```
    }
}
```

重新运行项目，执行更新数据操作，将显示对话框提示错误信息，如图 5.3 所示。

图 5.3

知识拓展

Android 提供了针对常见多媒体格式的编码、解码 API，可以非常方便地操作图片、音频、视频等多媒体文件，也可以操纵 Android 终端的录音、摄像设备。本章知识拓展将简单介绍 Android 的多媒体应用。

1. 播放音频、视频

使用 android.media.MediaPlayer 类可以实现音频、视频文件的播放，包括播放、暂停、停止、重复播放等功能，文件可以位于本地文件系统或者项目资源目录，也可以是网络的文件流。

（1）播放音频

下列内容实现了简单的音频播放功能，编写布局文件如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:a="http://schemas.android.com/apk/res/android"
    a:layout_width="fill_parent"
    a:layout_height="wrap_content">
    <Button
        a:id="@+id/startBtn"
        a:layout_width="fill_parent" a:layout_height="wrap_content"
```

```
        a:text="开始" a:layout_weight="1"/>
    <Button
        a:id="@+id/pauseBtn" a:layout_weight="1"
        a:text="暂停" a:enabled="false"
        a:layout_width="fill_parent" a:layout_height="wrap_content"/>
    <Button
        a:id="@+id/stopBtn" a:text="停止"
        a:enabled="false" a:layout_weight="1"
        a:layout_width="fill_parent" a:layout_height="wrap_content"/>
</LinearLayout>
```

上述文件中，定义了开始、暂停、停止三个按钮，编写 Activity 代码如下：

```
public class SoundPlayer extends Activity {

    MediaPlayer player;

    Button startBtn;
    Button pauseBtn;
    Button stopBtn;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.sound);

        // 播放 res/raw 目录下的 windowsxp 文件
        player = MediaPlayer.create(SoundPlayer.this, R.raw.windowsxp);
        startBtn = (Button) findViewById(R.id.startBtn);
        pauseBtn = (Button) findViewById(R.id.pauseBtn);
        stopBtn = (Button) findViewById(R.id.stopBtn);

        // 播放完成事件
        player.setOnCompletionListener(new OnCompletionListener() {
            @Override
            public void onCompletion(MediaPlayer mp) {
                startBtn.setEnabled(true);
                pauseBtn.setEnabled(false);
                stopBtn.setEnabled(false);
            }
        });

        // 开始按钮
```

```
        startBtn.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                player.start(); // 开始播放
                startBtn.setEnabled(false);
                pauseBtn.setEnabled(true);
                stopBtn.setEnabled(true);
            }
        });

        // 暂停按钮
        pauseBtn.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                player.pause(); // 暂停播放
                startBtn.setEnabled(true);
                pauseBtn.setEnabled(false);
                stopBtn.setEnabled(true);
            }
        });

        // 停止按钮
        stopBtn.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                player.stop(); // 停止播放
                startBtn.setEnabled(true);
                pauseBtn.setEnabled(false);
                stopBtn.setEnabled(false);
                try {
                    player.prepare(); // 为下次播放做准备
                    player.seekTo(0); // 回到音频起点
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        player.release(); // 释放资源
```

```
    }
}
```

上述代码中，调用 MediaPlayer.create()方法创建了 MediaPlayer 对象，并指定播放 res/raw 目录下的 windowsxp 资源文件；分别为开始、暂停、停止按钮注册单击事件；为 MediaPlayer 注册了 OnCompletionListener 事件，当播放完毕时会自动调用其 onCompletion()方法。运行项目，单击“开始”按钮，将播放 res/raw 目录下的 windowsxp.wmv 文件，单击“暂停”按钮会暂停播放，再次单击“开始”按钮，将继续播放；单击“停止”按钮，将结束播放，效果如图 5.4 所示。

图 5.4

（2）播放视频

使用 MediaPlayer 类也可以播放视频文件，但是比较复杂，Android 提供了 VideoView 控件，其内置了 MediaPlayer，使用 VideoView 结合 MediaController 类可以方便地实现视频播放。

下列内容实现了简单的视频播放功能，编写布局文件如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:a="http://schemas.android.com/apk/res/android"
    a:layout_width="fill_parent"
    a:layout_height="wrap_content">
    <VideoView
        a:id="@+id/videoView"
        a:layout_width="fill_parent" a:layout_height="fill_parent"/>
</LinearLayout>
```

上述文件中，定义了一个 VideoView 控件。编写 Activity 代码如下：

```
public class VideoPlayer extends Activity {

    VideoView videoView;
    MediaController controller;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```



