进度条 ---------->呈显应用执行进度(比如下载,上传,联网等运行进度提示)

1、进度条组件：ProgressBar　　模式情况下为圆形进度条

2、代码动态创建进度条：ProgressBar pro = new ProgressBar(this);

3、设置进度条的最大值：setMax(50);

4、设置进度条的运行任务量：setProgress(u);

5、设置进度条的显示或消失：

setVisibility(View.INVISIBLE)　隐藏,但位置保留，不被别人占用

setVisibility(View.GONE)　移除,位置不保留，被别人占用

如果是代码创建进度条一定要将创建好的进度要添加到布局中：

LinearLayout　LL = new LinearLayout();

LL.setOrientation(LinearLayout.VERTICAL);　//设置布局UI组件排列方向

ll.addView(pb);将进度条添加到布局中去。

通过样式可以将进度条设置为条形的

style="@android:style/Widget.Material.ProgressBar.Horizontal"/>是高版本进度条样式，你的模拟器必须保存在5.0及其以上

style="?android:attr/progressBarStyleHorizontal"低版本中进度条的样式　条形样式，

```
    //设置LinearLayout的方向性
    ll.setOrientation(LinearLayout.VERTICAL);
    //创建进度条
    ProgressBar pb = new ProgressBar(this);
    //添加到LinearLayout中去
    ll.addView(pb);
    //装入集合中
    pbs.add(pb);
}//for结束
for(final ProgressBar pb : pbs){
    //用子线程操作每一个进度条
    new Thread(){
        public void run() {
            //设置进度条的最大值
            pb.setMax(100);
            //假设：每个线程下载100个字节
            for(int i=1;i<=100;i++){
                pb.setProgress(i);
                //休息500毫秒
                Random r = new Random();
                int duration = r.nextInt(500);
                SystemClock.sleep(duration);}
            //用主线程将圆形进度条消失
            MainActivity.this.runOnUiThread(new Thread(){
                public void run() {
                    pb.setVisibility(View.INVISIBLE);       //隐藏,但位置保留，不被别人占用
                    pb.setVisibility(View.GONE);        //移除,位置不保留，被别人占用
```

清单文件 -------->主要作用配置应用运行版本,名称,图标以及注册广播,服务,添加相应权限等

它描述了应用程序中的每个程序组件—Activity，Service，Broadcast Receiver和Content Provider。它描述了实现每个应用程序组件的类名称和组件能力（比如组件能够处理哪种类型的Intent消息）。这些描述帮助Andoid操作系统了解这些程序组件和在何种条件下可以启动这些程序组件。

```
package="cn.itheima"                                  应用唯一标示  包名
android:versionCode="1"                               应用版本号
android:versionName="安全卫士1.0" >                    版本说明
<uses-sdk
    android:minSdkVersion="14"
    android:targetSdkVersion="21" />                  应用运行的系统版本范围
<!-- 应用
    application::icon属性：应用的图标
    android:label属性：应用的标题
 -->
<application
    android:allowBackup="true"
    android:icon="
    android:label=
    android:theme="@style/AppTheme" >

<!-- 第二个界面 -->
<activity
    android:name="cn.itcast.OtherMainActivity"
    android:label="第二个界面"
    android:icon="@drawable/ic_launcher">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

<!-- 第一个界面
 application::name属性：Activity的全路径名,建议初学者不要简写，  写Activity的全路径名
 android:label属性：Activity的的名称
 -->

    <!-- 意图过滤器 -->
    <intent-filter>
        <!-- 动作：android.intent.action.MAIN表示是程序的入口 -->
        <action android:name="android.intent.action.MAIN" />
        <!-- 类别：android.intent.category.LAUNCHER表示在手机中运行，是一个启动器 -->
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
```

类型	含义
android.intent.category.LAUNCHER	启动器
android.intent.category.DEFAULT	默认类型，一般使用这个默认类型
android.intent.category.CAR_DOCK	指定手机被插入汽车底座(硬件)时运行该Activity
android.intent.category.CAR_MODE	设置该Activity可在车载环境下使用

意图------->传递数据,不同页面的跳转,广播,服务之间的转换

只用于A界面->B界面,无法接收B界面的返回值，无人监听

this.startActivity(intent);

当页面跳转后要接收到跳转页面的数据回来时如下操作

A界面：

既用于A界面->B界面,又接收B界面的返回值

参数一：意图

参数二：请求码，是int型，随意写

理解：以下代码执行完后，A界面时时监听B界面的返回结果

只要B界面有返回结果，A立刻回调onActivityResult

this.startActivityForResult(intent,0);

