**AsyncTask**

（1）Handler使用比较复杂

（2）Handler和AsyncTask都是为了不阻塞主线程，且UI的更新只能在主线程中完成，异步处理是不可避免的。

（3）AsyncTask的特点是由系统来创建和删除执行任务的线程，并有效的管理它们。

（4）开发者只要创建AsyncTask的子类，实现相应的方法。再创阿金AsyncTask子类的对象，并执行execute()方法就可以了。

AndroidManifest.xml

*<?***xml version="1.0" encoding="utf-8"***?>*<**manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.hjs.asynctaskdemo.asynctask"**>  
  
 <**uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE"**/>  
  
 <**application  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/AppTheme"**>  
 <**activity android:name=".DemoActivity"**>  
 <**intent-filter**>  
 <**action android:name="android.intent.action.MAIN"** />  
  
 <**category android:name="android.intent.category.LAUNCHER"** />  
 </**intent-filter**>  
 </**activity**>  
 </**application**>  
  
</**manifest**>

activity\_main.xml

*<?***xml version="1.0" encoding="utf-8"***?>*<**RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity"**>  
  
  
 <**TextView  
 android:id="@+id/tvTest"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentStart="true"  
 android:layout\_alignParentLeft="true"  
 android:layout\_alignParentTop="true"  
 android:layout\_marginStart="52dp"  
 android:layout\_marginLeft="52dp"  
 android:layout\_marginTop="39dp"  
 android:text="TextView"** />  
  
 <**Button  
 android:id="@+id/button1"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignLeft="@+id/tvTest"  
 android:layout\_alignParentTop="true"  
 android:layout\_marginTop="121dp"  
 android:text="Button"  
 android:onClick="changeNum"**/>  
</**RelativeLayout**>

MainActivity.java

**package** com.hjs.asynctaskdemo.asynctask;  
  
**import** android.os.AsyncTask;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.TextView;  
**import** android.widget.Toast;  
  
**public class** MainActivity **extends** AppCompatActivity {  
 **private** TextView **tvTest**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
  
 **tvTest**=(TextView)findViewById(R.id.***tvTest***);  
 }  
  
 **public void** changeNum(View view){  
 **new** MyTask().execute();*//自动调用 doInBackground* }  
  
 */\*\*  
 \* Params 执行子线程所需传入的参数  
 \* Progress 进度指示时所需的类型  
 \* Result 运行后的结果的类型  
 \*/* **class** MyTask **extends** AsyncTask<Void,Integer,Void>{  
  
 */\*\*  
 \* 该方法运行在后台，相当于是Thread 中run方法  
 \* 该方法是运行在子线程中的方法，控件的显示不能在这里调用，只能作为中间过程处理。  
 \* 最后传值用publishProgress，类似于handler的传值  
 \** ***@param voids*** *\** ***@return*** *\*/* @Override  
 **protected** Void doInBackground(Void... voids) {  
 **for** (**int** i = 0;i<=100;i++){  
 publishProgress(i);*//每一次循环都进行通知，自动调用运行在主线程的 onProgressUpdate* **try** {  
 Thread.*sleep*(1000);  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 **return null**;  
 }  
  
 */\*\*  
 \* 非抽象方法，进度指示的方法  
 \* 该方法运行在主线程中，在这个方法可以直接更新控件的属性  
 \** ***@param values*** *\*/* @Override  
 **protected void** onProgressUpdate(Integer... values) {  
 **int** num = values[0];  
 **tvTest**.setText(**"num:"**+num);*//实时更新主线程中UI控件的属性* **super**.onProgressUpdate(values);  
 }  
  
  
 */\*\*  
 \* 运行在主线程中，当后台运行结束，自动调用的方法  
 \* 当 doInBackground 后台方法运行结束后，线程自动结束，会自动回调这个方法，可用于设置最终结果是什么  
 \** ***@param aVoid*** *\*/* @Override  
 **protected void** onPostExecute(Void aVoid) {  
 Toast.*makeText*(MainActivity.**this**,**"子线程运行结束"**,Toast.***LENGTH\_LONG***).show();  
 **super**.onPostExecute(aVoid);  
 }  
 }  
}

activity\_demo.xml

*<?***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="match\_parent"  
 android:orientation="vertical"**>  
  
 <**Button  
 android:id="@+id/button1"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Read"** />  
  
  
 *<!--进度条-->* <**android.support.v4.widget.ContentLoadingProgressBar  
 android:id="@+id/progressBar1"  
 style="?android:attr/progressBarStyleHorizontal"  
 android:layout\_height="wrap\_content"  
 android:layout\_width="match\_parent"**>  
  
