项目5

记忆的仓库——备忘录

【教学导航】

学习目标	(1) 了解 SharedPreferences 实现简单的数据存储方法。 (2) 理解 Android 中的文件存储方法。 (3) 掌握摄像头调用和相册获取图片的方法。 (4) 了解 Android 运行时权限设置。 (5) 具备较复杂案例的设计开发能力。
教学方法	任务驱动法、理论实践一体化、探究学习法、分组讨论法
课时建议	8 课时

5.1 SharedPreferences 存储

Android 中常用的三大存储方式分别是 SharedPreferences 存储、文件存储和数据库存储。其中，SharedPreferences 存储在程序中有少量的数据需要保存时使用，它存储数据的格式很简单，采用键值对的方式来存储数据。文件存储主要用来存储资源文件，如一篇日记、一首歌等。数据库存储常用于当程序中有大量数据需要存储、访问时，就需要借助于数据库，Android 内置了 SQLite 数据库。本节将针对 SharedPreferences 存储进行讲解。


微课视频

5.1.1 SharedPreferences 简介

SharedPreferences 是 Android 平台上一个轻量级的存储类，常用于存储一些应用程序的配置参数，例如用户名、密码、是否打开同步、是否打开音响、自定义参数的设置等。SharedPreferences 中存储的数据是以 key-value(键值)对的形式保存在 XML 文件中，需要注意的是，SharedPreferences 中的 value 值只能是 float、int、long、boolean、String、StringSet 类型数据。

1. SharedPreferences 常用方法

SharedPreferences 提供了一系列的方法用于获取应用程序中的数据，具体如表 5.1 所示。

表 5.1 SharedPreferences 常用方法

属 性	功能描述	实 例
getSharedPreferences (String name, String mode)	获取 SharedPreference 的实例对象	SharedPreference sp = this. getSharedPreferences("myinfo",0)
getString(String key,String defValue)	获取 SharedPreferences 中指定的 key 对应的 String 值	getSharedPreferences("myinfo", 0). getString("name","");
getInt(String key,int defValue)	获取 SharedPreferences 中指定的 key 对应的 int 值	getSharedPreferences("myinfo", 0). getInt("age",0);
getBoolean (String key, boolean defValue)	获取 SharedPreferences 中指定的 key 对应的 boolean 值	getSharedPreferences("info", 0). getBoolean("married",false);
getFloat(String key,float defValue)	获取 SharedPreferences 中指定的 key 对应的 float 值	getSharedPreferences("info",0). getFloat("weight",0);
getLong(String key,long defValue)	获取 SharedPreferences 中指定的 key 对应的 long 值	getSharedPreferences("info",0). getLong("birthdistance",0);

表 5.1 中，使用 SharedPreferences 类存储数据时，首先需要通过 context. getSharedPreferences(String name, int mode) 获取 SharedPreferences 的实例对象，其中 context 表示上下文环境，在 Activity 中可以直接使用 this 代表上下文，如果不是在 Activity 中则需要传入一个 Context 对象获取上下文。

getSharedPreferences(String name,String mode)方法中，第一个参数 name 表示文件名，如果指定的文件不存在则会创建一个新文件，该文件存放在 data/data/< package name >/ shared_prefs 目录下。第二个参数 mode 用于指定文件操作模式，目前只有 MODE_ PRIVATE 这一种模式可选，指定该 SharedPreferences 数据只能被本应用程序读写，它是默认的操作模式，和直接传入 0 效果是相同的，其他几种模式从 Android 4.2 版本开始均已废弃。

getXxx(String,xxx value)获取 SharedPreferences 数据里指定 key 对应的 value。如果该 key 不存在，则返回默认值 value。其中，xxx 可以是 boolean、float、int、long、String 等多种基本类型。

2. SharedPreferences. Editor 常用方法

SharedPreferences 对象本身只能获取数据，并不支持数据的存储和修改，数据的存储和修改需要通过 SharedPreferences. Editor 对象来实现。要想获取 Edit 实例对象，需要调用 SharedPreferences. Editor 的 edit()方法，Editor 提供一系列方法来向 SharedPreferences 写入数据，如表 5.2 所示。

表 5.2 SharedPreferences.Editor 常用方法

方　法	功能描述	实　例
SharedPreferences.Editor edit()	创建一个 Editor 对象	SharedPreferences.Editor editor=this.getSharedPreferences("myinfo",0).edit();
SharedPreferences.Editor putString(String key,String value)	向 SharedPreferences 中存入指定 key 对应的 String 值	editor.putString("name","张三");
SharedPreferences.Editor putInt(String key,int value)	向 SharedPreferences 中存入指定 key 对应的 int 值	editor.putInt("age",20);
SharedPreferences.Editor putBoolean(String key,boolean value)	向 SharedPreferences 中存入指定 key 对应的 boolean 值	editor.putBoolean("married",false);
SharedPreferences.Editor putFloat(String key,float value)	向 SharedPreferences 中存入指定 key 对应的 float 值	editor.putFloat("weight",76.5f);(其中,f 表示 float)
SharedPreferences.Editor putLong(String key,long value)	向 SharedPreferences 中存入指定 key 对应的 long 值	editor.putLong("birthdistance",1546729204391);(其中,l 表示 long)
SharedPreferences.Editor remove(String key)	删除 SharedPreferences 中指定 key 对应的数据	editor.remove("weight");
boolean commit()	提交数据,返回 boolean 表明提交是否成功	editor.commit();
SharedPreferences.Editor apply()	提交数据,没有返回值	editor.apply();

SharedPreferences.Editor clear()	清除 SharedPreferences 中的所有数据	SharedPreferences.Editor editor1=getSharedPreferences("myinfo",0).edit();
		editor1.clear();
		editor1.commit();


微课视频

5.1.2 SharedPreferences 的使用

1. 将数据存储到 SharedPreferences 中

使用 SharedPreferences 存储数据时,需要先获取 SharedPreferences 对象,通过该对象的 edit()方法获取到 Editor 对象,然后通过 Editor 对象的相关方法存储数据,具体代码如下。

```
SharedPreferences.Editor editor = getSharedPreferences("myinfo",0).edit();   //获取编辑器 editor
editor.putString("name","张三");                  //存入 String 类型数据
editor.putInt("age",20);                          //存入 int 类型数
editor.putBoolean("married",false);               //存入 boolean 类型数据
editor.putFloat("weight",76.8f);                  //存入 float 类型数据
editor.putLong("birthdistance",1546729204391);    //存入 long 类型数据
editor.commit();                                  //提交数据,保存成功
```

2. 从 SharedPreferences 中读取数据

SharedPreferences 获取数据比较简单，只需要创建 SharedPreferences 对象，然后使用该对象获取相应 key 的值即可。具体代码如下。

```
String myname = getSharedPreferences("myinfo",0).getString("name",""); //获取用户名
int myage = getSharedPreferences("myinfo",0).getInt("age",0);          //获取年龄
float weight = getSharedPreferences("info",0).getFloat("weight",0);    //获取体重
boolean mymarried = getSharedPreferences("info",0).getBoolean("married",false);
                                                                        //获取婚姻状态
long mybirthdistance = getSharedPreferences("info",0).getLong("birthdistance",0);
                                                                    //获取出生到现在的时长
```

3. 删除 SharedPreferences 中的数据

SharedPreferences 删除数据与存储数据类似，同样需要先获取到 Editor 对象，然后通过该对象删除数据，最后提交，具体代码如下。

```
SharedPreferences.Editor editor1 = getSharedPreferences("myinfo",0).edit();
                                                        //获取编辑器对象,取名为 editor1
editor1.remove("name");          //删除一条数据
editor1.clear();                 //删除所有数据
editor1.commit();                //提交数据,保存成功
```

SharedPreferences 使用很简单，但一定要注意以下两点。

(1) 存入数据和删除数据时，一定要在最后使用 editor.commit()方法提交数据。

(2) 读取数据的 key 值与存入数据的 key 值的数据类型要一致，否则查找不到数据。

4. SharedPreferences 案例

1) 案例分析

(1) 界面分析。布局界面中有 5 个控件，其中两个 EditText 控件用于输入用户名和密码，复选框 CheckBox 供用户选择是否需要记住密码，“登录”按钮保存信息并跳转到第二个页面。

(2) 设计思路。布局界面中的控件通过添加属性的方式达到用户需求，在登录按钮单击事件内部实现用户名、密码、复选框状态等数据的存储，下次启动 App 时，根据复选框的状态决定是否显示存储的用户名和密码。

2) 实现步骤

(1) 创建一个新的工程，工程名为 ZSSharedPreferences。

(2) 切换工程的 Project 项目结构。选择该模式下方的 App，依次展开，便看到工程的布局界面和工程的类文件，其中，activity_main.xml 是布局界面，MainActivity.java 为类文件。

(3) 在工程中添加一个新的页面。右击 com.example.zssharedpreferences 包→New→Activity→Empty Activity，会弹出一个创建活动的对话框，将活动命名为 SecondActivity，默认勾选 Generate Layout File 关联布局界面，布局界面名称为 activity_second，但不要勾选 Launcher Activity。单击 Finish 按钮，便完成了第二个页面的添加。

(4) 准备一张图片,图片名为 bgtwo.jpg,将其粘贴到 app 目录结构中 res 下方的 drawable 文件夹下,作为登录页面的背景图片。

(5) 修改布局界面。在 app 目录下的结构中,双击 layout 文件夹下的 activity_main.xml 文件,便可打开布局编辑器,切换到 Text 选项卡,输入代码如下。

```
<?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:background = "@drawable/bgtwo"
    tools:context = ".MainActivity">
    < ImageView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
       android:src = "@mipmap/ic_launcher_round"
        android:layout_marginTop = "150dp"
        android:layout_marginBottom = "50dp"
        android:layout_gravity = "center"/>
    < EditText
        android:id = "@ + id/editText_inputname"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:hint = "请输入用户名"
        android:textColor = "#000000"
        android:textSize = "25sp"
        android:textStyle = "bold"/>
    < EditText
        android:id = "@ + id/editText_inputpwd"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:hint = "请输入密码"
        android:inputType = "textPassword"
        android:textColor = "#000000"
        android:textSize = "25sp"
        android:textStyle = "bold"/>
    < CheckBox
        android:id = "@ + id/checkBox_reme"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "记住密码"
        android:layout_gravity = "right"
        android:layout_marginRight = "10dp"
        android:textColor = "#000000"
        android:textSize = "20sp"
```

```
        android:textStyle = "bold"
        android:layout_marginTop = "10dp"
        android:layout_marginBottom = "10dp"/>
    <Button
        android:id = "@ + id/button_login"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:text = "登录"
        android:textColor = "#000000"
        android:textSize = "25sp"
        android:textStyle = "bold"/>
</LinearLayout>
```

上述代码中,根节点为线性布局,方向为 vertical,其内部放置了一个图片控件、输入用户名的 EditText 控件和输入密码的 EditText 控件,以及复选框 checkBox 控件,方便用户选择是否记住密码。最下方是 Button 控件,用来实现数据存储和跳转功能。

(6) 实现第一个 Activity 页面的功能。打开 MainActivity.java 类文件,修改 MainActivity 的代码如下。

```
public class MainActivity extends AppCompatActivity {
//第一步:定义对象
EditText edit_name,edit_pwd;
CheckBox check_reme;
Button btn_login;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //第二步:绑定控件
        edit_name = findViewById(R.id.editText_inputname);
        edit_pwd = findViewById(R.id.editText_inputpwd);
        check_reme = findViewById(R.id.checkBox_reme);
        btn_login = findViewById(R.id.button_login);
        //第三步:按钮单击事件
        btn_login.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //单击按钮将输入的用户名、密码、复选框的状态保存起来
         SharedPreferences.Editor editor = getSharedPreferences("myfile",0).edit();
         editor.putString("name", edit_name.getText().toString());
         editor.putString("pwd",edit_pwd.getText().toString());
         editor.putBoolean("st",check_reme.isChecked());
         editor.commit();
         Intent intent = new Intent(MainActivity.this, SecondActivity.class);
         startActivity(intent);
         finish();
            }
        });
```

```
            //第四步:如果选中了"记住密码"复选框,下一次启动,能够获取用户名和密码并显示出来
            String myname = getSharedPreferences("myfile",0).getString("name","");
            String mypwd = getSharedPreferences("myfile",0).getString("pwd","");
            Boolean myst = getSharedPreferences("myfile",0).getBoolean("st",false);
            if(myst == true){
                edit_name.setText(myname);
                edit_pwd.setText(mypwd);
                check_reme.setChecked(true);
            }else{
                edit_name.setText("");
                edit_pwd.setText("");
                check_reme.setChecked(false);
            }
        }
    }
```