参数一：请求码

参数二：结果码

参数三：意图

重写onActivityResult(int requestCode, int resultCode, Intent data)方法

判断请求码与返回码是否相同，且返回有数据

if(requestCode==0 && resultCode==0 && data!=null){

//从意图中获取数据，

String tel = data.getStringExtra("TEL");

```
public void click(View view){
    Intent intent = new Intent(this,ContactActivity.class);
    //只用于A界面->B界面,无法接收B界面的返回值，无人监听
    //this.startActivity(intent);

    //既用于A界面->B界面,又接收B界面的返回值
    //参数一：意图
    //参数二：请求码，是int型，随意写
    //理解：以下代码执行完后，MainActivity时时监听ContactActivity的返回结果
    //只要ContactActivity有返回结果，MainActivity立刻回调onActivityResult
    this.startActivityForResult(intent,0);
}
/**
 * 在MainActivity中，重写一个方法
 * 参数一：请求码
 * 参数二：结果码
 * 参数三：意图
 */
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if(requestCode==0 && resultCode==0 && data!=null){
        //从意图中获取数据，如188
        String tel = data.getStringExtra("TEL");
```

B界面：

1、创建意图

Intent intent = new Intent();

2、将tel变量绑定到意图中去

intent.putExtra("TEL",tel);

3、将Intent绑定到Result中去

参数一：结果码，int,任意书写

参数二：返回到A界面的Intent对象

本类类名.this.setResult(0,intent);

4、关闭本类界面

本类类名.this.finish();

```
//动态添加单击条目事件
lv_contact.setOnItemClickListener(new OnItemClickListener(){
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {//position=0
        Map<String,String> nameTelMap = list.get(position);
        String tel = nameTelMap.get("tel");
        //创建意图
        Intent intent = new Intent();
        //将tel变量绑定到意图中去
        intent.putExtra("TEL",tel);
        //将Intent绑定到Result中去
        //参数一：结果码，int,任意书写
        //参数二：返回到MainActivity的Intent对象
        ContactActivity.this.setResult(0,intent);
        //关闭ContactActivity界面
        ContactActivity.this.finish();
    }
});
```

通过在B界面的点击事件来返回数据到A界面

通过意图绑定数据返回给A界面

Intent 意图：做一件事情的想法。

Intent包含：动作action(必须)，数据data，有时也会附件类型type

Intent的作用：激活组件和携带参数。

意图设计的目的：解耦，实现应用程序的高内聚、低耦合。保证应用程序之间能够相互独立运行，又能彼此相互调用

解耦：当用一些方法打开手机浏览器的时候，手机上装有多个浏览器时，可能出现系统不知道要打开哪个浏览器，而意图就可以通过动作，数据等来给用户提供选择同时便于系统识别判断运行就相似应用程序

意图分为：

一、显示意图

自己写的类自己知道类名及清单的部署

通过指定应用在Intent中，显示指明了你要跳转的Activity的字节码

总之：显示意图一定是有相应的页面类，相应的布局文件，清单有注册。

通过显示意图跳转页面　类中指明相应的布局，在清单中的活动中指明相应的类全名

方式一

推荐使用：

1、Intent　intent = new Intent(MainActivity.this , 类名.class);

创建意图对象，参数1是上下文，参数2是需要跳转的Activity;

方式二

```
//创建意图　Intent　intent　=　new　Intent();
        //设置意图指向的Activity
参数一：上下文　参数二：跳转到的Activity的字节码
        intent.setClass(this,类名.class);
        //真正跳转发意图指向的Activity
        this.startActivity(intent);
```

方式三

不推荐：(Intent　intent = new　Intent();

intent.setClassName("com.itheima.rpcalc","com.itheima.rpcalc.ResultActivity");　首先创建意图对象，然后给意图对象设置类名，参数1代表包名，参数2代表需要开启的Activity的全路径名；）

2、开启意图：startActivity(intent);

注意参数中的类名:

开发Activity时

01_读需求

02_写xml布局文件，建议名字：activity_login.xml

03_写Activity.java界面文件，建议：LoginActivity.java

04_千万别忘了，注册清单文件

1、这个类名一定是在清单文件中有注册 界面/活动/activity

<activity android:name="cn.itheima.SecondActivity"/> </activity> 并在activity中指明了相关的类。

2、这个类同时具有相应的布局文件，类中指明相应的布局文件：setContentView(R.layout.dierge);

3、在意图中指明要跳转的类

```
//创建意图
Intent intent = new Intent();
//设置意图指向的Activity
//参数一：上下文
//参数二：跳转到的Activity的字节码
//显示意图：在Intent中，显示指明了你要跳转的Activity的字节码
intent.setClass(this,SecondActivity.class);
//真正跳转发意图指向的Activity
this.startActivity(intent);
```