```
        // 不
        reque
        // 全
        getWi

        // 横
        setRe
        setCo

        vide
        cont
        // 设
        vide

        // 关
        cont
        vide
        // 开
        vide
    }
}
```

上述代码
后指定了 Vide
式指定；还需
播放，运行项

单击屏幕
进度条，使用

```
        // 不显示标题
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        // 全屏
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN);
        // 横屏
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
        setContentView(R.layout.video);

        videoView = (VideoView) findViewById(R.id.videoView);
        controller = new MediaController(this);
        // 设置播放的文件
        videoView.setVideoURI(Uri.parse("android.resource://"
                + getPackageName() + "/" + R.raw.td));
        // 关联 VideoView 和 MediaController
        controller.setMediaPlayer(videoView);
        videoView.setMediaController(controller);
        // 开始播放
        videoView.start();
    }
}
```

上述代码中，首先设置不显示应用程序标题、全屏以及横屏，这样更适合播放视频；然后指定了 VideoView 播放的视频文件，VideoView 无法直接访问资源文件，需要以 URI 的形式指定；还需要关联 VideoView 和 MediaController，最后调用 VideoView 的 start()方法开始播放，运行项目，将显示正在播放的视频文件，如图 5.5 所示。

单击屏幕后，将显示播放控制区，其中包括前进、后退、暂停按钮，以及一个可拖动的进度条，使用这些可以调整播放进度，如图 5.6 所示。

图 5.5

图 5.6

2. 录制音频、视频

Android 提供了对音频、视频录制的支持，如果终端带有相应的设备，则应用程序可以录音、录像。音频、视频的录制主要依靠 MediaRecorder 类实现，下述内容完成简单的音频、视频录制功能。编写布局文件如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:a="http://schemas.android.com/apk/res/android"
    a:orientation="vertical"
    a:layout_width="fill_parent" a:layout_height="fill_parent">
    <TextView
        a:text="录音" a:textSize="20sp"
        a:layout_width="fill_parent" a:layout_height="wrap_content"/>
    <LinearLayout
        a:layout_width="fill_parent" a:layout_height="wrap_content">
        <Button
            a:id="@+id/startBtn1"
            a:text="开始" a:layout_weight="1"
            a:layout_width="fill_parent" a:layout_height="wrap_content"/>
        <Button
            a:id="@+id/stopBtn1" a:text="停止"
            a:enabled="false" a:layout_weight="1"
            a:layout_width="fill_parent" a:layout_height="wrap_content"/>
    </LinearLayout>
    <TextView
        a:text="录像" a:textSize="20sp"
        a:layout_width="fill_parent" a:layout_height="wrap_content"/>
    <LinearLayout
        a:layout_width="fill_parent" a:layout_height="wrap_content">
        <Button
            a:id="@+id/startBtn2"
            a:text="开始" a:layout_weight="1"
            a:layout_width="fill_parent" a:layout_height="wrap_content"/>
        <Button
            a:id="@+id/stopBtn2" a:text="停止"
            a:enabled="false" a:layout_weight="1"
            a:layout_width="fill_parent" a:layout_height="wrap_content"/>
    </LinearLayout>
    <SurfaceView
        a:id="@+id/view"
        a:layout_width="fill_parent" a:layout_height="fill_parent"/>
</LinearLayout>
```


上述文件

个 SurfaceVie

```
public clas

    MediaRe
    Button
    Button

    MediaRe
    Button
    Button
    Surface
    Surface

    @Overri
    public
        supe
        setC
        init
        init
    }

    // 录音
    void ini
        reco
        star
        stop

        // 音
        reco
        // 输
        reco
        // 编
        reco
        // 输
        reco

        star
```