上述代码中,在按钮单击事件内部,首先使用 SharedPreferences 将输入的用户名、密码、复选框的状态存储起来,然后实现页面的跳转,即从当前页面跳转到第二个页面。下次启动时,从 SharedPreferences 中获取用户名、密码和复选框的状态等数据信息,根据获取的复选框的状态进行判断,如果为真,说明用户选中了"记住密码",此时将获取的用户名和密码显示到 EditText 控件上,并将复选框设置为选中。如果为假,说明用户没有选中"记住密码",EditText 控件内容设置为空,并将复选框设置为未选中。

(7) 运行程序,效果如图 5.1(a)所示,在界面中输入用户名和密码,选中"记住密码"复选框,单击"登录"按钮,将用户名和密码保存并跳转到第二个页面,再次运行程序,效果如图 5.1(b)所示。

(a) (b)

图 5.1 程序运行效果

(8) 查看 SharedPreferences 中保存的数据。在百度中下载 RE 文件管理器,把下载好的 RE 文件管理器软件拖放到模拟器中,模拟器便会开始安装 RE 文件管理器,大约 5s 即可安装成功。在模拟器中打开 RE 文件管理器,挂载为可读写模式,根据文件的存储路径依次操作:data/data/< package name >/shared_prefs/文件名,单击 data 文件夹,在打开的模拟器页面中再次单击 data 文件夹,在列表中找到工程的包名 com. example. zssharedpreferences,在打开的页面中双击 Shared_prefs 将其打开,看到了保存的文件名 myfile. xml,双击查看保存的用户名和密码信息。

图 5.2 在模拟器中查看 SharedPreferences 中保存信息

5.2 Android 的文件存储

文件存储是 Android 中最基本的一种数据存储方式,它不会对存储的内容进行任何的格式化处理,它与 Java 中的文件存储类似,都是通过 I/O 流的形式把数据原封不动地存储到文档中。不同的是,Android 中的文件存储分为内部存储和外部存储,本节分别针对这两种存储方式进行详细的讲解。

5.2.1 内部存储


微课视频

内部存储不是内存,它是手机中的一块存储区域,是系统本身和系统应用程序主要的数据存储位置。内部存储是指将应用程序中的数据以文件方式存储到设备的内部存储空间中,与 SharedPrefences 存储位置类似,文件存储的位置也位于 data/data/< package name >/files/目录下。内部存储方式所存储的文件被其所创建的应用程序私有,如果其他应用程序要操作本应用程序中的文件,需要设置权限。当创建的应用程序被卸载时,其内部存储文件也随之删除。

1. 文件内部存储常用方法

Android 提供了 openFileInput()和 openFileOutput()两种方法,向应用程序"读取"和"写入"数据流。按照流向分可以分为输入流和输出流,输入流只能从中读取数据,不能写入数据;输出流只能向其中写入数据,不能读取数据。写入和读取数据时,按照操作的数据单元分为字节流和字符流,字节流操作的数据单元是 8 位的字节,字符流操作的数据单元是 16 位的字节。

系统提供了文件内部存储的方法,如表 5.3 所示。

表 5.3 文件内部存储方法

方　法	功能描述	实　例
将数据存储到文件中的方法		
openFileOutput (String name, int model)	得到一个文件输出流对象 FileOutputStream 类向文件中写入数据	FileOutputStream fout = openFileOutput("zs. txt",0)
write(int b) throws IOException	一次写一个字节,b 表示要写入的字节	fout. write(edit_input. getText (). toString(). getBytes())
write (byte [ ] b) throws IOException:	一次写一个字节数组	
write(byte[] b, int off, int len) throws IOException:	一次写一部分字节数组	
close()throws IOException:	关闭此文件输出流并释放与此流有关的所有系统资源	fout. close()
从文件中读取数据的方法		
openFileInput (String name)	得到一个文件输入流对象 FileInputStream 类 从某个文件中读取字节	FileInputStream fin = openFileInput("zs. txt");
public int read() throws IOException	从此输入流中读取一个数据字节	
public int read (byte [ ] b) throws IOException	从此输入流中将最多 b. length 个字节的数据读入一个字节数组中	byte[] arr = new byte[fin. available()]; fin. read(arr);
public int read(byte[] b, int off, int len) throws IOException	从此输入流中将最多 len 个字节的数据读入一个字节数组中	
public voidclose() throws IOException	关闭此文件输入流并释放与此流有关的所有系统资源	fout. close();
删除文件的方法		
deleteFile(filename)	可以根据文件名,删除文件	deleteFile("zs. txt");

上述方法中,openFileOutput()得到一个文件输出流对象 FileOutputStream,用于打开应用程序中对应的输出流,将数据存储到指定的文件中。其中,参数 name 表示文件名;mode 表示文件的操作模式,有两种模式可选:MODE_PRIVATE 和 MODE_APPEND,其中,MODE_PRIVATE 是默认的操作模式,表示当指定同名文件的时候,所写入的内容会覆盖原文件中的内容,而 MODE_APPEND 是追加模式,表示如果该文件已存在,就往文件里面追加内容,不存在就创建新的文件。openFileInput()得到一个文件输入流对象 FileInputStream,用于打开应用程序中对应的输入流,读取文件中的内容并显示出来。

Android 中文件读写的原理:首先,所有文件的存储都是字节的存储。其次,在磁盘上保留的并不是文件的字符,而是先把字符编码转换成字节,再存储这些字节到磁盘。最后,在读取文件特别是文本文件时,也是一个字节一个字节地读取以形成字节序列。因此,在存储数据时,需要将字符串类型的数据转换成二进制字节流进行保存。在读取数据时,则需将二进制字节流转换为字符串再显示出来。

2. 文件内部存储案例

1）案例分析

（1）界面分析。布局界面中有三个控件：TextView 控件用来显示评论人，EditText 控件用来输入评论内容，Button 控件用来提交评论，提交后评论内容以文件方式保存到内存中。

（2）设计思路。布局界面中的控件通过添加属性的方式达到用户需求，在按钮单击事件中，调用文件内部存储的方法，将文件保存到内存中，下次运行程序时，读取内存中的文件内容并显示出来，用户可输入新的评论内容并提交，达到用户的需求。

2）实现步骤

（1）创建一个新的工程，工程名为 ZSFileSave。

（2）切换工程的 Project 项目结构。选择该模式下方的 app，依次展开，便可看到工程的布局界面和工程的类文件，其中，activity_main. xml 是布局界面，MainActivity. java 为类文件。

（3）准备一张图片，图片名为 filesavebg. jpg，将其粘贴到 app 目录结构中 res 下方的 drawable 文件夹下，图片作为评论页面的背景图片。

（4）修改布局界面。在 app 目录下的结构中，双击 layout 文件夹下的 activity_main. xml 文件，便可打开布局编辑器，切换到 Text 选项卡，输入代码如下。

```
<?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:background = "@drawable/filesavebg"
    tools:context = ".MainActivity">
    < TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "贝壳评论:"
        android:textSize = "40sp"
        android:layout_marginBottom = "70dp"
        android:layout_marginTop = "20dp"/>
    < EditText
        android:id = "@ + id/editText_input1"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:layout_margin = "5dp"
        android:layout_weight = "1"
        android:background = "#A4E49C"
        android:gravity = "top"
        android:hint = "请在此留下你的内容"
        android:textSize = "20sp"
        android:textColor = "#000000"          />
```

```
    <Button
        android:id="@+id/button_sava1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="提交评论"
        android:textSize="30sp"
        android:textStyle="bold"
        android:background="#009688"
        android:textColor="#ffffff"/>
</LinearLayout>
```

上述代码中，根节点为线性布局，方向为 vertical，其内部放置了一个 TextView 控件用来显示发表评论的用户名，使用 EditText 控件输入评论内容，最下方是 Button 控件，用来实现文件存储功能。

(5) 实现提交评论内容的功能。打开 MainActivity.java 类文件，修改 MainActivity 的代码如下。

```
public class MainActivity extends AppCompatActivity {
    //第一步:定义对象
    private EditText edit_input;
    private Button btn_save;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
       //绑定控件
        initView();
        //按钮单击事件,把输入的内容保存到文件里面
        onbuttonClick();
        //下次打开页面时,读取文件内容,显示出来
        readFile();
        //完善工程,自动显示当前时间
        displaytime();
    }
    //第二步:绑定控件
    private void initView() {
        edit_input = findViewById(R.id.editText_input1);
        btn_save = findViewById(R.id.button_sava1);
      }
    //第三步:按钮单击事件
    private void onbuttonClick() {
        btn_save.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //把输入的内容保存到文件里面
                try {
                    //Context 类中提供了一个方法 openFileOutput 可以实现向文件中写入数据
                    //参数:("文件名", 文件的操作模式:覆盖/追加),路径:data/data/包名/files/
```

```
                    FileOutputStream fout = openFileOutput("zs.txt",0);
                    fout.write(edit_input.getText().toString().getBytes());
                                                                    //数据写入文件中
                    fout.close(); //关闭文件输出流
Toast.makeText(MainActivity.this,"提交成功",Toast.LENGTH_SHORT).show();
                                                                    //弹出提示,提交成功
                } catch (FileNotFoundException e) {
                    e.printStackTrace();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        });
    }
    //第四步:下次打开页面时,读取文件内容,显示出来
    private void readFile() {
        try {
            //调用 Context 类的 openFileInput 方法得到一个文件输入流对象
            FileInputStream fin = openFileInput("zs.txt");
            byte[] arr = new byte[fin.available()];
                                        //创建一个数组缓冲区,用来存放读取的很多字节数据
            fin.read(arr); //数据的读取
            fin.close(); //关闭输入流
            edit_input.setText(new String(arr)); //数据显示出来,byte类型的数据转为字符串
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    //第五步:完善工程:自动显示当前时间
    private void displaytime() {
        Time time = new Time(); //创建一个时间类,注意导入包 import android.text.format.Time
        time.setToNow(); //获取系统当前时间
        edit_input.append("\n\n" + time.year + "年" + (time.month + 1) + "月" + time.monthDay
+ "日" + "\n");//把获取的年月日显示出来
        edit_input.setSelection(edit_input.getText().length());  //将光标移动至文字末尾
    }
}
```

上述代码中,考虑到程序越来越复杂,onCreate()方法中的代码量会越来越多,为了避免代码混乱,根据程序实现的功能自定义几个方法,这种方式不仅可以让 onCreate()内部的代码结构清晰,而且独立的方法有利于程序的复用。方法命名规范一般由两个或多个单词组成,从第二个单词开始首字母大写。

从功能实现角度,在按钮单击事件中,通过 openFileOutput 文件输出流来保存数据到 zs.txt 文件中,由于从 EditText 控件获取用户输入的内容为字符串类型,而所有文件的存储都是采用字节存储的,因此采用 getBytes()方法将字符串转换为二进制字节流,再调用 write()方法将一个一个字节写入文件进行存储。下次运行程序,用户希望原来存储的数据

可以直接显示，此时需要调用 openFileInput 文件输入流从 zs.txt 文件中读取内容。首先建立数据存储空间即字节类型的数组 arr，然后调用 read()方法开始读取数据，并将读取的数据存放到数组 arr 中，最后将数组中的字节流转换为字符串并显示在 EditText 控件上。为了提升用户体验，在重新运行 App 时，自动显示系统时间。

(6) 运行程序，效果如图 5.3 所示。

(7) 查看保存到内存中的评论内容。在模拟器中打开 RE 文件管理器，挂载为可读写模式，根据文件的存储路径依次操作：data/data/< package name >/files/文件名，单击 data，在打开的模拟器页面中再次单击 data，在列表中找到工程的包名 com.example.zsfilesave，在打开的页面中显示了刚刚保存的文件 zs.txt，如图 5.4 所示。双击 zs.txt 将其打开，看到了保存的评论内容，如图 5.5 所示。

图 5.3 文件内部存储运行效果图

图 5.4 查看内部存储文件的保存位置

图 5.5 查看文件内部存储中保存的信息


微课视频

5.2.2 外部存储

手机的内部存储通常不会很大，一旦手机的内部存储容量被用完，可能会出现手机无法使用的情形。对于开发者来说，不宜存储视频等大文件，而适合存储一些小文件。例如，常用的 SharedPreferences 和 SQLite 数据库都是存储在内部存储中的，不会占用太大的空间。

1. 外部存储简介

外部存储是指将文件存储到一些外围设备上，如 SD 卡或者设备内嵌的存储卡，属于永久性的存储方式。外部存储的文件可以被其他应用程序所共享，当将外围存储设备连接到计算机时，这些文件可以被浏览、修改和删除，因此这种方式不安全。

