# Edit检测内容变化

edit1.addTextChangedListener(**new** TextWatcher() {  
 @Override  
 **public void** beforeTextChanged(CharSequence s, **int** start, **int** count, **int** after) {  
 Log.*d*(**"TAG"**, **"beforeTextChanged: "**+s+**" "**+start+**" "**+count+**" "**+after);  
 *//这个方法被调用，说明在s字符串中，从start位置开始的count个字符即将被长度为after的新文本所取代。* }  
  
 @Override  
 **public void** onTextChanged(CharSequence s, **int** start, **int** before, **int** count) {  
 Log.*d*(**"TAG"**, **"onTextChanged: "**+s+**" "**+start+**" "**+count+**" "**+before);  
 *//这个方法被调用，说明在s字符串中，从start位置开始的count个字符刚刚取代了长度为before的旧文本。  
 //s是改变之后的* }  
  
 @Override  
 **public void** afterTextChanged(Editable s) {  
 Log.*d*(**"TAG"**, **"afterTextChanged: "**+s.toString());  
 }  
});

# Sensor传感器

**public class** MainActivity **extends** AppCompatActivity **implements** SensorListener {  
 **private** TextView **txt1**;  
 **private** TextView **txt2**;  
 **private** TextView **txt3**;  
  
 SensorManager **sm**;  
  
 ImageView **imgBlock**;  
  
 RelativeLayout **rLayout**;  
 **double x\_2** = 0.0;  
 **double y\_2** = 0.0;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 *//* **txt1** = findViewById(R.id.***txt1***);  
 **txt2** = findViewById(R.id.***txt2***);  
 **txt3** = findViewById(R.id.***txt3***);  
 *//* **imgBlock** = findViewById(R.id.***imgBlock***);  
 *//* **rLayout** = findViewById(R.id.***rLayout***);  
 }  
  
 @Override  
 **protected void** onResume() {  
 **super**.onResume();  
 *//初始化传感器* **sm** = (SensorManager) getSystemService(***SENSOR\_SERVICE***);  
 Sensor s = **sm**.getDefaultSensor(Sensor.***TYPE\_ACCELEROMETER***);  
 **sm**.registerListener(**this**,SensorManager.***SENSOR\_DELAY\_NORMAL***);  
  
 }  
  
 @Override  
 **public void** onSensorChanged(**int** sensor, **float**[] values) {  
 *//显示传感器数据* **txt1**.setText(**"X : "**+values[0]);  
 **txt2**.setText(**"Y : "**+values[1]);  
 **txt3**.setText(**"Z : "**+values[2]);  
 *//（容器尺寸-方块尺寸）/2* **if**(**x\_2**==0){  
 **x\_2** = (**rLayout**.getWidth()-**imgBlock**.getWidth())/2;  
 **y\_2** = (**rLayout**.getHeight()-**imgBlock**.getHeight())/2;  
 }  
 *//移动imgBlock* RelativeLayout.LayoutParams blockParam =  
 (RelativeLayout.LayoutParams)**imgBlock**.getLayoutParams();  
 blockParam.**leftMargin** = (**int**)(**x\_2** + values[0]\*20);  
 blockParam.**topMargin** = (**int**)(**y\_2** + values[1]\*20);  
 **imgBlock**.setLayoutParams(blockParam);  
 }  
  
 @Override  
 **public void** onAccuracyChanged(**int** sensor, **int** accuracy) {  
 *//暂不处理* }  
  
 @Override  
 **protected void** onPause() {  
 **super**.onPause();  
 *//注销传感器* **sm**.unregisterListener(**this**);  
 }  
}

# Intent 解析

当系统收到一个启动 Activity 的隐式 Intent 后，将会查找最合适的 Activity ， 这通过比较 Intent 和 Intent 过滤器的三个部分来实现：

* Intent 的 Action
* Intent 的数据（包括 URI 和 数据类型）
* Intent 的类型

以下章节说明了 Intent 与组件的匹配方式，基于应用程序的 Manifest 文件中声明的 Intent 过滤器。

### Action 验证

为了指定可接受的 Intent Action，Intent 过滤器可以声明0个以上的 [<action>](http://developer.android.com/guide/topics/manifest/action-element.html) 元素。例如：

<intent-filter>

<action android:name="android.intent.action.EDIT" />

<action android:name="android.intent.action.VIEW" />

...

</intent-filter>

为了通过这个过滤器的验证， [Intent](http://developer.android.com/reference/android/content/Intent.html) 中指定的 Action 必须匹配上述过滤器中的一种 Action。

如果过滤器没有给出任何 Action ， Intent 就没有可匹配的目标，因此所有的验证都会失败。 但是，如果是 [Intent](http://developer.android.com/reference/android/content/Intent.html) 未给出 Action ，则它会通过验证（假设过滤器中至少包含了一个 Action）。

### Category 验证

为了指定可接收的 Intent 类型， Intent 过滤器可以声明0个以上的 [<category>](http://developer.android.com/guide/topics/manifest/category-element.html) 元素。例如：

<intent-filter>

<category android:name="android.intent.category.DEFAULT" />

<category android:name="android.intent.category.BROWSABLE" />

...

</intent-filter>

为了通过类型验证， [Intent](http://developer.android.com/reference/android/content/Intent.html) 中的每个 Category 都必须与过滤器中的相匹配。 反之则不然 — Intent 过滤器中声明的 Category 可以多于 [Intent](http://developer.android.com/reference/android/content/Intent.html) 中的，验证将会通过。 因此，不带任何 Category 的 Intent 也将会通过验证，不管过滤器中的 Category 怎么声明都会通过。

**注意：** Android 会对传入 [startActivity()](http://developer.android.com/reference/android/content/Context.html#startActivity(android.content.Intent)) 和 [startActivityForResult()](http://developer.android.com/reference/android/app/Activity.html#startActivityForResult(android.content.Intent,%20int)) 的隐式 Intent 自动应用 [CATEGORY\_DEFAULT](http://developer.android.com/reference/android/content/Intent.html#CATEGORY_DEFAULT) 类型。 因此，如果 Activity 需要接收隐式 Intent ，它就必须在 Intent 过滤器中包含 "android.intent.category.DEFAULT" （正如上述<intent-filter> 所示）。

### Data 验证

为了指定可接收的 Intent 数据， Intent 过滤器可以声明0个以上的 [<data>](http://developer.android.com/guide/topics/manifest/data-element.html) 元素。例如：

1 <intent-filter>

2 <data android:mimeType="video/mpeg" android:scheme="http" ... />

3 <data android:mimeType="audio/mpeg" android:scheme="http" ... />

4 ...