显示意图，清单中只需要注册类名，不用写意图过滤的方法

二、隐式意图

自己写的类自己知道类名及清单的部署

1、先在清单文件中注册活动，活动中指明相应的类，并定义意图过滤器，指明意图的动作及（类别，数据）可选注册项

2、这个类同时具有相应的布局文件，类中指明相应的布局文件：setContentView(R.layout.布局文件名);

3、在意图中指明相应的意图过滤器的动作及（类别，数据）

4、执行页面跳转

```
<!-- 第二个界面/活动/activity -->
<activity
    android:name="cn.itheima.SecondActivity"
    android:label="@string/app_name">

    <intent-filter>

        <!-- 动作， 用自定义 ， 必须-->
        <action android:name="www.baidu.com"/>

        <!-- 类别， 可选-->
        <category android:name="android.intent.category.DEFAULT"/>

        <!-- 数据， 可选 （前缀） -->
        <data android:scheme="tel"/>
```

隐式意图，通过清单中注册的意图过滤器来打开页面

```
public void click(View view){
    /*
    <action android:name="www.baidu.com"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <data android:scheme="tel"/>
    */
    //创建意图
    Intent intent = new Intent();
    //动作，参数是一个字符串
    intent.setAction("www.baidu.com");
    //类别
    intent.addCategory("android.intent.category.DEFAULT");
    //数据
    intent.setData(Uri.parse("tel:110"));
    //跳转到意图指写的Activity
    this.startActivity(intent);
```

隐式意图，自己定义的类 用隐式打开

通过意图过滤器中定义的动作，类别，数据来完成显示意图的页面跳转

代码中必须与清单文件中注册的意图内容相同

三、隐式意图打开第三方应用

完全不知道第三方应用的类名及包名的情况下

```
//获取网络路径
String path = et_path.getText().toString().trim();
//判断
if(!TextUtils.isEmpty(path)){
    /*
    以下采用隐式方式打开第三方应用的Activity(推荐)
    //创建意图
    Intent intent = new Intent();
    //动作，必须
    intent.setAction(Intent.ACTION_VIEW);
    //类别，可选
    intent.addCategory("android.intent.category.DEFAULT");
    //数据，可选
    intent.setData(Uri.parse(path));
    //跳转到意图指写的Activity
    this.startActivity(intent);
    */
```

意图的动作也是必须要有的

隐式意图打开第三方应用：意图的数据是必须要有的

隐式意图

动作：

数据：

类别：

通过意图A界面跳转到B界面，同时携带数据的操作

意图传递的数据类型：

1、8大基本数据类型、数组；

2、Bundle类似于map的数据结构；

3、Serializable序列化到内存；

4、Parcelable序列化到文件；

A界面的操作

传递数据方式一：

直接通过意图的putExtra("MARK",mark);方法携带数据传递给B界面

参数一：字符串,建议大写　　参数二：需要携带的数据，数据类型不限

```
Intent intent = new Intent(this,SecondActivity.class);
intent.putExtra("MARK",mark);//int
intent.putExtra("USERNAME",username);//String
intent.putExtra("SEX",sex);//String
this.startActivity(intent);
```

B界面的接受处理：

1、获取意图getIntent()

```
Intent intent = this.getIntent();
```

2、从Intent中取三个变量的值

参数一：原来绑定到Intent对象中的KEY　　参数二：如果在Intent对象中根据KEY找不到对应的值时，返回的默认值

```
int mark = intent.getIntExtra("MARK",0);
String username = intent.getStringExtra("USERNAME");
String sex = intent.getStringExtra("SEX");
```

3、将获取的数据再处理.....

```
//*****************************************************************传递数据方式一
//A界面跳转到B界面，同时携带数据
//Intent intent = new Intent(this,SecondActivity.class);
//将mark变量绑定到Intent对象中去
//参数一：字符串,建议大写
//参数二：需要携带的数据，数据类型不限        常规方法
//intent.putExtra("MARK",mark);//int
//intent.putExtra("USERNAME",username);//String
//intent.putExtra("SEX",sex);//String
//this.startActivity(intent);
```

```
//获取FirstActivity传递过来的Intent对象
//Intent intent = this.getIntent();                常规方法对应的获取数据
//从Intent中取三个变量的值
//参数一：原来绑定到Intent对象中的KEY
//参数二：如果在Intent对象中根据KEY找不到对应的值时，返回的默认值
//int mark = intent.getIntExtra("MARK",0);
//String username = intent.getStringExtra("USERNAME");
//String sex = intent.getStringExtra("SEX");
```