由于外围存储设备可能被移除、丢失或者处于其他状态，因此在使用外围设备之前，建议用户使用 Environment.getExternalStorageState()方法来确认外围设备是否可用，当外围设备可用并且具有读写权限时，就可以通过 FileInputStream、FileOutputStream 或者 FileReader、FileWrite 对象来读写外围设备中的文件。

2. 外部存储调用的方法

访问外部存储的方法如表 5.4 所示。

表 5.4 获取外部存储的方法

获取路径的方法	功能描述	实例和存储路径
getExternalStorageDirectory()	主要的外部存储目录/mnt/sdcar 或者/storage/emulated/0 或者/storage/sdcard0	String path=Environment.getExternalStorageDirectory()+ "/"+"zs.txt"; /storage/emulated/0/zs.txt(存储路径)
context.getExternalCacheDir()	应用在外部存储上的缓存目录	Stringpath = MainActivity.this.getExternalCacheDir()+ "/"+"zs.txt"; /storage/emulated/0/Android/data/com. example.filesavetest/cache/zs.txt(存储路径)
context.getExternalFilesDir(Environment.DIRECTORY_MOVIES)	应用在外部存储上的目录，其中，movies 可更换为任意目录	Stringpath = MainActivity.this.getExternalFilesDir(Environment.DIRECTORY_MOVIES)+ "/"+"zs.txt"; /storage/emulated/0/Android/data/com. example.filesavetest/files/Movies/ zs.txt（存储路径，文件夹可改为 MUSIC、DCIM、PICTURES 等）
context.getFilesDir()	获取当前程序路径应用在内存上的目录	Stringpath = MainActivity.this.getFilesDir()+ "/"+"zs.txt"; /data/data/com.example.filesavetest/files/zs.txt(存储路径)
context.getCacheDir()	应用在内存上的缓存目录	Stringpath = MainActivity.this.getExternalCacheDir()+ "/"+"zs.txt"; /storage/emulated/0/Android/data/com. example.filesavetest/cache/zs.txt(存储路径)

在表 5.4 中，前三行列举了获取外部存储路径调用的方法，后两行列举了获取当前程序

路径应用在内存上目录调用的方法。在 Android 开发中，用户可根据程序需求自行将文件存放到需要的位置。

3. 文件外部存储案例

1）案例分析

（1）界面分析。布局界面中有四个控件，TextView 控件用来显示标题，EditText 控件用来输入内容，两个 Button 控件用来保存内容和删除内容，保存后内容以文件方式保存到外存 SD 卡中。

（2）设计思路。布局界面中的控件通过添加属性的方式达到用户需求，在“保存”按钮单击事件内部，调用文件外部存储的方法，将文件保存到 SD 卡中，下次运行程序时，读取 SD 卡中的文件内容并显示出来。单击“删除”按钮，删除文件内容，达到用户的需求。

2）实现步骤

（1）创建一个新的工程，工程名为 ZSFileSaveTest。

（2）切换工程的 Project 项目结构。选择该模式下方的 app，依次展开，便看到工程的布局界面和工程的类文件，其中，activity_main.xml 是布局界面，MainActivity.java 为类文件。

（3）准备一张图片，图片名为 filesavebgt.jpg，将其粘贴到 app 目录结构中 res 下方的 drawable 文件夹下，图片作为页面的背景图片。

（4）修改布局界面。在 app 目录下的结构中，双击 layout 文件夹下的 activity_main.xml 文件，便可打开布局编辑器，切换到 Text 选项卡，输入代码如下。

```
<?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:background = "@drawable/filesavebgt"
    tools:context = ".MainActivity">
   < TextView
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:text = "随手记事"
        android:textSize = "30sp"
       android:textColor = "#000000"
       android:textStyle = "bold"
       android:gravity = "center"
        android:layout_marginBottom = "10dp"
        android:layout_marginTop = "80dp"/>
< View
    android:layout_width = "match_parent"
    android:layout_height = "5dp"
    android:background = "#000000"/>
    < EditText
        android:id = "@ + id/editText_input1"
```

```
            android:layout_width = "match_parent"
            android:layout_height = "wrap_content"
            android:layout_margin = "5dp"
            android:layout_weight = "1"
            android:background = "#DF9CE4"
            android:gravity = "top"
            android:hint = "请在此留下你的内容"
            android:textSize = "18sp"
            android:textColor = "#000000"          />
        <View
            android:layout_width = "match_parent"
            android:layout_height = "8dp"
            android:background = "#000000"/>
        <LinearLayout
            android:layout_width = "match_parent"
            android:layout_height = "wrap_content"
            android:orientation = "horizontal"
            android:layout_marginTop = "3dp">
            <Button
                android:id = "@ + id/button_sava1"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:text = "保存"
                android:textSize = "30sp"
                android:textStyle = "bold"
                android:background = "#FF5722"
                android:textColor = "#ffffff"
                android:layout_weight = "1"/>
            <Button
                android:id = "@ + id/button_delete"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:text = "删除"
                android:textSize = "30sp"
                android:textStyle = "bold"
                android:background = "#E91E63"
                android:textColor = "#ffffff"
                android:layout_weight = "1"
                android:layout_marginBottom = "2dp"/>
        </LinearLayout>
    </LinearLayout>
```

(5) 实现外部文件存储与读取功能。重点关注向外围设备(SD 卡)中存储数据和从外围设备(SD 卡)中读取数据的示例代码,具体如下。

```
public class MainActivity extends AppCompatActivity {
    private static final String TAG = "MainActivity";
    //第一步:定义对象
   private EditText edit_input1;
   private Button btn_save1;
```

```
    private String path;
    private File file;
    private String filename = "note.txt";
     @Override
     protected void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.activity_main );
        //绑定控件
         initView();
         //按钮单击事件,把输入的内容保存到文件里面
         onbuttonClick();
         //下次打开页面时,读取文件内容,显示出来
         readFile();
         //完善工程,自动显示当前时间
         displaytime();
     }
     //第二步:绑定控件
     private void initView() {
         edit_input1 = findViewById(R.id.editText_input1);
         btn_save1 = findViewById(R.id.button_sava1);
       }
     //第三步:按钮单击事件
     private void onbuttonClick() {
         btn_save1.setOnClickListener(new View.OnClickListener() {
             @Override
             public void onClick(View view) {
                 //把输入的内容保存到文件里面
                 try {
            //利用外部 getExternalStorageDirectory 获取根目录,直接在后面加上想创建的文件
    path = Environment.getExternalStorageDirectory() + "/" + filename; //路径是:/storage/
emulated/0/ note.txt
     file = new File(path); //创建文件
         if(!file.exists()){ //文件若不存在则调用 file.createNewFile()
                 file.createNewFile();
         }
     FileOutputStream fout = openFileOutput(filename,0); //向文件中写内容,自然要创建文件
                                                         //输出流的操作
     fout.write(edit_input1.getText().toString().getBytes()); //数据写入文件中,调用输出
                                                              //流的 write 方法
     fout.close(); //关闭文件输出流
     Toast.makeText (MainActivity.this,"提交成功",Toast.LENGTH_SHORT).show(); //弹出提示,
                                                                            //提交成功
                 } catch (FileNotFoundException e) {
                     e.printStackTrace();
                 } catch (IOException e) {
                     e.printStackTrace();
                 }
             }
         });
```

```
    btn_delete.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    deleteFile(filename); //删除文件
     Toast.makeText(MainActivity.this,"删除成功",Toast.LENGTH_SHORT).show(); //弹出提示,
                                                                              //删除成功
                }
        });
       }
    //第四步:下次打开页面时,读取文件内容,显示出来
    private void readFile() {
        //检查 SD 卡的状态,Environment.MEDIA_MOUNTED 表示 SD 卡在手机上正常使用状态
        if(Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED)){
            //SD 卡可用,可以开始读取数据
            try {
FileInputStream fin = openFileInput(filename); //调用 Context 类 openFileInput 方法得到一个
                                               //文件输入流对象
byte[] arr = new byte[fin.available()]; //创建一个数组缓冲区,用来存放读取的数据
                fin.read(arr); //数据的读取
                fin.close(); //关闭输入流
                edit_input1.setText(new String(arr));//数据显示出来,byte 类型的数据转
                                                     //为字符串
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }else{
            //SD 卡不可用
           Toast.makeText(MainActivity.this,"SD 卡不可用",Toast.LENGTH_SHORT).show();
        }
    }
     //第五步:完善工程:自动显示当前时间
    private void displaytime() {
        Time time = new Time(); //创建一个时间类,导入包 import android.text.format.Time;
        time.setToNow(); //获取系统当前时间
        edit_input1.append("\n\n" + time.year + "年" + (time.month + 1) + "月" + time.monthDay
+ "日" + "\n");//把获取的年月日显示出来
        edit_input1.setSelection(edit_input.getText().length()); //将光标移动至文字末尾
    }
}
```

上述代码中,向外围设备(SD 卡)中存储数据时,使用到了 Environment.getExternalStorageDirectory()方法,该方法用于获取 SD 卡的路径。针对不同手机厂商 SD 卡根目录也不完全一致,用这种方法可以避免把路径写死而找不到 SD 卡。在该路径下创建文件,运行时文件若不存在则调用 file.createNewFile()方法创建文件,之后创建文件输出流对象,调用 write()方法向文件中写内容。

从外围设备(SD 卡)中读取数据时,需要判断外围设备是否可用,Environment.

getExternalStorageState( )方法用来确认外围设备是否可用，Environment. MEDIA _ MOUNTED 表示 SD 卡处于正常使用状态，当外围设备可用并且具有读写权限时，通过 FileInputStream 对象读取指定目录下的文件。

(6) 声明权限。Android 系统为了保证应用程序的安全性做了相应规定，如果程序需要访问系统的一些关键信息，如读写 SD 卡、访问 Internet、打开相机、访问相册、访问通讯录等，必须在工程的配置文件 AndroidManifest. xml 中声明权限才可使用，否则程序运行时会直接崩溃。这里操作 SD 卡的信息就是系统中比较关键的信息，因此需要在清单文件的< manifest >节点下配置权限信息，切换工程的 Project 项目结构。选择该模式下方的 app→src→main→AndroidManifest. xml 依次展开，双击打开 AndroidManifest. xml 配置文件，具体代码如下。

```
<manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    package = "com.example.filesavetest">
    <uses - permission android:name = "android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses - permission android:name = "android.permission.READ_EXTERNAL_STORAGE"/>
    <application
      ...
```

上述代码指定了当前 App 程序具有读写外部 SD 卡的权限，因此应用程序便可以操作 SD 卡中的数据。

(7) 运行程序，效果如图 5.6 所示。

(8) 查看保存到外存中的随手记事内容。在模拟器找到文件管理器，双击打开文件管理器，根据文件的存储路径依次操作：/storage/emulated/0/note. txt 文件名，单击 storage 文件夹，打开的模拟器页面中单击 emulated 文件夹，单击 0 文件夹，在打开的页面中显示了刚刚保存的文件 note. txt，如图 5.7 所示。双击 note. txt 将其打开，便可查看随手记事的文件内容。

图 5.6　文件外部存储运行效果图

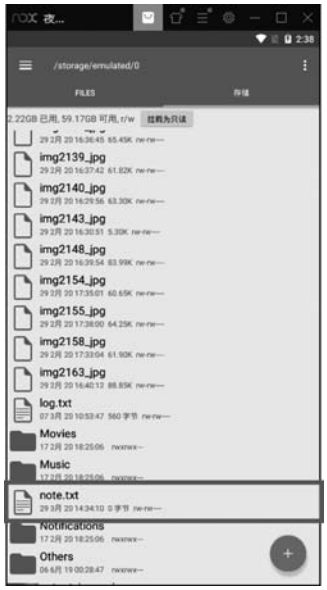

图 5.7　查看外部存储文件的保存位置


微课视频

5.3 调用摄像头和相册

在使用 QQ 或者微信的时候经常要和别人分享图片，这些图片可以是用手机摄像头拍的，也可以是从相册中选取的。类似这样的功能太常见了，那么本节就学习一下调用摄像头和相册方面的知识。


微课视频

5.3.1 调用摄像头拍照

现在很多应用都会要求用户上传一张图片来作为头像，这时打开摄像头拍张照片是最简单快捷的，下面通过一个例子来学习一下，如何才能在应用程序中调用手机的摄像头进行拍照。

1. 案例分析

(1) 界面分析。布局界面中有两个控件，其中，Button 控件用来打开相机拍照，ImageView 控件用来把拍摄的照片显示出来。

(2) 设计思路。布局界面中的控件通过添加属性的方式达到用户需求，单击拍照按钮时利用隐式跳转打开系统相机，并将拍摄的照片存放到指定的路径下，拍照结束后通过系统回调将拍摄的照片显示出来，达到用户的需求。