5 </intent-filter>

每个 [<data>](http://developer.android.com/guide/topics/manifest/data-element.html) 元素都可以指定一个 URI 和一个数据类型（MIME 媒体类型）。 URI 的每个部分都对应有单独的属性 — scheme 、host 、 port 和 path：

<scheme>://<host>:<port>/<path>

例如：

content://com.example.project:200/folder/subfolder/etc

在上述 URI 中， Scheme 是 content ， Host 是 com.example.project ， Port 是 200 ， Path 是 folder/subfolder/etc 。

这些属性都是 [<data>](http://developer.android.com/guide/topics/manifest/data-element.html) 元素的可选项，但却有顺序的依赖关系：

* 如果未给出 Scheme ，则 Host 将被忽略。
* 如果未给出 Host ，则 Port 将被忽略。
* 如果 Scheme 和 Host 均未给出，则 Path 将被忽略。

在把 Intent 中的 URI 和过滤器中的 URI 规则进行比较时，只会按照过滤器 URI 的各个部分来进行。 例如：

* 如果过滤器只声明了 Scheme ，则所有带有此 Scheme 的 URI 都将是与其匹配的。
* 如果过滤器声明了 Scheme 和 Authority ，但没有声明 Path ， 则所有带有同样 Scheme 和 Authority 的 URI 都会通过该过滤器， Path 怎么样都无所谓。
* 如果过滤器同时声明了 Scheme 、 Authority 和 Path ， 则只有那些 Scheme 、 Authority 和 Path 都符合的 URI 才能通过该过滤器。

**注意：** Path 规则中可以包含通配符（\*），这样只需要匹配部分路径名称即可。

数据验证将会同时比较 Intent 和过滤器的 URI 和 MIME 类型。 规则如下：

1. 如果 Intent 即不包含 URI ，也不含 MIME 类型， 则仅当过滤器也未指定任何 URI 和 MIME 类型时，才能通过验证。
2. 如果 Intent 包含了 URI ，但不含 MIME 类型（既没有显式给出也无法从 URI 中推导出来）， 则仅当 URI 与过滤器的 URI 规则吻合且过滤器未指定 MIME 类型时，才能通过验证。
3. 如果 Intent 包含了 MIME 类型，但不含 URI ， 则仅当过滤器中列出了相同的 MIME 类型且未指定 URI 规则时，才能通过验证。
4. 如果 Intent 同时包含了 URI 和 MIME 类型（显式给出或可从 URI 中推导出来）， 则当过滤器中列出了相同的 MIME 类型时通过 MIME 类型验证。 当 URI 与过滤器中的 URI 相匹配，或者 URI 带有 content: 或 file:且过滤器未指定 URI 时，通过 URI 验证。 也就是说，如果某个组件的 Intent 过滤器只给出了 MIME 类型， 那么就表示它将支持 content: 和 file: 数据的处理。

最后一条规则说明，系统期望所有组件都能够读取文件或者 Content Provider 中的本地数据。 因此，过滤器只需要列出数据类型即可，不必显式地命名 content: 和 file: 类型的 Scheme。 这是一种典型的应用场景。 例如，下述的 [<data>](http://developer.android.com/guide/topics/manifest/data-element.html) 元素将会通知 Android 系统，该组件可以从 Content Provider 读取图片数据并显示出来：

<intent-filter>

<data android:mimeType="image/\*" />

...

</intent-filter>

因为大部分数据都是由 Content Provider 发布出来的， 所以指定数据类型但不给出 URI 也许是 Intent 过滤器最常见的配置方式。

Intent 过滤器另一种常见的配置方式就是包含 Scheme 和数据类型。 例如，类似下述的 [<data>](http://developer.android.com/guide/topics/manifest/data-element.html) 元素将会通知 Andorid 系统，为了执行 Action ，该组件可以从网络读取视频数据：

<intent-filter>

<data android:scheme="http" android:type="video/\*" />

...

</intent-filter>

### Intent 匹配

Intent 与 过滤器的匹配机制，不仅用于查找需要激活的组件，还用于发现当前设备中所有组件的某些信息。 例如，通过把所有带有 [ACTION\_MAIN](http://developer.android.com/reference/android/content/Intent.html#ACTION_MAIN) Action 和 [CATEGORY\_LAUNCHER](http://developer.android.com/reference/android/content/Intent.html#CATEGORY_LAUNCHER) Category 过滤器的 Activity 找出来，管理主屏幕的应用程序就可以组建应用程序启动器。

应用程序可以采用统一的方式来使用 Intent 过滤机制。 [PackageManager](http://developer.android.com/reference/android/content/pm/PackageManager.html) 拥有一些 query...() 方法， 这些方法将会返回能够接受某类 Intent 的全部组件， 类似的另一些 resolve...() 方法将会确定响应某个 Intent 的最佳组件。例如， [queryIntentActivities()](http://developer.android.com/reference/android/content/pm/PackageManager.html#queryIntentActivities(android.content.Intent,%20int)) 将返回可执行传入参数 Intent 的 Activity 列表， [queryIntentServices()](http://developer.android.com/reference/android/content/pm/PackageManager.html#queryIntentServices(android.content.Intent,%20int)) 将返回类似的服务列表。 这两个方法均不会激活组件，而只是列出能够响应 Intent 的组件。 另一个类似的方法，就是列出广播接收器列表的 [queryBroadcastReceivers()](http://developer.android.com/reference/android/content/pm/PackageManager.html#queryBroadcastReceivers(android.content.Intent,%20int))。

**public class** Scene1Activity **extends** AppCompatActivity {  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***scene1\_layout***);  
  
 *//恢复临时数据-旋转屏幕后恢复文件* **i****f**(savedInstanceState != **null**){  
 TextView txt1 = findViewById(R.id.***txt1***);  
 txt1.setText(savedInstanceState.getString(**"txt1"**));  
 }  
 ActionBar actionBar = getSupportActionBar();  
 **if**(actionBar != **null**){ 隐藏ActionBar标题框  
 actionBar.hide();  
 }  
 Button btn1 = findViewById(R.id.***btn1***);  
 Button btn2 = findViewById(R.id.***btn2***);

按钮一功能  
 View.OnClickListener onClickListener = **new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 Intent intent = **new** Intent(Scene1Activity.**this**,Scene2Activity.**class**);

//从一个activity到另一个activity  
 intent.putExtra(**"hello"**,**"来自Scene1Activity的问候:)"**); //添加附加数据  
 startActivityForResult(intent,1); //执行有返回值的intent  
 }  
 };  
 btn1.setOnClickListener(onClickListener);

按钮二功能  
 btn2.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 Intent intent = **new** Intent(**"com.example.activitydemo.ACTION\_START"**);//action  
 intent.addCategory(**"com.example.activitydemo.Scene2"**);  
 startActivity(intent);  
 }  
 });  
 }  
  