 </**android.support.v4.widget.ContentLoadingProgressBar**>  
  
 <**EditText  
 android:id="@+id/edContent"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:ems="10"  
 android:text="Name"** />  
</**LinearLayout**>

DemoActivity.java

**package** com.hjs.asynctaskdemo.asynctask;  
  
**import** android.Manifest;  
**import** android.app.Activity;  
**import** android.content.pm.PackageManager;  
**import** android.os.AsyncTask;  
**import** android.os.Bundle;  
**import** android.os.Environment;  
**import** android.support.v4.app.ActivityCompat;  
**import** android.support.v4.content.ContextCompat;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.EditText;  
**import** android.widget.ProgressBar;  
  
**import** java.io.BufferedReader;  
**import** java.io.File;  
**import** java.io.FileInputStream;  
**import** java.io.FileNotFoundException;  
**import** java.io.IOException;  
**import** java.io.InputStreamReader;  
  
**public class** DemoActivity **extends** Activity {  
 **private** EditText **etContent**;  
 **private** Button **btnRead**;  
 **private** ProgressBar **pBar**;*//为了当在子线程执行耗时操作时，防止用户以为程序“死掉”，可设置进度条* @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_demo***);  
  
 requestPower();  
  
 **pBar** = (ProgressBar)findViewById(R.id.***progressBar1***);  
 **etContent**=(EditText)findViewById(R.id.***edContent***);  
 **btnRead**=(Button)findViewById(R.id.***button1***);  
 **btnRead**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 *//执行子线程，完成读取文件工作* File file = **new** File(Environment.*getExternalStorageDirectory*(),**"test.txt"**);  
 **new** ReadFileTask().execute(file);  
 }  
 });  
 }  
  
  
 **public void** requestPower() {  
 *//判断是否已经赋予权限* **if** (ContextCompat.*checkSelfPermission*(**this**,  
 Manifest.permission.***READ\_EXTERNAL\_STORAGE***)  
 != PackageManager.***PERMISSION\_GRANTED***) {  
 *//如果应用之前请求过此权限但用户拒绝了请求，此方法将返回 true。即第一次拒绝，就不再请求* **if** (ActivityCompat.*shouldShowRequestPermissionRationale*(**this**,  
 Manifest.permission.***READ\_EXTERNAL\_STORAGE***)) {*//这里可以写个对话框之类的项向用户解释为什么要申请权限，并在对话框的确认键后续再次申请权限.它在用户选择"不再询问"的情况下返回false* } **else** {  
 *//申请权限，字符串数组内是一个或多个要申请的权限，1是申请权限结果的返回参数，在onRequestPermissionsResult可以得知申请结果* ActivityCompat.*requestPermissions*(**this**,  
 **new** String[]{Manifest.permission.***READ\_EXTERNAL\_STORAGE***}, 1);  
 }  
 }  
 }  
  
  
  
 **class** ReadFileTask **extends** AsyncTask<File,Integer,String>{  
  
 @Override  
 **protected** String doInBackground(File...params) {  
 **return** read(params[0]);  
 }  
  
 @Override  
 **protected void** onPostExecute(String result) {  
 **etContent**.setText(result);  
 }  
  
 **private** String read(File file) {  
 StringBuilder sb = **new** StringBuilder();  
 BufferedReader br = **null**;  
  
 *//得到文件长度* **long** fileLength = file.length();  
 *//初始化已读内容长度* **long** readLength = 0;  
  
 **try** {  
 br = **new** BufferedReader(**new** InputStreamReader(**new** FileInputStream(file)));  
 String line = br.readLine();  
 **while**(line != **null**){  
 sb.append(line+**"\n"**);  
 readLength += line.getBytes().**length**;  
 **int** per = (**int**)(readLength\*100.0/file.length());  
 publishProgress(per);  
 **try** {  
 Thread.*sleep*(100);*//模拟耗时 异步线程不会阻塞主线程的操作* } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 line = br.readLine();  
 }  
  
 **return** sb.toString();  
 } **catch** (FileNotFoundException e) {  
 e.printStackTrace();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }**finally** {  
 **if** (br != **null**){  
 **try** {  
 br.close();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 **return null**;  
 }  
  
 @Override  
 **protected void** onProgressUpdate(Integer... values) {  
 **int** per = values[0];  
 **pBar**.setProgress(per);  
 **super**.onProgressUpdate(values);  
 }  
 }  
}