2. 实现步骤

(1) 创建一个新的工程，工程名为 ZSPhoto。

(2) 切换工程的 Project 项目结构，选择该模式下方的 app，依次展开，便看到工程的布局界面和工程的类文件，其中，activity_main.xml 是布局界面，MainActivity.java 为类文件。

(3) 准备一张图片，图片名为 zsphotobg.jpg，将其粘贴到 app 目录结构中 res 下方的 drawable 文件夹下，作为页面的背景图片。

(4) 设计布局界面。双击 layout 文件夹下的 activity_main.xml 文件，便可打开布局编辑器，修改布局类型为 LinearLayout，添加方向属性为 vertical。依次添加两个控件，代码如下。

```
<?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:background = "@drawable/zsphotobg"
    tools:context = ".MainActivity">
    < ImageView
        android:id = "@ + id/imageView_cam"
        android:layout_width = "match_parent"
        android:layout_height = "200dp"
```

```
        android:src = "@mipmap/ic_launcher_round"
        android:layout_gravity = "center"          />
    <Button
        android:id = "@ + id/button_cam"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:textColor = "#000000"
        android:text = "拍照"
        android:textSize = "30sp"/>
</LinearLayout
```

上述代码中，Button 控件用来打开系统相机，ImageView 控件用来显示拍摄的图片。

(5) 按钮单击事件处理。打开 MainActivity.java 类文件，修改 MainActivity 的代码，具体如下。

```
public class MainActivity extends AppCompatActivity {
private static final String TAG = "MainActivity";
//定义对象
private ImageView img_camer;
private Button btn_camer;
//定义一个路径：①拍照的临时路径；②显示照片的最终路径
String tmp_path,disp_path;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main );
    //绑定控件
     initView();
   //"拍照"按钮单击事件
    btnCamOnclick();
    //拍完照之后,接收照片并显示,系统回调
  }
//第一步:绑定控件
private void initView() {
    img_camer = findViewById(R.id.imageView_cam);
    btn_camer = findViewById(R.id.button_cam);
 }
//第二步:拍照按钮单击事件
< private void btnCamOnclick() {
    btn_camer.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Log.d(TAG, "onClick:1 " + "单击了按钮");>
            //拍照代码
            //前期准备:路径
               < tmp_path = Environment.getExternalStorageDirectory() + "/img_" +
randFileName() + ".jpg";
            File imgfile = new File(tmp_path);>//创建保存照片的文件,照片都保存到本路径下
```

```
                try {
                    if(imgfile.exists()){
                        imgfile.delete();
                    }
                    imgfile.createNewFile();
                } catch (IOException e) {
                    e.printStackTrace();
                }
    Intent intent = new Intent("android.media.action.IMAGE_CAPTURE");
                                                        //1.创建一个打开相机的 Intent
    intent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(imgfile));
                                                        //2.告诉相机图片的保存位置
    startActivityForResult(intent,11); //3.打开相机
            }
        });
    }
  }
    //第三步:拍完照之后,接收照片并显示,系统回调
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        switch (requestCode){
            case 11:
                if (resultCode == RESULT_OK){
                    //根据照片的真实路径来显示照片
                    disp_path = tmp_path; //确定图片的路径
                    Glide.with(MainActivity.this).load(disp_path).into(img_camer);
                                                                    //显示照片
                }
                break;
            default:
                break;
        }
    }
//时间戳,为了保证每次拍的照片都能以不同的文件名保存到 SD 卡上
    private String randFileName(){
        Time t = new Time();
        t.setToNow(); //取得系统时间
        String strtime = t.year + "" + (t.month + 1) + "" + t.monthDay + "" + t.hour + "" + t.
minute + "" + t.second + "";
        return strtime;
    }
}
```

上述代码中,在 onCreate()方法内部自定义两个方法:initView()方法内部放绑定控件的代码,btnCamOnclick()方法内部放按钮单击事件的代码。这种方式不仅可以让 onCreate()内部的代码结构清晰,而且独立的方法有利于程序的复用。

在按钮单击事件内部书写拍照代码,拍照之前先准备好照片的存放路径,调用 Environment.getExternalStorageDirectory()方法将拍摄的照片存放到外存的 SD 卡上。

为了保证拍摄的照片能够全部保存到 SD 卡上，采用时间戳为图片文件名的一部分，这样可以确保文件名的唯一。路径确定好之后，调用 createNewFile()方法在该路径下创建保存照片的文件，一切前期工作准备就绪，就可以通过隐式跳转的方式打开系统相机，同时告诉相机拍完之后图片的保存位置，通过 startActivityForResult(intent,11)启动活动，方法中传递了期望在活动销毁时能够返回一个结果给上一个活动的请求码 11，用于在之后的回调中判断数据的来源。因此，当相机拍照结束以后，会回调上一个活动的 onActivityResult()方法，因此需要在 MainActivity 类文件中重写这个方法来得到返回的数据。

onActivityResult()方法中有三个参数：第一个参数 requestCode 是在启动相机活动时传入的请求码，第二个参数 resultCode 是返回数据时传入的处理结果，第三个参数 data 是携带者返回数据的 Intent。首先通过检查 requestCode 判断数据的来源，再通过 resultCode 的值判断处理结果是否成功，最后通过 Glide 图片加载库根据图片的路径将图片显示到 ImageView 控件中。

(6) Glide 图片加载库。Glide 是一个被 Google 所推荐的图片加载库，作者是 bumptech。这个库被广泛运用在 Google 的开源项目中。Glide 是滑行的意思，可以看出这个库的主旨就在于让图片加载变得流畅。使用 Glide 时需要在 Android Studio 上添加依赖，切换工程的 Project 项目结构。选择该模式下方的 app→src→ build.gradle 依次展开，双击打开 build.gradle 文件，添加 Glide 依赖，只需添加一行代码即可，代码如下。

```
apply plugin: 'com.android.application'
…
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    …
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
    implementation 'com.github.bumptech.glide:glide:4.9.0'
}
```

(7) 声明权限。因为要访问系统相机，因此需要在清单文件的< manifest >节点下配置权限信息，切换工程的 Project 项目结构。选择该模式下方的 app → src → main → AndroidManifest.xml 依次展开，双击打开 AndroidManifest.xml 配置文件，具体代码如下。

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.filesavetest">
        <uses-permission android:name="android.permission.CAMERA"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>

    <application
    …
```

上述代码指定了当前 SD 卡具有可写可读的权限，同时指定了本程序具有访问系统相机的权限，因此应用程序不仅可以打开系统相机拍照，而且还可以操作 SD 卡中的图片

文件。

（8）运行程序，效果如图 5.8 所示。

图 5.8　打开摄像头拍照和拍照的最终效果


微课视频

5.3.2　从相册中选择照片

虽然调用摄像头拍照既方便又快捷，但并不是每次都需要去当场拍照，因为每个人的手机相册里应该会有很多图片，直接从相册里选择一张现有的照片会比打开相机拍摄一张照片更加常用。一个优秀的应用程序应该将这两种选择方式都提供给用户，由用户决定使用哪一种。下面就来看一下如何才能实现从相册中选择照片。

1. 案例分析

（1）界面分析。布局界面中有两个控件，其中，Button 控件用来打开相册，ImageView 控件用来把选择的照片显示出来。

（2）设计思路。布局界面中的控件通过添加属性的方式达到用户需求，单击 Button 打开相册时利用隐式跳转打开手机相册，照片选择结束后通过系统回调将选择照片显示出来，达到用户的需求。

2. 实现步骤

（1）无须创建新工程，在拍照案例工程 ZSPhoto 中继续完善程序即可。

（2）完善设计布局界面。双击 layout 文件夹下的 activity_main.xml 文件，便可打开布局编辑器，在布局界面上拍照案例的两个控件保持不动，继续添加两个控件，代码如下。

```
<?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:background = "@drawable/zsphotobg"
    tools:context = ".MainActivity">

    …

    < Button
        android:id = "@ + id/button_photo"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:textColor = "#000000"
        android:text = "从相册选择"
        android:textSize = "30sp" />
    < ImageView
        android:id = "@ + id/imageView_photo"
        android:layout_width = "match_parent"
        android:layout_height = "200dp"
        android:src = "@drawable/ic_launcher_background"
        android:layout_gravity = "center"          />
</LinearLayout >
```

上述代码中，Button 控件用来打开系统相册，ImageView 控件用来显示选择的图片。

（3）“选择相册”按钮功能。打开 MainActivity.java 类文件，完善从相册中选择图片的功能，代码如下。

```
public class MainActivity extends AppCompatActivity {
    private static final String TAG = "MainActivity";
    //定义对象
    private ImageView img_camer,img_photo;
    private Button btn_camer,btn_photo;
    //定义一个路径：①拍照的临时路径；②显示照片的最终路径
    String tmp_path,disp_path;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //手动开启程序权限之后,拍照仍闪退,解决拍照闪退问题
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
         StrictMode.VmPolicy.Builder builder = new StrictMode.VmPolicy.Builder();
         StrictMode.setVmPolicy(builder.build());
        }
```

```
        //绑定控件
        initView();
        //拍照按钮单击事件
        btnCamOnclick();
//拍完照之后,接收照片并显示,系统回调
        //"打开相册"按钮单击事件
        btnPhoOnclick();
    }
    //第一步:绑定控件
    private void initView() {
        img_camer = findViewById(R.id.imageView_cam );
        btn_camer = findViewById(R.id.button_cam );
        img_photo = findViewById(R.id.imageView_photo );
        btn_photo = findViewById(R.id.button_photo );
    }
    //第二步:"拍照"按钮单击事件
    private void btnCamOnclick() {
        btn_camer.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
             //打开相机拍照按钮,拍照案例已有,这里代码省略
                ...
                }
        });
    }
    //第四步:"打开相册"按钮单击事件
    private void btnPhoOnclick() {
        btn_photo.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //打开相册
           Intent intent = new Intent("android.intent.action.GET_CONTENT");
                                                        //创建打开相册的 Intent
           intent.setType("image/*"); //打开页面的类型
           startActivityForResult(intent,22); //打开相册
             }
        });
    }
    //第三步:拍完照之后,接收照片并显示,系统回调
    @Override
     protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent
data) {
        switch (requestCode){
            case 11:
                if (resultCode == RESULT_OK){
                    //根据照片的真实路径来显示照片
                    disp_path = tmp_path; //确定图片的路径
                    Glide.with(MainActivity.this).load(disp_path).into(img_camer);
                                                                    //显示照片
```

```
                }
                break;
            case 22:
                if(resultCode == RESULT_OK){
                    //根据选择照片的路径,显示照片
                    Uri imageuri = data.getData(); //这是一个缩略图
                    if (imageuri == null){
                        return;
                    }
                    disp_path =  UriUtils.uri2File (imageuri).getPath();
                                                        //将缩略图转换为真实的图片路径
                    Glide.with (MainActivity.this).load( disp_path ).into(img_photo);
                }
                break;
            default:
                break;
        }
    }
    //时间戳,为了保证每次拍的照片都能以不同的文件名保存到 SD 卡上
    private String randFileName(){
        Time t = new Time();
        t.setToNow(); //取得系统时间
        String strtime = t.year + "" + (t.month + 1) + "" + t.monthDay + "" + t.hour + "" + t.
minute + "" + t.second + "";
        return strtime;
    }
}
```

上述代码中,利用隐式跳转 android.intent.action.GET_CONTENT 打开系统页面,调用 setType("image/*")方法设定打开页面的类型为图片,系统回调中,通过返回的数据 data.getData()得到图片的缩略图路径,之后调用 UriUtils.uri2File(imageuri).getPath()方法将缩略图转换为真实的图片路径,最后通过 Glide 图片加载库根据图片的真实路径将图片显示到 ImageView 控件中。

(4) 将图片的缩略图路径转换为真实路径的方法。UriUtils 是一个依赖,需要在 Android Studio 上添加依赖,切换工程的 Project 项目结构,选择该模式下方的 app→src→build.gradle 依次展开,双击打开 build.gradle 文件,添加 UriUtils 依赖,代码如下。