上述文件中，针对录制音频和视频分别定义了开始和停止按钮，并在界面下部放置了一个 SurfaceView，用于录像时输出预览效果。编写 Activity 代码如下：

```
public class A extends Activity {

    MediaRecorder recorder1; // 录音的 MediaRecorder
    Button startBtn1;
    Button stopBtn1;

    MediaRecorder recorder2; // 录像的 MediaRecorder
    Button startBtn2;
    Button stopBtn2;
    SurfaceView view; // 录像时的预览视图
    SurfaceHolder holder;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        initAudio();
        initVideo();
    }

    // 录音
    void initAudio() {
        recorder1 = new MediaRecorder();
        startBtn1 = (Button) findViewById(R.id.startBtn1);
        stopBtn1 = (Button) findViewById(R.id.stopBtn1);

        // 音频源是麦克风
        recorder1.setAudioSource(MediaRecorder.AudioSource.MIC);
        // 输出格式 THREE_GPP
        recorder1.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
        // 编码格式 AMR_NB
        recorder1.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
        // 输出文件路径
        recorder1.setOutputFile("/sdcard/"
                + new SimpleDateFormat("yyyyMMddHHmmss").format(new Date())
                + ".3gp");
        startBtn1.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                try {
```

```
                recorder1.prepare(); // 准备录制
                recorder1.start(); // 开始录制
                startBtn1.setEnabled(false);
                stopBtn1.setEnabled(true);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
    stopBtn1.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            try {
                recorder1.stop(); // 停止录制
                recorder1.release(); // 释放资源
                startBtn1.setEnabled(true);
                stopBtn1.setEnabled(false);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}

// 录像
void initVideo() {
    recorder2 = new MediaRecorder();
    startBtn2 = (Button) findViewById(R.id.startBtn2);
    stopBtn2 = (Button) findViewById(R.id.stopBtn2);
    view = (SurfaceView) findViewById(R.id.view);
    holder = view.getHolder();

    holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);

    // 使用 holder 的 surface 作为录像时的预览视图
    recorder2.setPreviewDisplay(holder.getSurface());
    // 音频源是麦克风
    recorder2.setAudioSource(MediaRecorder.AudioSource.MIC);
    // 视频源是摄像头
    recorder2.setVideoSource(MediaRecorder.VideoSource.CAMERA);
    // 输出格式 MPEG_4
    recorder2.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
    // 音频编码格式 AMR_NB
```

```
        recorder2.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
        // 视频分辨率
        recorder2.setVideoSize(320, 240);
        // 视频帧率
        recorder2.setVideoFrameRate(20);
        // 视频编码格式 H263 .
        recorder2.setVideoEncoder(MediaRecorder.VideoEncoder.H263);
        // 输出文件路径
        recorder2.setOutputFile("/sdcard/"
                + new SimpleDateFormat("yyyyMMddHHmmss").format(new Date())
                + ".mp4");
        startBtn2.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                try {
                    recorder2.prepare();// 准备录制
                    recorder2.start();// 开始录制
                    startBtn2.setEnabled(false);
                    stopBtn2.setEnabled(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
        stopBtn2.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                try {
                    recorder2.stop();// 停止录制
                    recorder2.release();// 释放资源
                    startBtn2.setEnabled(true);
                    stopBtn2.setEnabled(false);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }
}
```

上述代码中，针对音频录制，设置了音频源、输出格式、编码格式、输出文件路径；针对视频录制，设置了预览界面、音频源、视频源、输出格式、音频编码格式、视频分辨率、视频帧率、视频编码格式、输出文件路径；单击各自的开始和停止按钮时，调用 MediaRecorder

的 start()和 stop()方法开始和停止录制。

因为程序使用了麦克风、摄像头等硬件资源，因此，需要添加相应的权限，修改 AndroidMenifest.xml 文件，添加权限如下。

- android.permission.RECORD_AUDIO：录音权限；
- android.permission.WRITE_EXTERNAL_STORAGE：使用外部存储卡的权限；
- android.permission.CAMERA：使用摄像头的权限。

运行程序，界面如图 5.7 所示。

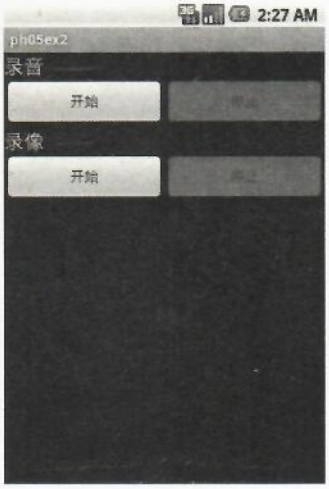

图 5.7

单击上部的“开始”按钮，将开始录音，单击“停止”按钮结束录音。单击下部的“开始”按钮，将开始录像，并在下方显示当前画面，单击“停止”按钮结束录像。录制的文件会保存在存储卡中。

注意 Android 模拟器不支持音频、视频捕捉，即使电脑上有麦克风和摄像头，模拟器也无法使用，因此，上述代码只能够在实际的 Android 终端上运行。

3. 照相

除了录像外，Android 也提供了用于照相的 Camera 类，下述内容完成简单的照相功能。编写布局文件如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:a="http://schemas.android.com/apk/res/android"
    a:orientation="vertical"
    a:layout_width="fill_parent" a:layout_height="fill_parent">
    <SurfaceView
        a:id="@+id/view"
```