 @Override  
 **protected void** onActivityResult(**int** requestCode, **int** resultCode, @Nullable Intent data) {  
 对activity返回的数据处理  
 **if** (requestCode == 1) {  
 TextView txt1 = findViewById(R.id.***txt1***);  
 **if** (resultCode == ***RESULT\_OK***) {  
 String s = data.getStringExtra(**"btn\_msg"**);  
 txt1.setText(s);  
  
 } **else if** (resultCode == ***RESULT\_CANCELED***) {  
 String s = data.getStringExtra(**"back\_msg"**);  
 txt1.setText(s);  
  
 }  
 }  
 **super**.onActivityResult(requestCode, resultCode, data);  
  
 }

onSaveInstanceState（）被系统杀死前保存数据

*//因为activity的销毁是因为它所在的进程被杀而造成的。  
// onSaveInstanceState()是在系统感觉需要杀死activity时调用的，  
// 它被传入一个参数：Bundle，这个Bundle可以被认为是个map,字典之类的东西，用”键－值”来保存数据。*

@Override  
 **protected void** onSaveInstanceState(@NonNull Bundle outState) {  
 **super**.onSaveInstanceState(outState);  
 TextView txt1 = findViewById(R.id.***txt1***);  
 String s = txt1.getText().toString();  
 outState.putString(**"txt1"**,s);  
  
 }  
}

另一个activity：

通过按钮返回数据

Intent intent = getIntent(); 得到附加的intent内容  
String s = intent.getStringExtra(**"hello"**);  
txt2.setText(s);

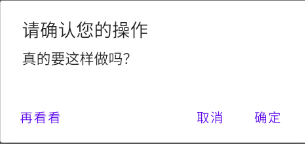
btn2.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 Intent intent = **new** Intent();  
 intent.putExtra(**"btn\_msg"**,**"单击按钮"**);  
 setResult(***RESULT\_OK***,intent);  
 finish();  
  
 }  
});

通过返回键返回数据

@Override  
**public void** onBackPressed() {  
 Intent intent = **new** Intent();  
 intent.putExtra(**"back\_msg"**,**"单机返回键"**);  
 setResult(***RESULT\_CANCELED***,intent);  
 **super**.onBackPressed();  
  
}

# AlertDialog

AlertDialog.Builder dlg = **new** AlertDialog.Builder(MainActivity.**this**);  
dlg.setTitle(**"请确认您的操作"**);  
dlg.setMessage(**"真的要这样做吗？"**);  
dlg.setCancelable(**false**); *//点击空白处不可以取消  
//肯定操作按钮*dlg.setPositiveButton(**"确定"**, **new** DialogInterface.OnClickListener() {  
 @Override  
 **public void** onClick(DialogInterface dialog, **int** which) {  
 Toast.*makeText*(MainActivity.**this**,**"操作确认"**,Toast.***LENGTH\_SHORT***).show();  
  
 }  
});  
*//取消操作按钮*dlg.setNegativeButton(**"取消"**, **new** DialogInterface.OnClickListener() {  
 @Override  
 **public void** onClick(DialogInterface dialog, **int** which) {  
 Toast.*makeText*(MainActivity.**this**,**"操作取消"**,Toast.***LENGTH\_SHORT***).show();  
 }  
});  
dlg.setNeutralButton(**"再看看"**, **new** DialogInterface.OnClickListener() {  
 @Override  
 **public void** onClick(DialogInterface dialog, **int** which) {  
  
 }  
});  
  
dlg.show();



# Bitmap-图片操作

**public void** onClick(View v) {  
 **int** vid = v.getId();  
 **if** (vid == R.id.***btn1***){  
 *//button单击响应  
  
 //\*\*\*\*缩放\*\*\*\** Bitmap bmp = BitmapFactory.*decodeResource*(getResources(),R.drawable.***result***);  
 */\*\*  
 \* bug 图片加载不出来 解决：换张图片，不同系统自带的  
 \*/  
  
 //参数： 1指定当前应用的资源 2指定资源ID* Matrix matrix = **new** Matrix();  
 matrix.setScale(10f,10f);  
*// 设置宽高比 会重置矩阵数据，其他操作会撤销* Bitmap bmp1 = Bitmap.*createBitmap*(bmp,0,0,bmp.getWidth(),bmp.getHeight(),matrix,**false**);  
 *//参数 1源文件 2.3复制源文件对象的左上角坐标 4.5 复制源文件的宽高 6指定变换数据（Matrix对象）  
 // 7是否使用过滤器* **img1**.setImageBitmap(bmp1);  
 }**else if**(vid == R.id.***btn2***){  
 *//\*\*\*\*选转\*\*\*\** Bitmap bmp = BitmapFactory.*decodeResource*(getResources(),R.drawable.***result***);  
  
 Matrix matrix = **new** Matrix();  
 matrix.preScale(10f,10f);  
 matrix.preRotate(90f); *//选转90° 顺时针为正，逆时针为负  
 //preScale将操作预处理 ，统一返回操作结果，这样可以同时进行多个操作* Bitmap bmp1 = Bitmap.*createBitmap*(bmp,0,0,bmp.getWidth(),bmp.getHeight(),matrix,**false**);  
 **img1**.setImageBitmap(bmp1);  
 }**else if**(vid == R.id.***btn3***){  
 *//\*\*\*\*扭曲\*\*\*\** Bitmap bmp = BitmapFactory.*decodeResource*(getResources(),R.drawable.***result***);  
  
 Matrix matrix = **new** Matrix();  
 matrix.preScale(10f,10f);  
 matrix.preSkew(1f,0.2f); *//扭曲 参数： 1水平 2竖直 不能太大* Bitmap bmp1 = Bitmap.*createBitmap*(bmp,0,0,bmp.getWidth(),bmp.getHeight(),matrix,**false**);  
 **img1**.setImageBitmap(bmp1);  
 }  
 }

# 按钮绑定事件

findViewById(R.id.***btn1***).setOnLongClickListener(**this**::onLongClick); 绑定长按事件

**public boolean** onLongClick(View v){  
 **if** (v.getId() == R.id.***btn1***){  
 Toast.*makeText*(MainActivity.**this**,**"长按事件"**, Toast.***LENGTH\_SHORT***).show();  
 *//震动提示* Vibrator vib = (Vibrator) getSystemService(getApplicationContext().***VIBRATOR\_SERVICE***);  
 vib.vibrate(500); *//震动半秒* }  
 **return true**;  
}

绑定触摸事件

findViewById(R.id.***btn2***).setOnTouchListener(**new** View.OnTouchListener() {  
 @Override  
 **public boolean** onTouch(View v, MotionEvent event) {  
 TextView txt = findViewById(R.id.***txt1***);  
 **if**(v.getId() == R.id.***btn2***){  
 **if**(event.getAction() == MotionEvent.***ACTION\_DOWN***){  
 txt.setText(**"手指按下"**+event.getX()+**" "**+event.getY());  
 }**else if**(event.getAction() == MotionEvent.***ACTION\_UP***){  
 txt.setText(**"手指离开"**+event.getX()+**" "**+event.getY());  
 }**else if** (event.getAction() == MotionEvent.***ACTION\_MOVE***){  
 txt.setText(**"手指滑动"**+event.getX()+**" "**+event.getY());  
 }  
 }  
 **return true**;  
 }  
});