```
dependencies {
implementation fileTree(dir: 'libs', include: ['*.jar'])
…
implementation 'com.github.bumptech.glide:glide:4.9.0'
api 'com.blankj:utilcode:1.23.7'
}
```

(5) 运行程序,效果如图 5.9 所示。

图 5.9　打开手机相册和选择照片的最终效果

5.4　Android 运行时权限设置

Android 的权限机制并不是什么新鲜事物，从系统的第一个版本开始就已经存在了，但其实之前 Android 的权限机制在保护用户安全和隐私等方面起到的作用非常有限，为了更好地保护用户的安全和隐私，安卓系统的权限管理机制从 API 23 即 Android 6.0 之后发生了比较大的改变，在一些比较危险的权限上要求必须申请动态权限。那么本节就来详细学习一下 Android 6.0 系统中引入的新特性。

5.4.1　Android 权限机制介绍

1. 动态权限需求原因

Android 6.0 之前，权限在应用安装过程中只询问一次，以列表的形式展现给用户，然而大多数用户并不会注意到这些，直接就下一步了，应用安装成功后就会被赋予清单文件中的所有权限，应用就可以在用户不知情的情况下进行非法操作，如偷偷地上传用户数据等。其次，有些常用软件普遍存在着滥用权限的情况，如微信所申请的权限列表：存储、您的位置、电话、相机、短信、通讯录、身体传感器、麦克风等，其中的短信权限与微信关系不大，但用户若不认可微信所申请的所有权限，那么微信则会安装失败。

Android 开发团队也意识到了这个问题，于是在 Android 6.0 系统中加入了运行时动态权限申请，也就是说，用户不需要在安装软件的时候一次性授权所有申请的权限，而是可以在软件使用的过程中再对某一项权限申请进行授权。例如，一款相机应用在运行时申请了

地理位置定位权限，即使用户拒绝了这个权限，仍然可以使用这个应用的其他功能，而不是像之前那样直接无法安装软件。

2. 需要动态申请的权限

并不是所有权限都需要在运行时申请，对于用户来说，不停地授权也很烦琐。Android现在将所有的权限归成了两类：一类是普通权限，一类是危险权限。普通权限指的是那些不会直接威胁到用户的安全和隐私的权限，对于这部分权限申请系统会自动授权，无须用户手动操作，例如，查看网络连接权限和开机启动权限。危险权限则是表示那些可能会触及用户隐私或者对设备安全性造成影响的权限，如获取设备联系人信息、定位设备的地址位置等，对于这部分权限申请，必须要由用户手动授权才可以，否则程序就无法使用相应的功能。

但是 Android 中一共有上百种权限，怎么区分哪些是普通权限、哪些是危险权限呢？其实并不难，因为危险权限总共就那么几个，除了危险权限之外，剩余的都是普通权限。Android 中所有的危险权限如表 5.5 所示。

表 5.5 需要动态申请的权限

权 限 名	功能描述	实 例
WRITE_EXTERNAL_STORAGE READ_EXTERNAL_STORAGE	外部存储权限	< uses-permission android: name = "android.permission. WRITE_EXTERNAL_STORAGE"/>
CAMERA	调用相机权限	
RECORD_AUDIO	录音权限	< uses-permission android: name = "android.permission. RECORD_AUDIO"/>
BODY_SENSORS	传感器权限	< uses-permission android: name = "android.permission. BODY_SENSORS"/>
WRITE_CALENDAR READ_CALENDAR	读写日历权限	< uses-permission android: name = "android.permission. WRITE_CALENDAR"/>
READ_CONTACTS WRITE_CONTACTS GET_ACCOUNTS	通讯录权限	< uses-permission android: name = "android.permission. READ_CONTACTS" />
SEND_SMS RECEIVE_SMS READ_SMS RECEIVE_WAP_PUSH RECEIVE_MMS USE _ SIP PROCESS _ OUTGOING _CALLS	消息权限	< uses-permission android: name = "android.permission. SEND_SMS"/>
CALL_PHONE READ_PHONE_STATE READ_CALL_LOG WRITE_CALL_LOG ADD_VOICEMAIL	手机状态相关	< uses-permission android: name = "android.permission. CALL_PHONE" />
ACCESS_FINE_LOCATION ACCESS_COARSE_LOCATION	定位权限	< uses-permission android: name = "android.permission. ACCESS_FINE_LOCATION"/>

表5.5中的权限共9大类，24个权限，使用时从表中查看即可，如果是属于这张表中的权限，不仅需要在AndroidManifest.xml文件中添加权限声明，而且还要在程序代码中进行权限动态申请，如果所需的权限不在这张表中，那么只需要在AndroidManifest.xml文件中添加权限声明就可以了。

5.4.2 在程序运行时申请权限

1. 判断Android系统版本

在官方文档中，可以看到低于API23是不需要使用动态权限申请的，如果是Android 6.0以上的系统，需要进行判断，代码如下。

```
if (Build.VERSION.SDK_INT >= 23) {
            //此处做动态权限申请
    }
else {
            //低于23不需要特殊处理
    }
}
```

2. 运行时申请权限案例

1）案例分析

（1）界面分析。布局界面中有两个控件，其中，TextView控件用来显示一段文字，Button控件用来打电话。

（2）设计思路。布局界面中的控件通过添加属性的方式达到用户需求，单击Button时，弹出“需要使用电话权限，是否允许?”的提示，如果禁止则程序关闭，如果允许则开始拨打电话，达到用户的需求。

2）实现步骤

（1）创建一个新的工程，工程名为ZSRuntimePermission。

（2）切换工程的Project项目结构，选择该模式下方的app，依次展开，便看到工程的布局界面和工程的类文件，其中，activity_main.xml是布局界面，MainActivity.java为类文件。

（3）准备一张图片，图片名为runtimepbg.jpg，将其粘贴到app目录结构中res下方的drawable文件夹下，作为页面的背景图片。

（4）修改布局界面。在app目录下的结构中，双击layout文件夹下的activity_main.xml文件，便可打开布局编辑器，切换到Text选项卡，输入如下代码。

```
<?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:background = "@drawable/runtimepbg"
    tools:context = ".MainActivity">
    <TextView
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:text = "动态申请权限"
        android:textStyle = "bold"
        android:textSize = "30sp"
        android:textColor = "#0000ff"
        android:gravity = "center"
        android:layout_marginTop = "100dp"/>
    <Button
        android:id = "@ + id/button_dadianhua"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:text = "拨打电话"
        android:textStyle = "bold"
        android:textSize = "40sp"
        android:textColor = "#ff0000"
        android:gravity = "center"
        android:layout_marginTop = "50dp"/>
</LinearLayout>
```

上述代码中,根节点为线性布局,方向为 vertical,其内部放置了一个 TextView 控件用来显示提示文字,下方是 Button 控件,单击按钮可拨打电话。

(5) 实现打电话功能。打开 MainActivity.java 类文件,修改 MainActivity 的代码,如下。

```
public class MainActivity extends AppCompatActivity {
    //定义对象
    private Button btn_dadianhua;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //绑定控件
        initView();
        //按钮单击事件
        btnOnClick();
    }
    //绑定控件
    private void initView() {
        btn_dadianhua = findViewById(R.id.button_dadianhua);
    }
    //按钮单击事件
    private void btnOnClick() {
        btn_dadianhua.setOnClickListener(new View.OnClickListener() {
            @Override
```

```
            public void onClick(View view) {
//检查权限:判断 Android 版本是否大于 23
//参数:上下文 Context 和权限的名称.PERMISSION_GRANTED 表示存在权限
                if (Build.VERSION.SDK_INT > 23) {
if (ContextCompat.checkSelfPermission(MainActivity.this, Manifest.permission.CALL_PHONE)!=
PackageManager.PERMISSION_GRANTED ) {
        //申请权限
ActivityCompat.requestPermissions(MainActivity.this, new String[]{Manifest.permission.CALL
_PHONE },11);
                        }else {
                          //权限同意,不需要处理,调用打电话的方法
                          callPhone();
                        }
                } else {
                   //低于 23,不需要特殊处理
                    callPhone();
                }
            }
        });
    }
    /** 注册权限申请回调
     * @param requestCode 申请码:在申请权限的时候使用的唯一的申请码
     * @param permissions 申请的权限:String[] permission 则是权限列表,一般用不到
     * @param grantResults 结果:int[] grantResults 是用户的操作响应,包含该权限则请求成功
     * 由于在权限申请的时候,我们就申请了一个权限,所以此处的数组的长度都是 1
     */
    @Override
    public void onRequestPermissionsResult(int requestCode, String[] permissions, int[]
grantResults)
    {
        switch (requestCode) //当权限较多时,建议使用 Switch
        {
            case 11:
                if (grantResults[0] == PackageManager.PERMISSION_GRANTED ){
                    //权限申请成功
                    callPhone();
                }else {
                    //权限申请失败
Toast.makeText (MainActivity.this, "权限申请失败,用户拒绝权限", Toast.LENGTH_SHORT ).show();
                }
                break;
            default:
                 break;
        }
    }
    /**
     * 拨号方法
```

```
        */
        private void callPhone()
        {
            Intent intent = new Intent();
            intent.setAction(Intent.ACTION_CALL);
            intent.setData(Uri.parse("tel:10086"));
            startActivity(intent);
        }
    }
```

以上代码中，低于 API23 是不需要使用动态权限申请的，所以首先检测 Android 系统版本，如果版本高于 23，使用 ContextCompat.CheckSlefPermission 方法检查一下有没有权限，该方法有两个参数：上下文 Context 和权限的名称。返回值 PERMISSION_GRANTED 表示存在权限，PERMISSION_DENIED 表示不存在权限，使用方法的返回值和 PackageManager.PERMISSION_GRANTED 做比较，相等就说明用户已经授权，不等就表示用户没有授权。

如果已授权，直接去执行拨打电话的逻辑操作就可以了。这里把拨打电话的逻辑封装到了 callPhone()方法当中。如果没有授权，则需要调用 ActivityCompat.requestPermissions()方法向用户授权。

requestPermissions 方法包含三个参数：参数 1 是当前上下文；参数 2 是一个权限数组，把权限名放到数组中即可；参数 3 是请求码。使用权限数组，说明可以一次申请多个权限。请求码只要是唯一值就可以了，取值要大于 0，小于 65535，这里传入 11。由于这个权限请求是异步操作的，所以用户判断权限后需要回调函数，就用到这个请求码。

调用完 requestPermissions()方法之后，系统会弹出一个权限申请的对话框，用户可以选择同意或拒绝权限申请，不论是哪种结果，最终都会回调到 onRequestPermissionsResult()方法中。

回调函数的处理。由于程序是异步操作，在用户完成了操作后，需要调用回调函数，而此回调函数则是 onRequestPermissionsResult()方法，该方法有 3 个参数：requestCode 是申请权限的时候使用的唯一的申请码；String[] permission 则是权限列表，一般用不到；int[] grantResults 中封装了授权的结果。由于在权限申请的时候，申请了一个权限，所以此处的数组的长度都是 1，这里需要判断一下最后的授权结果，如果用户已同意授权就调用 callPhone()方法拨打电话，如果用户拒绝的话程序放弃操作，并且弹出一条失败的提示。

(6) 声明权限。因为打电话会涉及手机的资费问题，因而被列为危险权限，因此必须在 AndroidManifes.xml 中加入权限声明，否则运行时程序就会崩溃。切换工程的 Project 项目结构。选择该模式下方的 app→src→main→AndroidManifest.xml 依次展开，双击打开 AndroidManifest.xml 配置文件，具体代码如下。

```
<manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    package = "com.example. zsruntimepermission">
```

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

    <application
            …
```

（7）运行程序，效果如图 5.10 所示。

图 5.10　运行时权限申请效果

如果用户在低于 6.0 系统的设备上安装该程序，用户可以在应用管理界面查看任意一个程序的权限申请情况。在 Android 5.0 系统模拟器中查看权限方法为："设置"→"应用管理"→ZSRuntimePermission，这样就可以非常清楚地知晓程序一共申请了哪些权限。同样，采用相同的步骤在 Android 6.0 系统真机上也可以查看该程序的权限，如图 5.11 所示。

图 5.11　Android 5.0 系统和 Android 6.0 系统权限展示

微课视频

5.5 实战案例——记忆的仓库：备忘录

时间如流水，几度夕阳红，回首往事，当迫不及待想记录一些事情的时候，有些人会选择用最原始的笔记本记录，但随着社会的发展，出现了新型的记录方式，经常伴随我们左右的智能手机和笔记本电脑也给用户随手记录提供了便利。本节将利用本章所学习的知识，设计和实现一款私人定制的手机备忘录 App，该备忘录功能简洁实用、界面美观、安全性高、易学易用，让我们一起动手开始私人订制的备忘录之旅吧！

