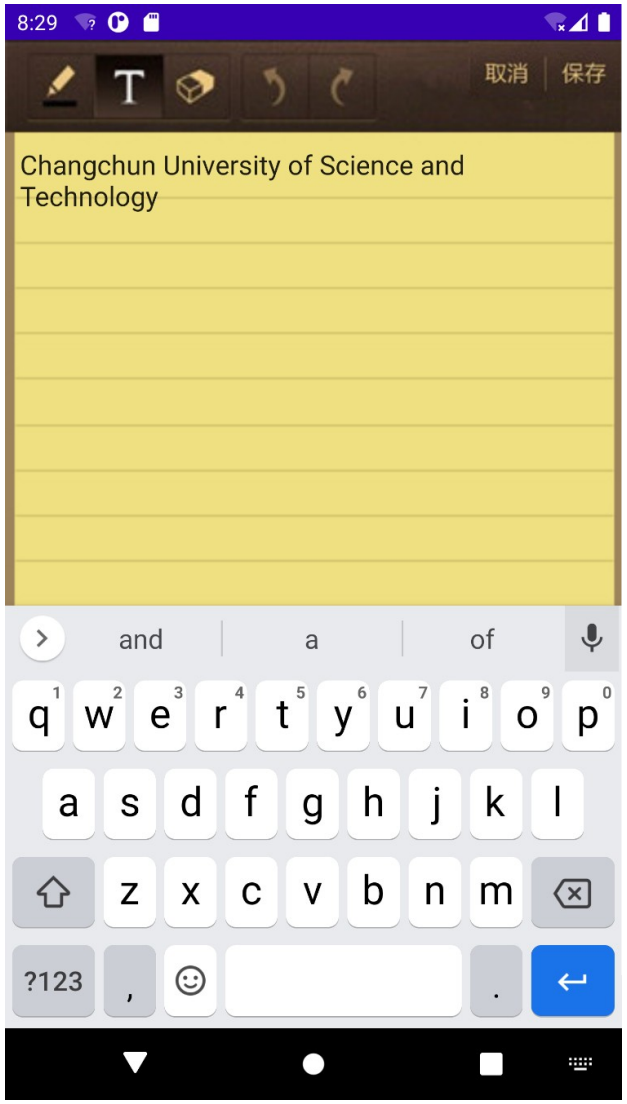
The name cannot contain any path information, such "/"

Small Demo




```
<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"
    android:background="@drawable/bg"
    tools:context="com.example.c11.MainActivity">
    <EditText
        android:id="@+id/editText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="@dimen/marginleft"
        android:layout_marginRight="@dimen/marginright"
        android:layout_marginTop="@dimen/margintop"
        android:background="@null"
        android:gravity="top" />
    <ImageButton
        android:id="@+id/btn_save"
        android:layout_width="@dimen/btn_width"
        android:layout_height="@dimen/btn_height"
        android:layout_alignParentRight="true"
        android:src="@drawable/btn1" />
    <ImageButton
        android:id="@+id/btn_cancel"
        android:layout_width="@dimen/btn_width"
        android:layout_height="@dimen/btn_height"
        android:layout_toLeftOf="@+id/btn_save"
        android:src="@drawable/btn2" />
</RelativeLayout>
```

Small Demo




```
public class MainActivity extends Activity {
    byte[] buffer = null;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final EditText etext = (EditText) findViewById(R.id.editText);
        ImageButton btn_save = (ImageButton) findViewById(R.id.btn_save);
        ImageButton btn_cancel = (ImageButton) findViewById(R.id.btn_cancel);
        btn_save.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                FileOutputStream fos = null;
                String text = etext.getText().toString();
                try {
                    fos = openFileOutput("memo", MODE_PRIVATE);
                    fos.write(text.getBytes());
                    fos.flush();
                } catch (FileNotFoundException e) {
                    e.printStackTrace();
                } catch (IOException e) {
                    e.printStackTrace();
                } finally {
                    if (fos != null) {
                        try {
                            fos.close();
                            Toast.makeText(MainActivity.this, "Successfully
saved", Toast.LENGTH_SHORT).show();
                        } catch (IOException e) {
                            e.printStackTrace();
                        }
                    }
                }
            }
        });
        FileInputStream fis = null;
        try {
            fis = openFileInput("memo");
            buffer = new byte[fis.available()];
            fis.read(buffer);
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (fis != null) {
                try {
                    fis.close();
                    String data = new String(buffer);
                    etext.setText(data);
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
        btn_cancel.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
            }
        });
    }
}
```