# CheckBox

check1.setOnCheckedChangeListener(**new** CompoundButton.OnCheckedChangeListener() {  
 @Override  
 **public void** onCheckedChanged(CompoundButton buttonView, **boolean** isChecked) {  
 TextView txt1 = findViewById(R.id.***txt1***);  
 String msg = buttonView.getText().toString();  
 **if**(isChecked){  
 msg += **" 选中"**;  
  
 }**else**{  
 msg += **" 未选中"**;  
 }  
 txt1.setText(msg);  
 }  
});

# DateTimeDialog

**public class** MainActivity **extends** AppCompatActivity  
 **implements** View.OnClickListener,  
 DatePickerDialog.OnDateSetListener,  
 TimePickerDialog.OnTimeSetListener {  
Calendar **calendar** = Calendar.*getInstance*(TimeZone.*getTimeZone*(**"GMT+8"**));  
 *//日历类，获得东八区的当前系统时间* @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 }

@Override  
 **public void** onClick(View v) {  
 **int** vid = v.getId();  
 **if** (vid == R.id.***txtDate***){  
 DatePickerDialog dlg = **new** DatePickerDialog(context:**this**,  
 listener:**this**, *//监听对象 DatePickerDialog.OnDateSetListener* **calendar**.get(Calendar.***YEAR***), *//设置默认年* **calendar**.get(Calendar.***MONTH***),  
 **calendar**.get(Calendar.***DAY\_OF\_MONTH***));  
 dlg.show();  
 }**else if** (vid == R.id.***txtTime***){  
 TimePickerDialog dlg = **new** TimePickerDialog(**this**,**this**,  
 **calendar**.get(Calendar.***HOUR\_OF\_DAY***),  
 **calendar**.get(Calendar.***MINUTE***),  
 **true**); *//是否是24小时制* dlg.show();  
 }  
 }  
  
 @Override  
 **public void** onDateSet(DatePicker view, **int** year, **int** month, **int** dayOfMonth) {  
 **txtDate**.setText(  
 String.*format*(**"选择的日期：%d年%d月%d日"**,year,month,dayOfMonth)  
 );  
 }  
  
 @Override  
 **public void** onTimeSet(TimePicker view, **int** hourOfDay, **int** minute) {  
 **txtTime**.setText(  
 String.*format*(**"选择的时间： %d时%d分"**,hourOfDay,minute)  
 );  
  
 }  
}

# ListView

**public class** MainActivity **extends** AppCompatActivity **implements** AdapterView.OnItemClickListener {  
 *//列表显示的数据* **private** String[] **data** = {**"aaa"**,**"bbb"**,**"ccc"**,**"ddd"**,**"eee"**,**"fff"**,**"ggg"**,**"hhh"**};  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 *//初始化列表* ArrayAdapter<String> ada = **new** ArrayAdapter<String>(MainActivity.**this**,android.R.layout.***simple\_list\_item\_1***,**data**);  
 *//* ListView lst1 = findViewById(R.id.***lst1***);  
 lst1.setAdapter(ada);  
 *//响应列表单击* lst1.setOnItemClickListener(**this**);  
 *//添加列表注脚* TextView footer = **new** TextView(**this**);  
 footer.setText(**"ListView Footer"**);  
 lst1.addFooterView(footer);  
 *//添加列表标题* TextView header = **new** TextView(**this**);  
 header.setText(**"ListView Header"**);  
 lst1.addHeaderView(header);  
  
 }  
  
 @Override  
 **public void** onItemClick(AdapterView<?> parent, View view, **int** position, **long** id) {  
 **if**(id>=0){ *//当其为-1时，为标题或注脚* String s = **data**[(**int**)id];  
 Toast.*makeText*(MainActivity.**this**,s,Toast.***LENGTH\_SHORT***).show();  
 }  
 }  
}

# LocalBroadcast

. **public class** MainActivity **extends** AppCompatActivity {  
 **private** MyReceiver **myReceiver**;  
 **private** LocalBroadcastManager **localBroadcastManager**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 *//* **myReceiver** = **new** MyReceiver();  
 IntentFilter intentFilter = **new** IntentFilter(**"LocalBroadcastTest"**);  
 **localBroadcastManager** = LocalBroadcastManager.*getInstance*(**this**);  
 **localBroadcastManager**.registerReceiver(**myReceiver**,intentFilter);  
 *//发送广播* Button button = findViewById(R.id.***btn***);  
 button.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 Intent intent = **new** Intent(**"LocalBroadcastTest"**);  
 **localBroadcastManager**.sendBroadcast(intent);  
  
 }  
 });  
 }  
  
 @Override  
 **protected void** onDestroy() {  
 **localBroadcastManager**.unregisterReceiver(**myReceiver**);  
 **localBroadcastManager** = **null**;  
 **super**.onDestroy();  
 }  
 接收广播  
 **public class** MyReceiver **extends** BroadcastReceiver{  
  
 @Override  
 **public void** onReceive(Context context, Intent intent) {  
 Toast.*makeText*(context,**"接收广播"**,Toast.***LENGTH\_SHORT***).show();  
 }  
 }  
}

# LoginView-自定义组件

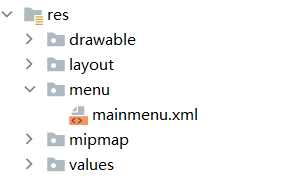
自定义组件类

**public class** LoginView **extends** android.widget.LinearLayout {  
 **private boolean loginResult** = **false**;  
  
 **public** LoginView(Context context, @Nullable AttributeSet attrs) {  
 **super**(context, attrs);  
 LayoutInflater.*from*(context).inflate(R.layout.***login\_layout***,**this**);  
 *//第二个参数 把第一个login\_layout加载到this(一个LinearLayout --ViewGroup) 可以为null  
  
 //登陆操作* Button btnLogin = findViewById(R.id.***btnLogin***);  
 btnLogin.setOnClickListener(**new** OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 *//登陆操作* **loginResult** = **true**;  
 *//* Toast.*makeText*(context,**"执行登陆操作"**,Toast.***LENGTH\_SHORT***).show();;  
  
 }  
 });  
  