微课视频

5.5.1 界面分析

本案例中包含三个界面：登录界面、主界面和信息添加界面，效果如图 5.12 所示。

图 5.12　备忘录界面效果图

微课视频

5.5.2 实现思路

第一个界面是登录界面，用来保护备忘录信息的安全，采用 SharedPreference 键值对存储技术来保存用户名和密码；第二个界面是主界面，以滚动列表方式显示所有的备忘录信息，采用 RecyclerView 滚动控件和 SQLite 数据库实现；第三个界面是信息添加界面，用来添加新的备忘录内容，输入的标题和内容及选择的心情图片都会保存到 SQLite 数据库中。

微课视频

5.5.3 任务实施

【任务 5-1】 使用 SharedPreferences 保存输入的用户名和密码

(1) 创建一个新的工程，工程名为 MyMemoTest。

(2) 切换工程的 Project 项目结构，选择该模式下方的 app，依次展开，便看到工程的布

局界面和工程的类文件，其中，activity_main. xml 是布局界面，MainActivity. java 为类文件。

（3）在工程中添加新的页面。右击 com. example. zsintent 包→New→Activity→Empty Activity，会弹出一个创建活动的对话框，将活动命名为 LoginActivity，默认勾选 Generate Layout File 关联布局界面，布局界面名称为 activity_login 但不要勾选 Launcher Activity。单击 Finish 按钮，便可在工程中完成第二个页面的添加。同理，添加第三个页面，类文件为 AddInfoActivity，布局界面名称为 activity_add_info。

（4）准备两张图片，图片名为 bgone. png 和 bgthree. jpg，将其粘贴到 app 目录结构中 res 下方的 drawable 文件夹下，作为登录页面和备忘录添加页面的背景图片。

（5）准备三张小图片，将其粘贴到 app 目录结构中 res 下方的 drawable 文件夹下。其中，buttonbg. png 和 savebg. png 两张小图片作为“添加备忘录”按钮和“保存”按钮的背景图片，sunshine. jpg 图片显示到备忘录添加页面的图片控件中。

（6）设计第一个 Activity 的布局界面。双击 layout 文件夹下的 activity_login. xml 文件，便可打开布局编辑器，修改布局类型为 LinearLayout，添加方向属性为 vertical。依次添加控件，代码如下。

```
<?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:background = "@drawable/bgone"
    tools:context = ".LoginActivity">
< TextView
    android:layout_width = "match_parent"
    android:layout_height = "wrap_content"
    android:text = "用户登录"
    android:textStyle = "bold"
    android:textColor = "#000000"
    android:textSize = "40sp"
    android:layout_margin = "100dp"
    android:gravity = "center"/>
    < LinearLayout
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:orientation = "horizontal"
        android:layout_margin = "10dp">
        < TextView
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "用户名:"
            android:textColor = "#000000"
            android:textSize = "30sp"          />
```

```
        <EditText
            android:id="@+id/editText_inputname"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="请输入用户名"
            android:textColor="#000000"
            android:textSize="30sp"          />
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_margin="10dp">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="密 码:"
            android:textColor="#000000"
            android:textSize="30sp"          />
        <EditText
            android:id="@+id/editText_inputpwd"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="请输入密码"
            android:textColor="#000000"
            android:inputType="textPassword"
            android:textSize="30sp"          />
    </LinearLayout>
    <CheckBox
        android:id="@+id/checkBox_reme"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="记住密码"
        android:layout_gravity="right"
        android:layout_margin="10dp"/>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_margin="10dp">
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="取消"
            android:textColor="#000000"
            android:textSize="30sp"
            android:layout_weight="1"/>
        <Button
            android:id="@+id/button_login"
```

```
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "登录"
            android:textColor = "#000000"
            android:textSize = "30sp"
            android:layout_weight = "1"/>
    </LinearLayout>
</LinearLayout>
```

（7）实现第一个 Activity 页面的功能。获取 EditText 控件输入的用户名和密码，采用 getSharedPreferences 键值对方式存储起来，根据复选框的状态进行判断，如果选中“记住密码”复选框则获取保存的用户名和密码并显示，如果没有选中“记住密码”复选框，则清空 EditText 控件的内容。打开 LoginActivity.java 类文件，修改 LoginActivity 的代码如下。

```
public class LoginActivity extends AppCompatActivity {
//定义对象
    private EditText edit_inputname,edit_inputpwd;
    private CheckBox check_reme;
    private Button btn_login;
     @Override
     protected void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.activity_login);
         //1.绑定控件
         initView();
         //2.单击“登录”按钮,将用户名和密码保存起来
         btnloginonClick();
         //3.下次启动,直接显示用户名和密码
         displayinfo();
     }
     //1. 绑定控件
     private void initView() {
         edit_inputname = findViewById(R.id.editText_inputname);
         edit_inputpwd = findViewById(R.id.editText_inputpwd);
         check_reme = findViewById(R.id.checkBox_reme);
         btn_login = findViewById(R.id.button_login);
     }
     //2.单击“登录”按钮,将用户名和密码保存起来
     private void btnloginonClick() {
         btn_login.setOnClickListener(new View.OnClickListener() {
             @Override
             public void onClick(View view) {
                 //保存用户名和密码
              SharedPreferences.Editor editor = getSharedPreferences("myinfo",0).edit();
              editor.putString("name",edit_inputname.getText().toString());
                 editor.putString("pwd",edit_inputpwd.getText().toString());
                 editor.putBoolean("st",check_reme.isChecked());
                 editor.commit();
```

```
                //跳转到第二页
                Intent intent = new Intent(LoginActivity.this,MainActivity.class);
                startActivity(intent);
            }
        });
    }
    //3.下次启动,直接显示用户名和密码
    private void displayinfo() {
        String strname = getSharedPreferences("myinfo",0).getString("name","");
        String strpwd = getSharedPreferences("myinfo",0).getString("pwd","");
        Boolean status = getSharedPreferences("myinfo",0).getBoolean("st",false);
        if(status == true){
            edit_inputname.setText(strname);
            edit_inputpwd.setText(strpwd);
            check_reme.setChecked(true);
        }else{
            edit_inputname.setText("");
            edit_inputpwd.setText("");
            check_reme.setChecked(false);
        }
    }
}
```

上述代码中,在按钮单击事件的内部,先将用户名和密码保存起来,再跳转到下一个页面。当用户第二次运行程序时会根据复选框是否被选中的状态进行判断,从而确定是否显示用户名和密码。

(8) 修改 AndroidManifest.xml 配置文件,使程序运行时从第一个界面启动,代码如下。

```
<activity android:name=".AddInfoActivity"></activity>
<activity android:name=".LoginActivity" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity android:name=".MainActivity">
</activity>
...
```

(9) 运行程序,效果如图 5.13 所示。


微课视频

【任务 5-2】 使用 RecyclerView 设计主界面

(1) 设计第二个 Activity 的布局界面。因在第二个布局界面中用到了 Android 新增的控件 RecyclerView,为了让 RecyclerView 在所有的 Android 版本上都能使用,Android 团队将 RecyclerView 定义在了 Support 库当中,如果想使用 RecyclerView 这个控件,只需在 Project 项目结构中 app→build.gradle 文件中添加相应的依赖库才行。打开 app/build.gradle 文件,在 dependencies 闭包中添加如下内容。

图 5.13　登录界面效果图

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:+'
    ...
    implementation 'com.android.support:recyclerview-v7:+'
    }
```

需要注意的是，recyclerview 版本号要与第三行代码 support 的版本号保持一致。添加完成后，单击本页面右上角的 Sync Now 来进行同步。

（2）修改主界面的布局界面。双击 layout 文件夹下的 activity_main.xml 文件，便可打开布局编辑器，修改布局类型为 LinearLayout，添加方向属性为 vertical。依次添加所需控件，代码如下。

```
<?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">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="愿这小小的备忘录，记下我生活中的点点滴滴"
        android:textStyle="bold"
```

```
            android:textColor = "#000000"
            android:layout_gravity = "center"
            android:layout_margin = "10dp"          />
        <Button
            android:id = "@ + id/button_add"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "添加备忘录"
            android:textStyle = "bold"
            android:textColor = "#000000"
            android:layout_gravity = "right"
            android:background = "@drawable/buttonbg"
            android:layout_margin = "10dp"          />
        <android.support.v7.widget.RecyclerView
            android:id = "@ + id/recy_view"
            android:layout_width = "match_parent"
            android:layout_height = "wrap_content"/>
    </LinearLayout>
```

上述代码设计的布局界面非常简单，整体为线性布局，自上而下的垂直方向，TextView控件用来显示一段文字，Button控件用来跳转到下个页面，RecyclerView控件用来显示用户添加的备忘录信息。

（3）实现第二个Activity页面的功能。打开MainActivity.java类文件，修改MainActivity的代码，如下。

```
public class MainActivity extends AppCompatActivity {
//定义对象
    private Button btn_add;
    private RecyclerView recy_view;
    @Override
     protected void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.activity_main);
         //1.绑定控件
         initView();
         //2.对按钮添加单击事件
         btnonclicknext();
      }
//1.绑定控件
private void initView() {
     btn_add = findViewById(R.id.button_add );
     recy_view = findViewById(R.id.recy_view );
}
//2.对按钮添加单击事件
private void btnonclicknext() {
         btn_add.setOnClickListener(new View.OnClickListener() {
             @Override
             public void onClick(View view) {
```

```
                //单击后跳转到一下页
                Intent intent = new Intent(MainActivity.this,AddInfoActivity.class);
                startActivity(intent);
                finish();
            }
        });
    }
}
```

上述代码是主界面的一部分代码，由于用户还未添加备忘录信息，因此 RecyclerView 控件并未显示任何内容，在完成备忘录添加功能之后，再来完善本页面的功能代码。

(4) 运行程序，效果如图 5.14 所示。

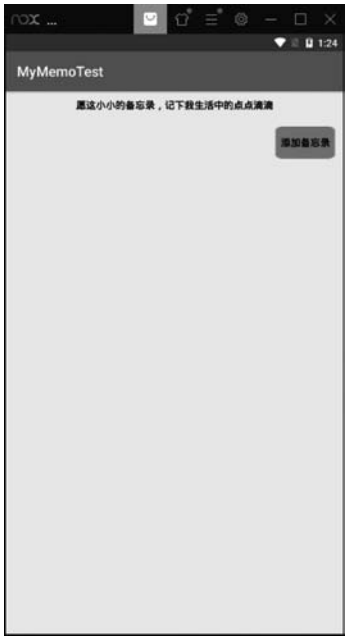

图 5.14　主界面设计效果图

【任务 5-3】 使用 SQLite 数据库实现信息添加

(1) 设计第三个 Activity 的布局界面。双击 layout 文件夹下的 activity_ add_info. xml 文件，便可打开布局编辑器，修改布局类型为 LinearLayout，添加方向属性为 vertical。依次添加所需控件，代码如下。

```
<?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:background = "@drawable/bgthree"
    tools:context = ".AddInfoActivity">
```

```
<EditText
    android:id="@+id/editText_title"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="备忘录的标题"
    android:textStyle="bold"
    android:textColor="#000000"
    android:textSize="30sp"/>
<View
    android:layout_width="match_parent"
    android:layout_height="2dp"
    android:background="#009688"
    android:layout_marginTop="-10dp"/>
<EditText
    android:id="@+id/editText_content"
    android:layout_width="match_parent"
    android:layout_height="200dp"
    android:hint="备忘录的内容"
    android:textStyle="bold"
    android:textColor="#000000"
    android:textSize="20sp"
    android:gravity="top"/>
<View
    android:layout_width="match_parent"
    android:layout_height="2dp"
    android:background="#009688"
    android:layout_marginTop="-10dp"/>
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="请选用下面的任一种方式,添加一张心情图片:"
    android:textStyle="bold"
    android:textColor="#000000"
    android:textSize="15sp"
    android:gravity="top"
    android:layout_margin="10dp"/>
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_margin="10dp">
    <Button
        android:id="@+id/button_camera"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="拍照"
        android:textStyle="bold"
        android:textColor="#000000"
        android:textSize="15sp"
        android:layout_margin="10dp"/>
```