传递数据方式二：

通过Bundle类的对象的putInt( ) 方法或者putString()方法添加数据，将Bundle类的对象传递给B界面

创建Bunlde这个Map对象

```
Bundle bundle = new Bundle();
bundle.putInt("MARK",mark);
bundle.putString("USERNAME",username);
bundle.putString("SEX",sex);
Intent intent = new Intent(this,SecondActivity.class);
```

将Bundle这个Map对象绑定到Intent对象中

```
intent.putExtras(bundle);
this.startActivity(intent);
```

B界面的接受处理：

1、获取意图getIntent()

Intent intent = this.getIntent();

2、获取Bundle对象

Bundle bundle = intent.getExtras();

3、通过Bundle对象的get方法获取数据

int mark = bundle.getInt("MARK");

String username = bundle.getString("USERNAME");

String sex = bundle.getString("SEX");

4、将获取的数据再处理.....

```
//*****************************************************************传递数据方式二
//创建Bunlde这个Map对象
//Bundle bundle = new Bundle();
//bundle.putInt("MARK",mark);
//bundle.putString("USERNAME",username);
//bundle.putString("SEX",sex);
//Intent intent = new Intent(this,SecondActivity.class);
//将Bundle这个Map对象绑定到Intent对象中
//intent.putExtras(bundle);
//this.startActivity(intent);
```

通过Bundle类的方法绑定数据

```
//Intent intent = this.getIntent();
//Bundle bundle = intent.getExtras();
//int mark = bundle.getInt("MARK");
//String username = bundle.getString("USERNAME");
//String sex = bundle.getString("SEX");
```

通过Bunle获取

传递数据方式三：

通过创建JavaBean类实现implements java.io.Serializable给出数据的

创建JavaBean

User user = new User(username,sex,mark);

注意如果是无参构造那么需要先给User进行set方法赋值.然后才可以绑定传数据

user.setName();

user.setName();

user.setName();.....

创建Bundle

Bundle bundle = new Bundle();

将JavaBean绑定到Bunlde对象中，参数二：实现Serializable接口的JavaBean

bundle.putSerializable("USER",user);

创建意图

Intent intent = new Intent(this,SecondActivity.class);

将Bundle对象绑定到Intent对象中

intent.putExtras(bundle);

跳转到意图指定的Activity

this.startActivity(intent);

B界面的接受处理：

1、获取意图getIntent();

Intent intent = this.getIntent();

2、获取Bundle对象

Bundle bundle = intent.getExtras();

3、通过获取序列化接口的JavaBean对象

User user = (User) bundle.getSerializable("USER");

4、通过对象中的get方法获取数据

String username = user.getUsername();

String sex = user.getSex();

int mark = user.getMark();

5、将获取的数据再处理.....

```
//**********************************************************************传递数据方式三
//创建JavaBean
User user = new User(username,sex,mark);
//创建Bundle
Bundle bundle = new Bundle();
//将JavaBean绑定到Bunlde对象中，参数二：实现Serializable接口的JavaBean
bundle.putSerializable("USER",user);
//创建意图
Intent intent = new Intent(this,SecondActivity.class);
//将Bundle对象绑定到Intent对象中
intent.putExtras(bundle);
//跳转到意图指定的Activity
this.startActivity(intent);
```

JAVABin类设置数据
将这个类绑定到Bundle对象

序列化绑定数据

携带Bundle对象到B界面

```
Intent intent = this.getIntent();
Bundle bundle = intent.getExtras();
User user = (User) bundle.getSerializable("USER");
String username = user.getUsername();
String sex = user.getSex();
int mark = user.getMark();
```

获取序列化对象

对象的get方法获取数据

```
public class User implements java.io.Serializable{
    private String username;//姓名
    private String sex;//性别
    private int mark;//分数
    public User(){}
    public User(String username, String sex,int mark) {
        this.username = username;
        this.sex = sex;
        this.mark = mark;
    }
    public String getUsername() {
        return username;
```

实现序列化接口

构造决定了设置数据是否要用set方法

URI和URL介绍及区别

URI统一资源标识符(Uniform Resource Identifier)，用来表示某一互联网资源名称的字符串。

URL统一资源定位符(Uniform Resource Locator)，是可以从互联网上得到资源的位置和访问方法的一种简洁的表示，是互联网上标准资源的访问地址

URI主要分三个部分：scheme，authority，path。

其中authority又分为host[必写]和port[可选]。

格式如下：

scheme://host:port/path