 }  
  
 **public boolean** getLoginResult(){  
 **return loginResult**;  
 }  
}

使用自定义组件

LoginView loginView = findViewById(R.id.***loginView***); *//获取对象不是创建*String msg = **"登陆失败"**;  
**if** (loginView.getLoginResult()){  
 msg = **"登陆成功"**;  
}  
Toast.*makeText*(MainActivity.**this**,msg,Toast.***LENGTH\_SHORT***).show();

# Menu



*<?***xml version="1.0" encoding="utf-8"***?>*<**menu xmlns:android="http://schemas.android.com/apk/res/android"**>  
 <**item android:id="@+id/menuitem1"  
 android:title=" 菜单项1"**></**item**>  
 <**item android:id="@+id/menuitem2"  
 android:title=" 菜单项2"**></**item**>  
 <**item android:id="@+id/menuitem3"  
 android:title=" 菜单项3"**></**item**>  
</**menu**>

@Override  
**public boolean** onCreateOptionsMenu(Menu menu) {  
 MenuInflater mi = getMenuInflater();  
 mi.inflate(R.menu.***mainmenu***,menu);  
 *//资源文件 被填进的menu* **return true**;  
  
}  
*//响应菜单项*@Override  
**public boolean** onOptionsItemSelected(@NonNull MenuItem item) {  
 **switch** (item.getItemId()){  
 **case** R.id.***menuitem1***:{  
 Toast.*makeText*(MainActivity.**this**,**"菜单项1"**,Toast.***LENGTH\_SHORT***).show();  
 }**break**;  
 **case** R.id.***menuitem2***:{  
 Toast.*makeText*(MainActivity.**this**,**"菜单项2"**,Toast.***LENGTH\_SHORT***).show();  
 }**break**;  
 **case** R.id.***menuitem3***:{  
 Toast.*makeText*(MainActivity.**this**,**"菜单项3"**,Toast.***LENGTH\_SHORT***).show();  
 }**break**;  
  
 }  
 **return true**;  
}

# Notification

*//1.创建通知管理器  
//获取相应系统服务对象 用于发送或取消通知*NotificationManager manager = (NotificationManager) getSystemService(***NOTIFICATION\_SERVICE***);*//获取NotificationManager实例*manager.cancel(1); *//取消前一个通知 id：通知标识ID  
  
//2.创建通知渠道***if** (Build.VERSION.***SDK\_INT*** >= 26) {*//判断Android的版本* NotificationChannel channel = **new** NotificationChannel(String.*valueOf*(1), **"name"**,  
 NotificationManager.***IMPORTANCE\_HIGH***); *//当Android版本大于等于8时，创建通知渠道（Notification Channels）* manager.createNotificationChannel(channel);  
}  
*//3.创建通知建立对象*NotificationCompat.Builder builder;  
builder = **new** NotificationCompat.Builder(MainActivity.**this**, String.*valueOf*(1));  
  
builder.setContentTitle(**"牛皮哄哄"**)*//通知标题* .setContentText(**"精神小伙"**) *//通知内容* .setWhen(System.*currentTimeMillis*()) *//指定通知被创建的时间* .setSmallIcon(R.mipmap.***ic\_launcher***) *//使用小图标* .setLargeIcon(BitmapFactory.*decodeResource*(getResources(), R.mipmap.***ic\_launcher***)); *//使用大图片*Intent intent = **new** Intent(**this**,MainActivity.**class**);  
*//PendingIntent是对Intent的封装，但它不是立刻执行某个行为，而是满足某些条件或触发某些事件后才执行指定的行为  
//PendingIntent将某个动作的触发时机交给其他应用；让那个应用代表自己去执行那个动作（权限都给他）*PendingIntent pi = PendingIntent.*getActivity*(**this**,0,intent,0);  
*//分别对应上下文对象、请求码、请求意图用以指明启动类及数据传递、关键标志位。*builder.setContentIntent(pi); *//转跳*builder.setAutoCancel(**true**);  
Notification notification = builder.build();  
manager.notify(1, notification); *//通知标识ID Notification对象*

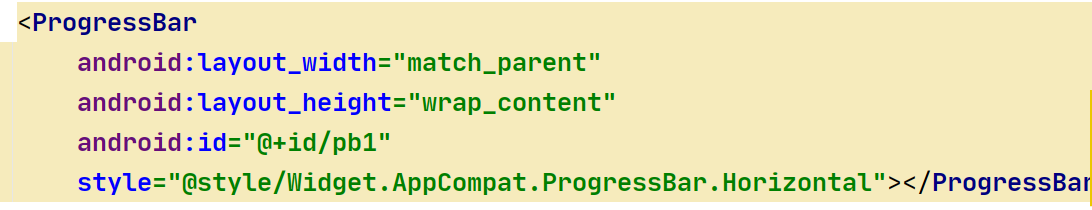
# OrderedBroadcast

有序广播，从优先级别最高的广播接收器开始接收，接收完了如果没有丢弃，就下传给下一个次高优先级别的广播接收器进行处理,依次类推，直到最后。

**public class** MainActivity **extends** AppCompatActivity {  
 **private** MyReceive **myReceive**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 *//* **myReceive** = **new** MyReceive();  
 IntentFilter intentFilter = **new** IntentFilter(**"OrderedBroadcastTest"**);  
 intentFilter.setPriority(1000); *//设置优先级* registerReceiver(**myReceive**,intentFilter);  
 *//* Button button = findViewById(R.id.***btn***);  
 button.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 Intent intent = **new** Intent(**"OrderedBroadcastTest"**);  
 sendOrderedBroadcast(intent,**null**); *//发送有序广播  
 //包含广播表示的intent对象 指定广播的权限* }  
 });  
 }  
  
@Override  
**protected void** onDestroy() {  
 unregisterReceiver(**myReceive**);  
 **super**.onDestroy();  
}

**public class** MyReceive **extends** BroadcastReceiver{  
  
 @Override  
 **public void** onReceive(Context context, Intent intent) {  
 abortBroadcast(); *//终止当前广播的传播* Toast.*makeText*(context,**"收到并终止广播"**,Toast.***LENGTH\_SHORT***).show();  
 }  
 }  
}

# ProgressBar



**pb1**.setProgress(0);  
Timer timer = **new** Timer();  
TimerTask task = **new** TimerTask() {  
 @Override  
 **public void** run() {  
 **int** cur = **pb1**.getProgress();  
 cur += 10;  
 **if** (cur<=100){  
 **pb1**.setProgress(cur);  
 }**else** {  
 **this**.cancel(); *//重写方法，this指TimerTask对象*

}  
 }  
};  
timer.scheduleAtFixedRate(task,1000,1000);  
*//参数：任务 第一次执行隔多少毫秒 剩下每次执行隔多少毫秒*