External storage

- **Use getExternalStorageDirectory() method of Environment class to obtain the directory of external media**
- **Permission should be declared**

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

Database storage

- **Android provides light-weight relational database, i.e., SQLite**
- **Create database**
  - **Use openOrCreateDatabase() method to create database**

    static SQLiteDatabase openOrCreateDatabase(String path, SQLiteDatabase.CursorFactory factory)

    path: specifies database file

    factory: initialize the cursor for further access to database

  - **Use SQLiteOpenHelper class to create database**

    Within the constructor of SQLiteOpenHelper class, call the methods of Context to create and open the particular database

    through extending SQLiteOpenHelper class and override onCreate() and onUpgrade() methods

Database storage

- **Data manipulation**
  - **SQLiteDatabase class provides insert(), update(), delete(), query() methods instead of typical SQL commands**

    public long insert(String table, String nullColmnHack, ContentValues values)
    - ✓ table: database table name
    - ✓ nullColmnHack: optional parameter to specify the specific attribute as null when values parameter is null, if the values parameter is not null, this parameter (i.e., nullColmnHack) could be null
    - ✓ values: specifies the specific attribute value. It stores data by key-value pattern

    public int update(String table, ContentValues values, String whereClause, String[] whereArgs)
    - ✓ values: specifies the target attribute and the corresponding value
    - ✓ whereClause: specifies the condition and allows to use placeholder, i.e., ?
    - ✓ whereArgs: specifies the corresponding value of the placeholder in whereClause, otherwise, is set as null

Database storage

- **Data manipulation**
  - **SQLiteDatabase class provides insert(), update(), delete(), query() methods instead of typical SQL commands**

public int delete(String table, String whereClause, String[] whereArgs)

- ✓ whereClause: specifies the condition and allows to use placeholder, i.e., ?
- ✓ whereArgs: specifies the corresponding value of the placeholder in whereClause, otherwise, is set as null

public Cursor query(boolean distinct, String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy, String limit)

- ✓ columns: specifies the target column, if set as null, then return all the columns
- ✓ selection: specifies where clause, allows to use placeholder, i.e., ?
- ✓ selectionArgs: specifies the condition value of where clause, if there exists placeholder. If no placeholder, set as null
- ✓ limit: sets the number of returned items

- ☐ **The returned Cursor object is a pointer to dataset and it provides multiple moving methods to access data**

Database storage

Data manipulation

public Cursor query(boolean distinct, String table, String[] columns, String selection, String[] selectionArgs,
String groupBy, String having, String orderBy, String limit)

- **The returned Cursor object is a pointer to dataset and it provides multiple moving methods to access data**

moveToFirst()	Move cursor to the first row
moveToNext()	Move cursor to the next row
moveToPrevious()	Move cursor to the previous row
getCount()	Return the number of all the rows
getColumnIndexOrThrow()	Return the index of the target attribute name, if not exists, throw some Exception
getColumnName()	Return the attribute name of the specific index
getColumnNames()	Return the String array corresponding to the attribute name
getColumnIndex()	Return the index of the specific attribute name
moveToPosition()	Move cursor to the target row
getPosition()	Return the current position where the cursor is located

Demo


```
<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"
    android:background="@drawable/bg1"
    tools:context="com.example.databasedictionary.MainActivity">
    <EditText
        android:id="@+id/search_et"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="@dimen/search_et_marginTop"
        android:background="@null"
        android:hint="@string/search_et_hint"
        android:paddingLeft="@dimen/search_et_paddingLeft" />
    <ImageButton
        android:id="@+id/search_btn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:layout_marginTop="@dimen/search_btn_margin"
        android:background="@color/colorLucency"
        android:gravity="bottom"
        android:paddingRight="@dimen/search_btn_margin"
        android:src="@drawable/btn1"
        android:text="@string/search_btn_text" />
    <ListView
        android:id="@+id/result_listView"
        android:layout_width="@dimen/result_listView_width"
        android:layout_height="@dimen/result_listView_height"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="@dimen/result_listView_marginBottom">
    </ListView>
    <Button
        android:id="@+id/btn_add"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="@dimen/add_btn_marginLeft"
        android:layout_marginTop="@dimen/add_btn_marginTop"
        android:background="@drawable/add" />
</RelativeLayout>
```