```
        a:layout_width="fill_parent" a:layout_height="fill_parent"/>
</LinearLayout>
```

上述文件中，只有一个 SurfaceView 控件，用于在照相时取景，可以显示当前的镜头画面。编写 Activity 代码如下：

```
public class A extends Activity {

    SurfaceView view; // 预览视图
    SurfaceHolder holder;
    Camera camera; // 相机

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // 不显示标题
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        // 全屏
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN);
        // 横屏
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
        setContentView(R.layout.main);

        view = (SurfaceView) findViewById(R.id.view);
        holder = view.getHolder();
        holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
        holder.addCallback(new Callback() {
            @Override
            public void surfaceCreated(SurfaceHolder holder) {
                camera = Camera.open(); // 打开相机
                try {
                    camera.setPreviewDisplay(holder); // 设置预览 View
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

            @Override
            public void surfaceChanged(SurfaceHolder holder, int format,
                    int width, int height) {
                Camera.Parameters params = camera.getParameters();
                params.setPictureFormat(PixelFormat.JPEG); // 图片格式
```

```
            camera.setParameters(params);
            camera.startPreview(); // 开始预览
        }

        @Override
        public void surfaceDestroyed(SurfaceHolder holder) {
            camera.stopPreview(); // 结束预览
            camera.release(); // 释放
        }
    });
}

@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    // 按拍照键或者搜索键进行拍照，有的设备没有拍照键
    if (keyCode == KeyEvent.KEYCODE_CAMERA
            || keyCode == KeyEvent.KEYCODE_SEARCH) {
        camera.stopPreview();
        camera.takePicture(null, null, new PictureCallback() {
            @Override
            public void onPictureTaken(byte[] data, Camera camera) {
                new AsyncTask<byte[], String, String>() {
                    @Override
                    protected String doInBackground(byte[]... params) {
                        File pic = new File("/sdcard/"
                                + new SimpleDateFormat("yyyyMMddHHmmss")
                                        .format(new Date()) + ".jpg");
                        try {
                            // 写入到文件
                            pic.createNewFile();
                            FileOutputStream fos = new FileOutputStream(pic);
                            fos.write(params[0]);
                            fos.close();
                        } catch (IOException e) {
                            e.printStackTrace();
                        }
                        return null;
                    }
                }.execute(data);
                camera.startPreview();
            }
        });
        return true;
```

```
        }
        return super.onKeyDown(keyCode, event);
    }
}
```

上述代码中，定义了相机对象 Camera，取景视图 SurfaceView 和关联的 SurfaceHolder 对象。为 SurfaceHolder 对象注册了 Callback 回调对象，其中在 surface 创建时打开相机，并设置预览视图；在 surface 改变时指定拍照参数、照片格式等，并开始预览；在 surface 销毁时，结束预览，释放相机。为 Activity 注册了按键单击事件，当按下拍照键或搜索键时，进行拍照并写入照片文件。因为程序使用了摄像头和存储卡，因此，需要添加相应的权限，修改 AndroidMenifest.xml 文件，添加权限如下。

- android.permission.WRITE_EXTERNAL_STORAGE：使用外部存储卡的权限；
- android.permission.CAMERA：使用摄像头的权限；

运行项目，如图 5.8 所示。

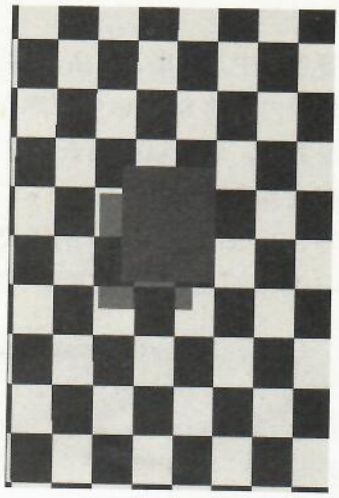

图 5.8

按下拍照键或者搜索键，将会拍摄成照片，并存储在存储卡中。需要注意，由于 Android 模拟器不支持使用摄像头，因此，在模拟器上运行时只能显示图 5.8 所示的黑白方格，而连接实际终端时可显示真实的图像。

拓展练习

练习 5.E.1

结合本章知识拓展 1、2、3，编写一个综合的多媒体程序，完成下列功能：

- 录制音频、视频；
- 回放录制过的音频、视频；
- 照相；
- 浏览拍摄过的照片。