# ProgressDialog

ProgressDialog dlg = **new** ProgressDialog(MainActivity.**this**);  
dlg.setTitle(**"程序正在处理"**);  
dlg.setMessage(**"请稍后...."**);  
dlg.setCancelable(**false**);  
*//*dlg.setOnShowListener(**new** DialogInterface.OnShowListener() {  
 @Override  
 **public void** onShow(DialogInterface dialog) {  
 Timer timer = **new** Timer();  
 TimerTask timerTask = **new** TimerTask() {  
 @Override  
 **public void** run() {  
 dialog.dismiss();  
 }  
 };  
 timer.schedule(timerTask,3000);  
 *//等待3s后关闭程序* }  
});  
dlg.show();

# **RadioButton**

<**RadioGroup  
 android:layout\_marginTop="10dp"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/rg\_sex"  
 android:orientation="horizontal"**>  
 <**RadioButton  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/sex\_0"  
 android:text="保密"**></**RadioButton**>  
 <**RadioButton  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/sex\_1"  
 android:text="男"**></**RadioButton**>  
 <**RadioButton  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/sex\_2"  
 android:text="女"**></**RadioButton**>  
</**RadioGroup**>

RadioGroup **rgSex**;

**rgSex** = findViewById(R.id.***rg\_sex***);

**rgSex**.setOnCheckedChangeListener(**new** RadioGroup.OnCheckedChangeListener() {  
 @Override  
 **public void** onCheckedChanged(RadioGroup group, **int** checkedId) {  
 RadioButton rb = findViewById(checkedId);  
 Toast.*makeText*(MainActivity.**this**,rb.getText(),Toast.***LENGTH\_SHORT***).show();  
 }  
});

**int** rdoId = **rgSex**.getCheckedRadioButtonId();  
RadioButton rb = findViewById(rdoId);

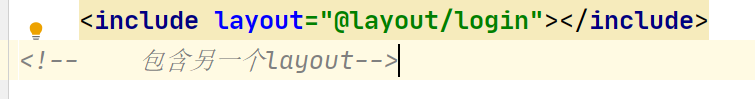
# NetworkstateReceive

*// 网络状态接收器*NetworkstateReceive **netReceive** = **null**;  
*//网络状态***boolean netReady** = **false**;  
  
@Override  
**protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 *//注册需要接受的通知* IntentFilter filter = **new** IntentFilter();  
 filter.addAction(**"android.net.conn.CONNECTIVITY\_CHANGE"**);  
 **netReceive** = **new** NetworkstateReceive();  
 registerReceiver(**netReceive**,filter);  
  
@Override  
**protected void** onDestroy() {  
 unregisterReceiver(**netReceive**);  
 **super**.onDestroy();  
}  
  
**public class** NetworkstateReceive **extends** BroadcastReceiver{  
  
 @Override  
 **public void** onReceive(Context context, Intent intent) {  
 *//更新网络状态  
  
 //网络连接管理器* ConnectivityManager m = (ConnectivityManager) getSystemService(Context.***CONNECTIVITY\_SERVICE***);  
  
 *//网络的连接状态相关信息* NetworkInfo netInfo = m.getActiveNetworkInfo();  
 **netReady** = (netInfo != **null** && netInfo.isAvailable());  
 *//提示信息* String msg;  
 **if** (**netReady**) msg = **"网络已连接"**;  
 **else** msg = **"没有可用网络"**;  
 Toast.*makeText*(context,msg,Toast.***LENGTH\_SHORT***).show();  
  
 }

# SaveFile

*//保存内容* **private void** saveFile(String filename, String content) {  
 **try** {  
 FileOutputStream s = openFileOutput(filename, ***MODE\_PRIVATE***);  
 s.write(content.getBytes()); *//转成二进制字节类型输入* s.flush();  
 s.close();  
*// 储存数据  
// SharedPreferences.Editor editor = getSharedPreferences("mydata",MODE\_PRIVATE).edit();  
// editor.putString("edit1",edit1.getText().toString());  
// editor.commit();* } **catch** (Exception e) {  
 e.printStackTrace();  
 Toast.*makeText*(**this**, e.getMessage(), Toast.***LENGTH\_SHORT***).show();  
 }  
 }  
  
 *//载入内容* **private** String loadFile(String filename) {  
 **try** {  
 FileInputStream s = openFileInput(filename);  
 **byte**[] buffer = **new byte**[s.available()];  
 s.read(buffer);  
 s.close();  
 String result = **new** String(buffer, **"utf-8"**);  
 *//将字符数组转换成utf-8格式的字符串* **return** result;  
  
  
*// SharedPreferences sp = getSharedPreferences("mydata",MODE\_PRIVATE);  
// edit1.setText(sp.getString("edit1","")); //第二个参数指定没找到数据时返回的默认值* } **catch** (Exception ex) {  
 ex.printStackTrace();  
 Toast.*makeText*(**this**, ex.getMessage(), Toast.***LENGTH\_SHORT***).show();  
 **return null**;  
 }  
 }

# SearchView





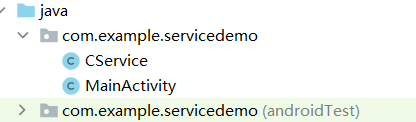
**public class** MainActivity **extends** AppCompatActivity **implements** SearchView.OnQueryTextListener {  
 **private** ListView **lstResult**;  
 **private** SearchView **searchView**;  
  
 **private** String[] **data** = {**"Tom"**,**"Jerry"**,**"John"**,**"Tim"**,**"Maria"**,**"Marry"**,**"Alice"**,**"Frank"**,**"Fan"**};  
  
 *//搜索结果* **private** List<String> **result** = **new** ArrayList<String>();  
 *//数据匹配* **private** ArrayAdapter<String> **adapter**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 *//* **lstResult** = findViewById(R.id.***lstResult***);  
 **searchView** = findViewById(R.id.***searchView***);  
 **searchView**.setIconifiedByDefault(**false**); *//默认true点击搜索图标后才能搜索* **searchView**.setSubmitButtonEnabled(**true**); *//搜索框右边是否有搜索按钮* **searchView**.setOnQueryTextListener(**this**);  
 *//数组适配器* **adapter** = **new** ArrayAdapter<String>(**this**,  
 android.R.layout.***simple\_list\_item\_1***,**result**);  
 **lstResult**.setAdapter(**adapter**);  
 }  
  
 @Override  
 **public boolean** onQueryTextSubmit(String query) {  
 **result**.clear();  
 **for**(String s:**data**){  
 **if** (s.toLowerCase().indexOf(query.toLowerCase())>=0){  
 *//从下标0开始* **result**.add(s);  
 }  
 **adapter**.notifyDataSetChanged(); *//更新列表* }  
 **return false**;  
 }  
  
  
  
 @Override  
 **public boolean** onQueryTextChange(String newText) {  
 **return false**;  
 }  
}

# SearchBar

<**SeekBar  
 android:layout\_width="match\_parent"  
 android:layout\_height="50dp"  
 android:id="@+id/seekbar1"  
 android:max="100"**></**SeekBar**>