Demo


```
public class MainActivity extends Activity {
    private DBOpenHelper dbOpenHelper;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        dbOpenHelper = new DBOpenHelper(MainActivity.this, "dict.db", null, 1);
        final ListView listView = (ListView) findViewById(R.id.result_listView);
        final EditText etSearch = (EditText) findViewById(R.id.search_et);
        ImageButton btnSearch = (ImageButton) findViewById(R.id.search_btn);
        Button btn_add = (Button) findViewById(R.id.btn_add);
        btn_add.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MainActivity.this, AddActivity.class);
                startActivity(intent);
            }
        });
        btnSearch.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                String key = etSearch.getText().toString();
                Cursor cursor=dbOpenHelper.getReadableDatabase().query("dict",null
                        ,"word = ?",new String[]{key},null,null,null);
                ArrayList<Map<String, String>> resultList = new ArrayList<Map<String, String>>();
                while (cursor.moveToNext()) {
                    Map<String, String> map = new HashMap<>();
                    map.put("word", cursor.getString(1));
                    map.put("interpret", cursor.getString(2));
                    resultList.add(map);
                }
                if (resultList == null || resultList.size() == 0) {
                    Toast.makeText(MainActivity.this,
                            "No record", Toast.LENGTH_LONG).show();
                } else {
                    SimpleAdapter simpleAdapter = new SimpleAdapter(MainActivity.this, resultList,
                            R.layout.result_main,
                            new String[]{"word", "interpret"}, new int[]{
                            R.id.result_word, R.id.result_interpret});
                    listView.setAdapter(simpleAdapter);
                }
            }
        });
    }
```

```
    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (dbOpenHelper != null) {
            dbOpenHelper.close();
        }
    }
}
```

Demo


```java
public class DBOpenHelper extends SQLiteOpenHelper {
    final String CREATE_TABLE_SQL =
            "create table dict(_id integer primary " + "key autoincrement , word , detail)";
    public DBOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, null, version);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(CREATE_TABLE_SQL);
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        System.out.println("---Version Update-----" + oldVersion + "--->" + newVersion);
    }
}
```

Demo


```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/bg2"
    android:orientation="vertical"
    tools:context="com.example.databasedictionary.AddActivity">
    <EditText
        android:id="@+id/add_word"
        android:layout_width="@dimen/add_word_width"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="@dimen/add_word_marginTop"
        android:hint="@string/add_word_hint" />
    <EditText
        android:id="@+id/add_interpret"
        android:layout_width="@dimen/add_interpret_width"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="@dimen/add_marginTop"
        android:hint="@string/add_interpret_hint" />
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal">
        <ImageButton
            android:id="@+id/save_btn"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="120dp"
            android:layout_marginTop="180dp"
            android:background="@color/colorLucency"
            android:src="@drawable/save" />
        <ImageButton
            android:id="@+id/cancel_btn1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="20dp"
            android:layout_marginTop="180dp"
            android:background="@color/colorLucency"
            android:src="@drawable/cancel" />
    </LinearLayout>
</LinearLayout>
```