```
        <Button
            android:id="@+id/button_photo"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="从图库中选择"
            android:textStyle="bold"
            android:textColor="#000000"
            android:textSize="15sp"
            android:layout_margin="10dp"/>
    </LinearLayout>
    <ImageView
        android:id="@+id/imageView_preview"
        android:layout_width="wrap_content"
        android:layout_height="200dp"
        android:src="@drawable/sunshine"
        android:layout_marginBottom="20dp"
        android:layout_gravity="center"/>
    <Button
        android:id="@+id/button_save"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="保存"
        android:textStyle="bold"
        android:textColor="#000000"
        android:textSize="30sp"
        android:background="@drawable/savebg"
        android:layout_margin="10dp"/>
</LinearLayout>
```

上述代码中，View 控件用来定义一条线，使用 layout_height 属性设置线的粗细，使用 background 属性设置线的颜色，layout_marginTop 属性取值为－10 的目的是想让这条线与 EditText 控件输入框底边线重合，增强页面的美观效果。

(2) 创建数据库类文件。因考虑到用户会存放很多的备忘录信息，而且每条备忘录信息包含文字和图片两种不同类型的数据，因此采用 SQLite 数据库存放备忘录信息。为了使程序文件条理清晰，创建 db 文件夹来存放数据库文件。方法为：右击 com.example.mymemotest→New→Package→输入名称“db”→ OK，便可完成 db 文件夹的创建。然后创建数据库类文件，右击 db→New→Java Class→在 Name 中输入类名“MyDbHelper”→Superclass 中输入父类“SQLiteOpenHelper”，会自动出现全称 android.database.sqlite.SQLiteOpenHelper→OK。在数据库类文件中输入如下代码。

```
public class MyDbHelper extends SQLiteOpenHelper {
    //定义数据库名和数据库的版本号
    private static String DBNAME = "zsmemo.db";
    private static int VERSION = 1;
    //构造方法
    public MyDbHelper( Context context) {
        super(context, DBNAME, null, VERSION );
```

```
    }
     //创建数据库
    @Override
    public void onCreate(SQLiteDatabase db) {
        //创建数据表
         db.execSQL("create table tb_memory(_id Integer primary key,title String (200),
content String (2000),imgpath String (200),mtime String (50))");
    }
    //升级数据库
    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
    }
}
```

上述代码中，SQLiteOpenHelper 是专门管理数据库的帮助类，借助于这个类就可以非常简单地对数据库进行创建和升级。SQLiteOpenHelper 是一个抽象类，使用时需要创建一个自定义的帮助类去继承它。

SQLiteOpenHelper 中有两个抽象方法，分别是 onCreate()和 onUpgrade()方法，用户必须在自定义的帮助类里面重写这两个方法，然后分别在这两个方法中去实现创建、升级数据库的逻辑。SQLiteOpenHelper 中还有两个非常重要的实例方法：getWritableDatabase()和 getReadableDatabase()，这两个方法用于创建或打开一个现有的数据库，并返回一个可对数据库进行读写操作的对象 SQLiteDatabase。

SQLiteOpenHelper 的构造方法中接收四个参数：第一个参数是上下文环境 Context；第二个参数是数据库名；第三个参数是查询数据的时候返回一个自定义的 Cursor，一般传入 null；第四个参数是当前数据库的版本号，版本号用于对数据库进行升级操作。SQLiteDatabase 的 execSQL()方法创建数据表 tb_memory。

(3) 添加图片加载库 Glide 依赖，同时添加图片缩略图路径与真实路径转换的依赖，方便第三个页面中图片控件中显示拍照图片或者显示从相册中选择的图片。打开 Project/app/build.gradle 文件，在 dependencies 闭包中添加如下内容。

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:+'
    ...
    implementation 'com.android.support:recyclerview-v7:+'
    implementation 'com.github.bumptech.glide:glide:4.9.0'
    api 'com.blankj:utilcode:1.23.7'
    }
```

(4) 实现第三个 Activity 页面的功能。打开 AddInfoActivity.java 类文件，修改 AddInfoActivity 的代码，具体如下。

```
public class AddInfoActivity extends AppCompatActivity {
//定义对象
   private EditText edit_title,edit_content;
```

```
    private Button btn_camera,btn_photo,btn_save;
    private ImageView img_preview;
    private String tmp_path,disp_path;
    private MyDbHelper mhelper;
    private SQLiteDatabase db;
     @Override
     protected void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.activity_add_info );
         //1.绑定控件
         initView();
         //2.单击按钮、拍照、从图库中选择照片
         btnOnClick();
         //3.接受拍好照片、接受从图库当中选择的照片 ------方法:系统回调
         //4.把信息保存到数据库中
         btnSave();
     }
     //1.绑定控件
     private void initView() {
         edit_title = findViewById(R.id.editText_title );
         edit_content = findViewById(R.id.editText_content );
         btn_camera = findViewById(R.id.button_camera );
         btn_photo = findViewById(R.id.button_photo );
         img_preview = findViewById(R.id.imageView_preview );
         btn_save = findViewById(R.id.button_save );
         mhelper = new MyDbHelper(AddInfoActivity.this); //
         db =  mhelper.getWritableDatabase();
     }
     //2.单击按钮、拍照
     private void btnOnClick() {
         btn_camera.setOnClickListener(new View.OnClickListener() {
             @Override
             public void onClick(View view) {
                 //拍照
                 Time time = new Time();
                 time.setToNow();
    String randtime = time.year + (time.month + 1) + time.monthDay + time.hour + time.minute +
time.second + "";
                 tmp_path =  Environment.getExternalStorageDirectory() + "/image" +  randtime
 + ".jpg";
                 File imgfile = new File(tmp_path);
                 try {
                     imgfile.createNewFile();
                 } catch (IOException e) {
                     e.printStackTrace();
                 }
                 Intent intent = new Intent("android.media.action.IMAGE_CAPTURE");
                 intent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(imgfile) );
                 startActivityForResult(intent,11);
             }
```

```
        });
        //从相册中选择图片
        btn_photo.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //选择照片
                Intent intent = new Intent("android.intent.action.GET_CONTENT");
                intent.setType("image/*");
                startActivityForResult(intent,22);
            }
        });
    }
    //3.接受拍好照片、接受从图库当中选择的照片 ------方法:系统回调
    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent
data) {
        switch (requestCode){
            case 11:
                if(resultCode == RESULT_OK){
                    disp_path = tmp_path;
                    Glide.with(AddInfoActivity.this).load(disp_path).into(img_preview);
                }
                break;
            case 22:
                Uri imageuri = data.getData();
                if (imageuri == null)
                {
                    return;
                }
               disp_path = UriUtils.uri2File(imageuri).getPath();
                Glide.with(AddInfoActivity.this).load(disp_path).into(img_preview);
                break;
                default:
                    break;
        }
    }
    //4.把信息保存到数据库中
    private void btnSave() {
        btn_save.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //保存信息到数据库代码
                Time time = new Time();
                time.setToNow();
                ContentValues contentValues = new ContentValues(); //1行
                contentValues.put("title",edit_title.getText().toString());
                                                                    //1行——title列
                contentValues.put("content",edit_content.getText().toString());
                                                                    //1行——content列
```

```
                contentValues.put("imgpath",disp_path); //1行——imgpath列
                contentValues.put("mtime",time.year + "/" + (time.month + 1) + "/" + time.
monthDay); // mtime列
                db.insert("tb_memory",null,contentValues);//将该行数据保存到数据表中
                Toast.makeText (AddInfoActivity.this,"保存成功",Toast .LENGTH_SHORT ).show();
                //跳转到主界面
                Intent intent = new Intent(AddInfoActivity.this,MainActivity.class);
                startActivity(intent);
                finish();
            }
        });
    }
}
```

上述代码中，调用摄像头拍照和从相册中选择图片的代码在5.3节中已学习过，这里不再重复讲解，需要说明的是，本页面是如何把数据保存到数据库中的呢？首先定义数据库帮助类MyDbHelper和数据库类SQLiteDatabase；其次调用new MyDbHelper(AddInfoActivity.this)语句实例化数据库帮助类，对象名为mhelper，调用数据库帮助类的getWritableDatabase()方法获得SQLiteDatabase数据库对象，对象名为db，借助于SQLiteDatabase对象就可以对数据库中的数据表进行增删改查操作了。

本页面主要实现了向数据库中tb_memory数据表添加数据，首先调用new ContentValues()方法构建一行数据，其次采用put()方法将用户输入的备忘录标题放到title列、备忘录内容放到content列、心情图片路径放到imgpath列、添加备忘录的时间放到mtime列，最后调用数据库的insert()方法将本行数据保存到数据表中。数据保存结束后跳转到主界面，及时查看添加的备忘录信息。

(5) 声明权限。双击打开AndroidManifest.xml配置文件，具体代码如下。

```
<manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    package = "com.example.filesavetest">
    <uses - permission android:name = "android.permission.CAMERA"/>
    <uses - permission android:name = "android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses - permission android:name = "android.permission.READ_EXTERNAL_STORAGE"/>

    <application
    …
```

上述代码指定了当前SD卡具有可写可读的权限，同时指定了本程序具有访问系统相机的权限，因此应用程序不仅可以打开系统相机拍照，而且还可以操作SD卡中的图片文件。

(6) 信息添加页面完成后，运行程序，在页面中输入标题、内容、图片等数据内容，单击“保存”按钮完成数据的添加。效果如图5.15(a)所示。

(7) 查看保存在数据库中的信息。在模拟器中打开RE文件管理器，挂载为可读写模式，根据存储路径依次操作：data/data/com.example.mymemotest/databases/zsmemo.db/tb_memory，双击tb_memory数据表将其打开，看到了保存的数据信息，效果如图5.15

(b)所示。用户可以将模拟器切换到横屏模式查看完整的数据表字段。

图 5.15　信息添加界面及查看存储的数据

【任务 5-4】 读取 SQLite 数据库中数据显示到备忘录主界面

RecyclerView 控件显示数据时，需要提前准备好数据、item 子布局、适配器、布局管理器。①数据：由于 RecyclerView 是滚动列表，可显示大量数据，因此一般会用 List 来存放 RecyclerView 控件所需要的数据。②item 子布局：RecyclerView 显示数据时，每一行都是一个 item，该行中的文字和图片排放位置如何，大小间距如何，都需要通过 item 子布局来设定每行数据显示的外观。③适配器：要将 List 中的数据以 item 子布局的外观样式显示出来，那么就需要在数据和子布局之间架起一座桥梁，这个桥梁即为适配器，通过适配器来控制数据以子布局设定的外观样式展示给用户。④布局管理器：RecyclerView 中的数据既可以横向滚动，又可以纵向滚动，用户可根据需要挑选 RecyclerView 控件的布局管理器。

(1) 新建数据类 MemoBean。为了让 RecyclerView 能够显示很多数据，定义 List 动态数组列表来存放数据，List 列表支持单一数据类型，不能把文字和图片两种不同类型的数据单独放到里面，此时需要把备忘录标题、备忘录内容、心情图片三个数据封装成一个类，以 List < MemoBean > arr1 方式定义列表。这种数据类 MemoBean 有一个特殊的统称，被称为标准的 JavaBean。为了使程序条理清晰，创建 bean 文件夹来存放 JavaBean 类文件。方法为：右击 com. example. mymemotest→New→Package→输入名称“bean”→OK，便可完成 bean 文件夹的创建。然后创建 JavaBean 类文件，右击 bean→New→Java Class→在 Name 中输入类名“MemoBean”→OK。双击 MemoBean 类文件将其打开，输入如下代码。

```
public class MemoBean {
    //定义四个属性
    private String title;
    private String content;
    private String imgpath;
    private String time;
//按 Alt + Insert 组合键,选择 Contructor 构造方法,全选四个属性,单击 OK 按钮
    public MemoBean(String title, String content, String imgpath, String time) {
        this.title = title;
        this.content = content;
        this.imgpath = imgpath;
        this.time = time;
    }
//按 Alt + Insert 组合键,选择 Getter and Setter 方法,全选四个属性,单击 OK 按钮
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getContent() {
        return content;
    }
    public void setContent(String content) {
        this.content = content;
    }
    public String getImgpath() {
        return imgpath;
    }
    public void setImgpath(String imgpath) {
        this.imgpath = imgpath;
    }
    public String getTime() {
        return time;
    }
    public void setTime(String time) {
        this.time = time;
    }
}
```

上述代码中,在 MemoBean 类文件中定义了四个属性,分别是 title、content、imgpath、time,这四个属性正好是 List 动态数组中要存放的四个数据,也是主页面 RecyclerView 控件中需要显示的内容,同时还是数据库 tb_memory 数据表里的四列字段名。