**seekBar1**.setOnSeekBarChangeListener(**new** SeekBar.OnSeekBarChangeListener() {  
 @Override  
 **public void** onProgressChanged(SeekBar seekBar, **int** progress, **boolean** fromUser) {  
 **txt1**.setText(String.*format*(**"当前值： %d"**,progress));  
  
 }  
  
 @Override  
 **public void** onStartTrackingTouch(SeekBar seekBar) {  
 seekBar.setBackgroundColor(Color.*rgb*(200,200,200));  
  
 }  
  
 @Override  
 **public void** onStopTrackingTouch(SeekBar seekBar) {  
 seekBar.setBackgroundColor(Color.***WHITE***);  
  
 }  
});

# Service



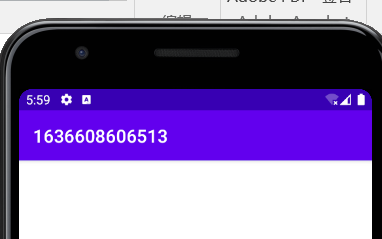


Cservice-Service

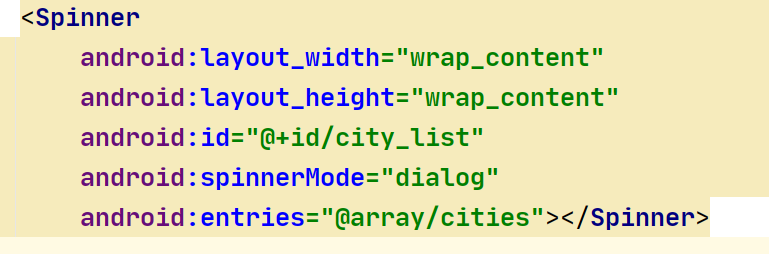
**public class** CService **extends** Service {  
 *//* **public class** CBinder **extends** Binder {  
 **public long** getTimestamp(){  
 **return** System.*currentTimeMillis*();  
 }  
 }  
 *//IBinder 接口类型对象 CBind必须实现了IBinder接口  
 //通过binder实现调用者client与Service之间的通信* **private** IBinder **MyBinder** = **new** CBinder();  
  
  
 *//构造函数* **public** CService() {  
 }  
  
 *//连接时返回binder* @Override  
 **public** IBinder onBind(Intent intent) {  
  
 **return MyBinder**;  
 }  
}

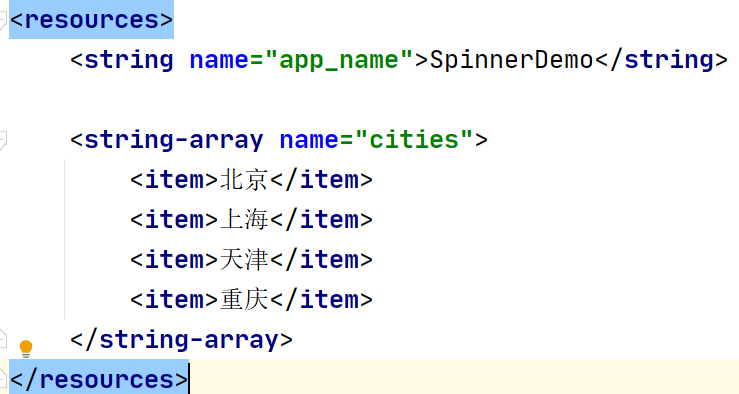
MainActivity-Client

**public class** MainActivity **extends** AppCompatActivity {  
 *//* CService.CBinder **binder**;  
 *//* **private** ServiceConnection **sCnn** = **new** ServiceConnection() {  
 @Override  
 **public void** onServiceConnected(ComponentName name, IBinder service) {  
 *//server参数为onBind函数返回值* **binder** = (CService.CBinder) service;  
 setTitle(String.*valueOf*(**binder**.getTimestamp()));  
 }  
  
 @Override  
 **public void** onServiceDisconnected(ComponentName name) {  
  
 }  
 };  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 Intent intent = **new** Intent(**this**,CService.**class**);  
 bindService(intent,**sCnn**,***BIND\_AUTO\_CREATE***);  
  
 }  
  
 @Override  
 **protected void** onDestroy() {  
 **super**.onDestroy();  
 unbindService(**sCnn**);  
 }  
}



# Spinner







*//获取城市名称***cityName** = getResources().getStringArray(R.array.***cities***);  
**cityList** = findViewById(R.id.***city\_list***);  
*//*Button btn1 = findViewById(R.id.***btn1***);  
  
**cityList**.setOnItemSelectedListener(**new** AdapterView.OnItemSelectedListener() {  
 @Override  
 **public void** onItemSelected(AdapterView<?> parent, View view, **int** position, **long** id) {  
 String msg = String.*format*(**"%d : %s"**,position,**cityName**[position]);  
 Toast t = Toast.*makeText*(MainActivity.**this**,msg,Toast.***LENGTH\_SHORT***);  
 t.show();  
 }  
  
 @Override  
 **public void** onNothingSelected(AdapterView<?> parent) {  
  
 }  
});

**int** index = **cityList**.getSelectedItemPosition(); // 获取当前选择item位置

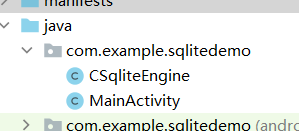
# Web资源获取

@Override  
 **public void** onClick(View v) {  
 **if**(v.getId()==R.id.***btn1***){  
 **new** Thread(**new** Runnable() {  
 @Override  
 **public void** run() {  
 HttpURLConnection cnn = **null**;  
 **try**{  
 URL url = **new** URL(**"http://10.0.2.2/getParam.aspx"**);  
 cnn = (HttpURLConnection) url.openConnection();  
 cnn.setRequestMethod(**"POST"**);  
 cnn.setConnectTimeout(10000);  
 cnn.setReadTimeout(10000);  
 *//设置参数* DataOutputStream outputStream = **new** DataOutputStream(cnn.getOutputStream());  
 outputStream.writeBytes(**"id=123456789"**);  
 *//* InputStream s = cnn.getInputStream();  
 StringBuilder sb = **new** StringBuilder();  
 BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(s));  
 String ln;  
 **while** ((ln = reader.readLine()) != **null**){  
 sb.append(ln+**'\n'**);  
  