Demo

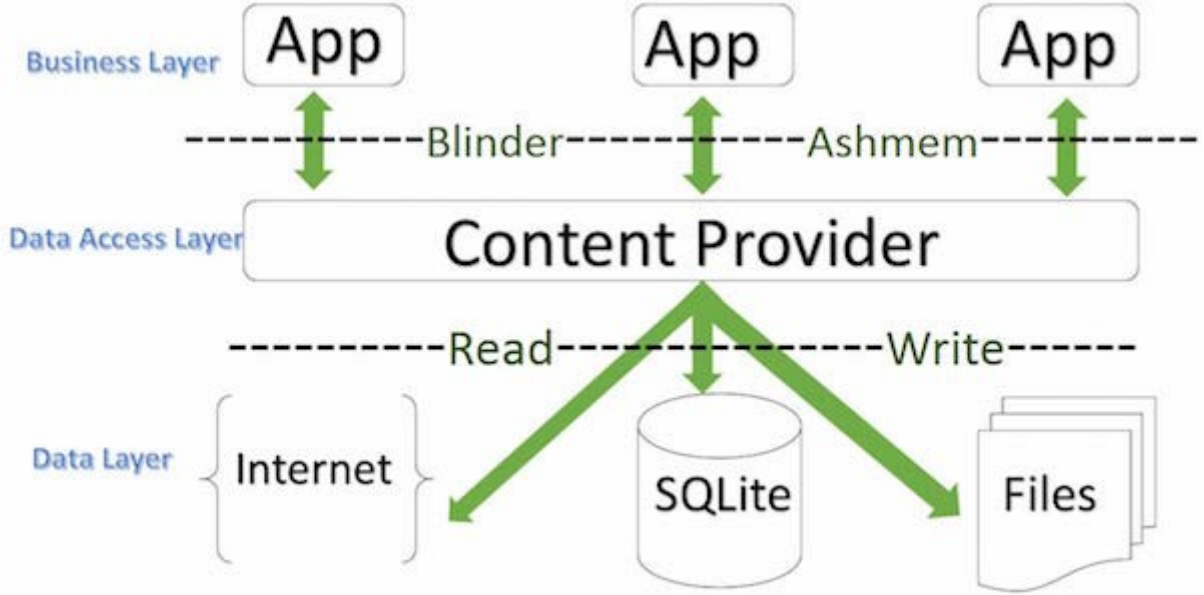

```
public class AddActivity extends Activity {
    private DBOpenHelper dbOpenHelper;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add);
        dbOpenHelper = new DBOpenHelper(AddActivity.this, "dict.db", null, 1);
        final EditText etWord = (EditText) findViewById(R.id.add_word);
        final EditText etExplain = (EditText) findViewById(R.id.add_interpret);
        ImageButton btn_Save = (ImageButton) findViewById(R.id.save_btn);
        ImageButton btn_Cancel = (ImageButton) findViewById(R.id.cancel_btn1);
        btn_Save.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String word = etWord.getText().toString();
                String explain = etExplain.getText().toString();
                if (word.equals("") || explain.equals("")) {
                    Toast.makeText(AddActivity.this,
                            "Word or translation is empty", Toast.LENGTH_SHORT).show();
                } else {
                    insertData(dbOpenHelper.getReadableDatabase(), word, explain);
                    Toast.makeText(AddActivity.this,
                            "Succesfully add new word", Toast.LENGTH_LONG).show();
                }
            }
        });
        btn_Cancel.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(AddActivity.this, MainActivity.class);
                startActivity(intent);
            }
        });
    }
    private void insertData(SQLiteDatabase readableDatabase, String word, String explain) {
        ContentValues values = new ContentValues();
        values.put("word", word);
        values.put("detail", explain);
        readableDatabase.insert("dict", null, values);
    }
}
```

Demo


```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/word"
            android:textSize="@dimen/result_textSize"/>
        <TextView
            android:id="@+id/result_word"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="@dimen/result_textSize"/>
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/interpret"
            android:textSize="@dimen/result_textSize"/>
        <TextView
            android:id="@+id/result_interpret"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="@dimen/result_textSize"/>
    </LinearLayout>
</LinearLayout>
```

Content Provider making data sharing

- **Developers determine the principle of storing data by Content Provider**



Decide whether to need a content provider

- To build a content provider if you want to provide one or more of the following features:
  - You want to offer complex data or files to other applications.
  - You want to allow users to copy complex data from your app into other apps.
  - You want to provide custom search suggestions using the search framework.
  - You want to expose your application data to widgets.
  - You want to implement the AbstractThreadedSyncAdapter, CursorAdapter, or CursorLoader classes.
- You don't need a provider to use databases or other types of persistent storage if the use is entirely within your own application and you don’t need any of the features listed above.