(2) 新建 RecyclerView 子布局。在子布局中明确标题、内容和图片的位置关系。将工程依次展开为 app→src→main→res→layout,右击 layout→new→Layout resource file→file name 输入"recy_item"→OK,双击打开 recy_item.xml 布局文件,修改布局代码如下。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout
    xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "match_parent"
    android:layout_height = "wrap_content"
    android:orientation = "horizontal"
    android:background = "#7AECCC"
    android:id = "@ + id/item_layout"
    android:layout_margin = "5dp" >
    < LinearLayout
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:orientation = "vertical"
        android:layout_gravity = "center"
        android:layout_weight = "1"
        android:layout_margin = "5dp" >
        < TextView
            android:id = "@ + id/item_title"
            android:layout_width = "match_parent"
            android:layout_height = "wrap_content"
            android:text = "标题"
            android:textSize = "20sp"
            android:textStyle = "bold"
            android:textColor = "#000000"/>
        < TextView
            android:id = "@ + id/item_content"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "内容"
            android:textColor = "#000000"/>
    </LinearLayout >
    < LinearLayout
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:orientation = "vertical"
        android:layout_margin = "5dp">
        < ImageView
            android:id = "@ + id/item_image"
            android:layout_width = "100dp"
            android:layout_height = "100dp"
            android:src = "@mipmap/ic_launcher_round"/>
        < TextView
            android:id = "@ + id/item_time"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "时间"
            android:textColor = "#000000"
            android:layout_gravity = "center"/>
    </LinearLayout >
</LinearLayout >
```

上述代码中，子布局总体采用水平方向线性布局，左侧嵌套垂直方向线性布局用来显示标题和内容，右侧嵌套垂直方向的线性布局，用来显示图片和日期，如图 5.16 所示。

图 5.16　RecyclerView 控件的子布局 item

（3）建立 RecyclerView 适配器 MemoAdapter。为了让 JavaBean 中数据能够以 recy_item 子布局的外观样式显示到 RecyclerView 控件上，需要在数据与子布局之间搭建一个桥梁，即适配器。为了使程序文件条理清晰，创建 adapter 文件夹来存放适配器类文件。方法为：右击 com.example.mymemotest→New→Package→输入名称“adapter”→OK，便可完成 adapter 文件夹的创建。然后创建 JavaBean 类文件，右击 bean→New→Java Class→Name 中输入类名“MemoAdapter”→Superclass 中输入父类“RecyclerView.Adapter”，会自动出现全称 android.support.v7.widget.RecyclerView.Adapter→OK。在适配器类文件中输入如下代码。

```
//1.类文件后面添加泛型
//2.鼠标定位类文件行红色波浪线处,按 Alt + Enter 组合键:添加未实现的方法
//3.鼠标定位类文件行 ViewHolder 处,按 Alt + Enter 组合键:添加内部类
//4.鼠标定位界面最下方内部类 ViewHolder 处,添加 extends RecyclerView.ViewHolder
//5.鼠标定位界面最下方内部类 ViewHolder 红色波浪线处,按 Alt + Enter 组合键:添加构造方法
//6.定义两个对象:上下文环境和数组
//7.定义两个对象下方的空白处:Alt + Insert 键,添加适配器的构造方法
public class MemoAdapter extends RecyclerView.Adapter < MemoAdapter.ViewHolder > {
    private Context mcontext;
    private List < MemoBean > arr1;
    public MemoAdapter(Context mcontext, List < MemoBean > arr1) {
        this.mcontext = mcontext;
        this.arr1 = arr1;
    }
//负责加载 item 布局
    @NonNull
    @Override
    public MemoAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int i) {
         View view = LayoutInflater.from(mcontext).inflate(R.layout.recy_item, parent,
false);
        ViewHolder mholder = new ViewHolder(view);
        return mholder;
    }
//负责加载 item 的数据
    @Override
    public void onBindViewHolder(@NonNull MemoAdapter.ViewHolder mholder, int i) {
        final MemoBean memoBean = arr1.get(i);
        mholder.item_title.setText(memoBean.getTitle());
```

```
            mholder.item_content.setText(memoBean.getContent());
            mholder.item_time.setText(memoBean.getTime());
            Glide.with(mcontext).load(memoBean.getImgpath()).into(mholder.item_img);
        }
    //recyView一共有多少个子项
        @Override
        public int getItemCount() {
            return arr1.size();
        }
        public class ViewHolder extends RecyclerView.ViewHolder {
            TextView item_title,item_content,item_time;
            ImageView item_img;
            LinearLayout item_layout;
            public ViewHolder(@NonNull View itemView) {
                super(itemView);
                item_title = itemView.findViewById(R.id.item_title);
                item_content = itemView.findViewById(R.id.item_content );
                item_img = itemView.findViewById(R.id.item_image );
                item_time = itemView.findViewById(R.id.item_time );
                item_layout = itemView.findViewById(R.id.item_layout );
            }
        }
    }
```

上述代码中，最下方首先定义了一个内部类 ViewHolder，ViewHolder 要继承自 RecyclerView.ViewHolder。然后在 ViewHolder 的构造方法中传入一个 itemView 参数，这个参数就是 RecyclerView 子项的最外层布局，通过 itemView.findViewById 方法获取布局中的 TextView 和 ImageView 及 LinearLayout 控件的实例。

由于 MemoAdapter 继承了 RecyclerView.Adapter，那么就必须重写 3 个方法。onCreateViewHolder()方法是用户创建 ViewHolder 实例的，在这个方法中把 recy_item 子布局加载进来。onBindViewHolder()方法用于对 RecyclerView 子项进行赋值，根据子项的位置 i 得到当前项的 MemoBean 实例，然后再将数据显示到对应的控件上。getItem()方法用于告诉 RecyclerView 一共有多少子项，直接返回数据源数组的长度即可。

(4) 完善 MainActivity 页面的功能，实现备忘录信息的展示。打开 MainActivity.java 类文件，修改 MainActivity 的代码如下。

```
public class MainActivity extends AppCompatActivity {
//定义对象
   private Button btn_add;
   private RecyclerView recy_view;
   private MyDbHelper mhelper;
   SQLiteDatabase db;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
        //1.绑定控件
        initView();
        //2.对按钮添加单击事件
        btnonclicknext();
        //3.完善:从数据库获取数据,显示到 RecyclerView 控件中
        recyDisplay();
    }
    //1.绑定控件
    private void initView() {
        btn_add = findViewById(R.id.button_add);
        recy_view = findViewById(R.id.recy_view);
        mhelper = new MyDbHelper(MainActivity.this);
        db = mhelper.getWritableDatabase();
    }
    //2.对按钮添加单击事件
    private void btnonclicknext() {
            btn_add.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    //单击后跳转到下一页
                    Intent intent = new Intent(MainActivity.this,AddInfoActivity.class);
                    startActivity(intent);
                    finish();
                }
            });
    }
    //3.完善:从数据库获取数据,显示到 RecyclerView 控件中
    private void recyDisplay() {
        //3.1 准备数据 ------------ 标题、内容、图片、时间(类)
       List<MemoBean> arr = new ArrayList();
       //从数据库中取数据
       Cursor cursor = db.rawQuery("select * from tb_memory",null);
      while(cursor.moveToNext()){
          String mytitle = cursor.getString(cursor.getColumnIndex("title"));
          String mycontent = cursor.getString(cursor.getColumnIndex("content"));
          String myimgpath = cursor.getString(cursor.getColumnIndex("imgpath"));
          String mymtime = cursor.getString(cursor.getColumnIndex("mtime"));
          MemoBean memoBean = new MemoBean(mytitle,mycontent,myimgpath,mymtime);
          arr.add(memoBean);
      }
       cursor.close();
        //3.2 子布局 recy_item
        //3.3 数据 ------ 桥(适配器 MemoAdapter) ---------------- 子布局
        MemoAdapter adapter = new MemoAdapter(MainActivity.this,arr);
        //3.4 确定显示的方式
StaggeredGridLayoutManager st = new StaggeredGridLayoutManager(2,StaggeredGridLayoutManager.
VERTICAL);
        recy_view.setLayoutManager(st);
        //3.5 让数据显示出来
```

```
        recy_view.setAdapter(adapter);
    }
}
```

(5) 备忘录删除功能的实现。单击页面上的某一个备忘录内容,在弹出的对话框中单击“删除”按钮,便可删除备忘录信息,同时刷新 RecyclerView 信息列表以显示最新的备忘录内容。修改每条备忘录的背景颜色,采用随机数字的方式设置背景颜色,同时设置 item 为圆角矩形。双击打开 MemoAdapter 适配器,在 onBindViewHolder()方法中完善代码,具体如下。

```
public class MemoAdapter extends RecyclerView.Adapter<MemoAdapter.ViewHolder>{
    private Context mcontext;
    private List<MemoBean> arr1;
    private MyDbHelper mhelper1;
    private SQLiteDatabase db;

    …//中间的这部分代码保持不变

//负责加载 item 的数据
@RequiresApi(api = Build.VERSION_CODES.JELLY_BEAN)
@Override
public void onBindViewHolder(@NonNull MemoAdapter.ViewHolder mholder, final int i) {
    final MemoBean memoBean = arr1.get(i);
    mholder.item_title.setText(memoBean.getTitle());
    mholder.item_content.setText(memoBean.getContent());
    mholder.item_time.setText(memoBean.getTime());
    Glide.with(mcontext).load(memoBean.getImgpath()).into(mholder.item_img);

   //完善:设置 RecyclerView 中每一个子项的颜色和形状
        Random random = new Random();
         int color = Color.argb(255, random.nextInt(256), random.nextInt(256), random.
nextInt(256));
        GradientDrawable gradientDrawable = new GradientDrawable();
        gradientDrawable.setShape(GradientDrawable.RECTANGLE);          //形状
        gradientDrawable.setCornerRadius(10f);                          //设置圆角 Radius
        gradientDrawable.setColor(color);                               //颜色
        mholder.item_layout.setBackground(gradientDrawable);            //设置为 background

  //完善:单击其中的一个子项,弹出删除功能
        mholder.item_layout.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //弹出对话框,删除
                AlertDialog.Builder dialog = new AlertDialog.Builder(mcontext);
                dialog.setMessage("确定删除吗?");
                dialog.setPositiveButton("确定", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int abc) {
```

```
                        //从数据库当中删除掉
                        mhelper1 =  new MyDbHelper(mcontext);
                        db = mhelper1.getWritableDatabase();
                        db.delete("tb_memory","title = ?",new String[]{arr1.get(i).getTitle
()}); //删除
                        arr1.remove(i); //从 list 动态数组中把该项移除
                        notifyItemRemoved(i); //刷新 RecyclerView 显示的内容
                        dialogInterface.dismiss(); //对话框消失
                    }
                });
                dialog.setNegativeButton("取消",null);
                dialog.setCancelable(false);
                dialog.create();
                dialog.show();
            }
        });

    }
    …//后面的代码保持不变
    …
```

（6）运行程序，单击第一条备忘录，弹出对话框，单击“确定”按钮便将该条备忘录删除，RecyclerView 会及时刷新并显示最新数据，效果如图 5.17 所示。

图 5.17　备忘录数据的删除

项目小结

本项目内容紧紧围绕 Android 中的 SharedPreferences 存储、Android 的文件存储、调用摄像头和相册以及运行时权限设置内容来展开，使用户根据存储数据的特点选择不同的存储方式，能够将拍摄的图片根据需要随时存放到内存或外存中。通过学习 Android 运行时权限，保障了用户的信息安全。最后，通过备忘录案例，培养学生具备复杂 App 的设计与开发能力。

习题

1. SharedPreferences 所存储的数据以(　　)的格式保存在 XML 文件中。

A. “姓-名”　　B. “键-值”　　C. 文件　　D. 字符

2. Android 的几种存储方式中，(　　)是以键值对的方式来存储数据的。

A. SharedPreferences 存储　　B. 文件存储

C. SQLite 数据库存储　　D. SD 卡存储

3. getSharedPreferences("myfile",0)方法中的第二个参数 0 表示以追加模式保存数据。(　　)

A. 对　　B. 错

4. Android 提供了标准的 Java 文件(　　)方式来对文件数据进行读写。

A. “姓-名”　　B. “键-值”　　C. 字符　　D. 输入输出流

5. Context 类的(　　)方法可以获得文件输入流对象。

A. openFileOutput　　B. openFileInput

6. 下面(　　)方法是关闭文件输入/输出流的方法。

A. close()　　B. exit()

7. 打开摄像头和相册的系统页面，属于(　　)跳转。

A. 显式跳转　　B. 隐式跳转

8. 下面(　　)可以打开摄像头。

A. android.media.action.IMAGE_CAPTURE

B. android.intent.action.GET_CONTENT

9. 在 Android 项目中，图片加载库有很多选择，常用的是(　　)。

A. Glide　　B. Image　　C. picture　　D. icon

10. 下面(　　)控件不能直接使用，需要添加(　　)依赖才可使用。

A. ImageView　　B. TextView

C. EditText　　D. RecyclerView