Decide whether to need a content provider

- **A content provider offers data in two ways:**
  - **File data**
    - Data that normally goes into files, such as photos, audio, or videos. Store the files in the application's private space.
    - In response to a request for a file from another application, the provider offers a handle to the file.
  - **"Structured" data**
    - Data that normally goes into a database, array, or similar structure. Store the data in a form that's compatible with tables of rows and columns.
      - A row represents an entity, such as a person or an item in inventory.
      - A column represents some data for the entity, such a person's name or an item's price.
    - A common way to store this type of data is in an SQLite database, but programmers can use any type of persistent storage.

Content Provider making data sharing

- Developers determine the principle of storing data by Content Provider
  - All Content Providers implements a set of common methods for building up insertion, deletion, update and query
  - However, developers usually employ ContentResolver object to manipulate Content Provider with getContentResolver() method
    - Activity and other component have been equipped with getContentResolver() method to obtain ContentResolver object

```
ContentResolver cr = getContentResolver();
```

Content Provider making data sharing

- The methods offered by ContentResolver enable to get any data in Content Provider
- During the query
  - Android system verifies the target Content Provider exists and is running
  - System will automatically initializes all the ContentProvider objects, by which developers doesn’t need to do anything and even to use ContentProvider object
- Each type of Content Provider has only one instance that could communicate with the multiple ContentResolver objects belong to the different apps/processes
  - Communication among the multiple processes is treated by ContentProvider class and ContentResolver class

Data model in Content Provider

- Content Provider utilizes database model-based simple tables to offer data
  - Row: one record
  - Column: data type and attribute
  - Each record has one numeric type of _ID attribute as the identity by default
- Query to Content Provider returns a Cursor object to traverse all the entities and cells
  - Referring to the various types of data, a corresponding specific method of accessing data

URI in Content Provider

- **Each Content Provider offers public URI(s) to label its dataset(s)**
  - **Each URI has a prefix “content://”**

If access multiple records, record ID should be omitted

Create Content Provider

- A Content Provider is created with extending ContentProvider class
- Steps
  - Extends ContentProvider class to offer the means for accessing data
  - Declare Content Provider within AndroidManifest.xml

Extending ContentProvider class

- **Six abstract methods should be implemented**
  - **public boolean onCreate()**
    - **Initialize Content Provider**
  - **public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder)**
    - **Return data to users**
  - **public Uri insert(Uri uri, ContentValue values)**
    - **Insert data into Content Provider**
  - **public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs)**
    - **Update the existed data in Content Provider**
  - **public int delete(Uri uri, String selection, String[] selectionArgs)**
    - **Delete data in Content Provider**
  - **public String getType(Uri uri)**
    - **Return MIME type of data in Content Provider**

Utilizing Content Provider

- **Android system offers a number of preset Content Providers corresponding to the various common data types**
  - **The Content Providers are located at android.provider package**

Browser	To manage the data of browser, such as favorites, history, etc.
CallLog	To manage the history of calls
Contacts	To manage the information of contact book
LiveFolders	To manage the specific folder filled with the content offered by Content Provider
MediaStore	To manage the multimedia information, such as audio, video, images
Setting	To manage the setting and preferences of the system, such as Bluetooth, ringtone, etc.
SearchRecentSuggestions	To assistant applications to suggest some hints for search
SyncStateContract	To associate the data with data array account. This may be used by providers that want to store this data in a standard way.
UserDictionary	To provide the customized words to input method through predicting the further input text

Intents and data access

- Applications can access a content provider indirectly with an Intent.
- The application does not call any of the methods of ContentResolver or ContentProvider. Instead, it sends an intent that starts an activity, which is often part of the provider's own application.
- The destination activity is in charge of retrieving and displaying the data in its UI.
  - Depending on the action in the intent, the destination activity may also prompt the user to make modifications to the provider's data.
  - An intent may also contain "extras" data that the destination activity displays in the UI; the user then has the option of changing this data before using it to modify the data in the provider.

Intents and data access

- Applications can access a content provider indirectly with an Intent.
- The application does not call any of the methods of ContentResolver or ContentProvider. Instead, it sends an intent that starts an activity, which is often part of the provider's own application.
- The destination activity is in charge of retrieving and displaying the data in its UI.
  - Depending on the action in the intent, the destination activity may also prompt the user to make modifications to the provider's data.
  - An intent may also contain "extras" data that the destination activity displays in the UI; the user then has the option of changing this data before using it to modify the data in the provider.