 }  
 Log.*d*(**"页面内容"**, sb.toString());  
 showData(sb.toString());  
 }**catch** (Exception ex){  
 Log.*e*(**"获取网站资源错误"**, ex.getMessage());  
 }**finally** {  
 **if** (cnn != **null**) cnn.disconnect();  
 }  
 }  
 }) .start();  
 }**else if**(v.getId() == R.id.***btn2***){  
 **new** Thread(**new** Runnable() {  
 @Override  
 **public void** run() {  
 HttpURLConnection cnn = **null**;  
 **try** {  
 URL url = **new** URL(**"http://10.0.2.2/records.json"**);  
 cnn = (HttpURLConnection) url.openConnection();  
 cnn.setRequestMethod(**"GET"**);  
 cnn.setConnectTimeout(10000);  
 cnn.setReadTimeout(10000);  
 InputStream s = cnn.getInputStream();  
 StringBuilder sb = **new** StringBuilder();  
 BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(s));  
 String ln;  
 **while** ((ln = reader.readLine()) != **null**){  
 sb.append(ln);  
 }  
 Log.*d*(**"页面全部内容"**, sb.toString());  
 printJSONRecords(sb.toString());  
  
 }**catch** (Exception ex){  
 Log.*e*(**"获取网站资源错误"**, ex.getMessage());  
 }**finally** {  
 **if** (cnn != **null**) cnn.disconnect();  
 }  
 }  
 }).start();  
 }**else if**(v.getId() == R.id.***btn3***){  
 **new** Thread(**new** Runnable() {  
 @Override  
 **public void** run() {  
 HttpURLConnection cnn = **null**;  
 **try** {  
 URL url= **new** URL(**"http://10.0.2.2/records.xml"**);  
 cnn = (HttpURLConnection) url.openConnection();  
 cnn.setRequestMethod(**"GET"**);  
 cnn.setReadTimeout(10000);  
 cnn.setConnectTimeout(10000);  
 InputStream s = cnn.getInputStream();  
 printXml(s);  
  
  
  
 }**catch** (Exception ex){  
 Log.*e*(**"获取网站资源错误"**, ex.getMessage());  
  
 }**finally** {  
 **if**(cnn != **null**) cnn.disconnect();  
 }  
 }  
 }).start();  
 }  
 }  
  
 */\*\*  
 \* 让执行的代码保持在界面的主线程上  
 \** ***@param s*** *\*/* **private void** showData(**final** String s){  
 runOnUiThread(**new** Runnable() {  
 @Override  
 **public void** run() {  
 TextView txt1 = findViewById(R.id.***txt***);  
 txt1.setText(s);  
 }  
 });  
 }  
  
 *//显示一条json数据* **private void** printJSONRecord(String data){  
 **try**{  
 JSONObject obj = **new** JSONObject(data);  
 String id = obj.getString(**"id"**);  
 String username = obj.getString(**"username"**);  
 String sex = obj.getString(**"sex"**);  
 *//* Log.*d*(**"id"**, id);  
 Log.*d*(**"username"**, username);  
 Log.*d*(**"sex"**, sex);  
  
 }**catch** (Exception ex){  
 ex.printStackTrace();  
 }  
 }  
  
 *//显示json数组* **private void** printJSONRecords(String data){  
 **try**{  
 JSONArray arr = **new** JSONArray(data);  
  
 **for** (**int** i=0;i<arr.length();i++){  
 JSONObject obj = arr.getJSONObject(i);  
 String id = obj.getString(**"id"**);  
 String username = obj.getString(**"username"**);  
 String sex = obj.getString(**"sex"**);  
 *//* Log.*d*(**"id"**, id);  
 Log.*d*(**"username"**, username);  
 Log.*d*(**"sex"**, sex);  
 }  
  
  
 }**catch** (Exception ex){  
 ex.printStackTrace();  
 }  
 }  
  
 *//处理Xml数据* **private void** printXml(InputStream data){  
 **try**{  
 XmlPullParser parser = Xml.*newPullParser*();  
*// parser.setInput(new StringReader(String data)); 载入的文本内容应是正确的XML格式* parser.setInput(data,**"utf-8"**);  
 *//* String id = **""**;  
 String username = **""**;  
 String sex = **""**;  
 *//* String nodeName = **""**;  
 **int** eventType = parser.getEventType();  
 **while** (eventType != XmlPullParser.***END\_DOCUMENT***){  
 nodeName = parser.getName();  
 **if** (eventType == XmlPullParser.***START\_TAG***){  
 *//开始标记，读取记录* **if** (nodeName.equals(**"id"**)){  
 id = parser.nextText();  
 }  
 **if** (nodeName.equals(**"username"**)){  
 username = parser.nextText();  
 }  
 **if** (nodeName.equals(**"sex"**)){  
 sex = parser.nextText();  
 }  
 }**else if**(eventType == XmlPullParser.***END\_TAG***){  
 Log.*d*(**"id"**, id);  
 Log.*d*(**"username"**, username);  
 Log.*d*(**"sex"**, sex);  
 }  
  
 *//读取下一个* eventType = parser.next();  
 }  
 }**catch** (Exception ex){  
 Log.*e*(**"获取网站资源错误"**, ex.getMessage());  
  
 }  
 }

# Upload

**private void** upload(){  
 **new** Thread(**new** Runnable() {  
 @RequiresApi(api = Build.VERSION\_CODES.***O***)  
 @Override  
 **public void** run() {  
 **try** {  
 *//准备文件字节数据* Bitmap bitmap = BitmapFactory.*decodeResource*(getResources(),R.mipmap.***bilibili***);  
 ByteArrayOutputStream outputStream = **new** ByteArrayOutputStream();  
 bitmap.compress(Bitmap.CompressFormat.***PNG***,100,outputStream);  
  
 outputStream.flush();  
 outputStream.close();  
 **byte**[] data = outputStream.toByteArray();  
 Base64.Encoder encode = Base64.*getEncoder*();  
 String s = encode.encodeToString(data);  
  
 *//接受数据的服务器资源* URL url = **new** URL(**"http://10.0.2.2/upload.aspx"**);  
 *//上传文件* HttpURLConnection cnn = (HttpURLConnection) url.openConnection();  
 cnn.setRequestMethod(**"POST"**);  
 cnn.setConnectTimeout(10000);  
 cnn.setReadTimeout(10000);  
 cnn.setDoInput(**true**);  
 cnn.setDoOutput(**true**);  
 cnn.setUseCaches(**true**);  
 *//* DataOutputStream dos = **new** DataOutputStream(cnn.getOutputStream());  
  
 dos.writeBytes(**"file1="**+s);  
 dos.flush();  
 dos.close();  
  
 *//接受返回消息* InputStream inputStream = cnn.getInputStream();  
 StringBuilder sb = **new** StringBuilder();  
 BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(inputStream));  
 String ln;  
 **while** ((ln = reader.readLine()) != **null**){  
 sb.append(ln);  
 }  
 Log.*d*(**"上传结果"**, sb.toString());  
  
 }**catch** (Exception ex){  
 ex.printStackTrace();  
 }  
 }  
 }).start();  
}

# SQLite



CsqliteEngine

**public class** CSqliteEngine **extends** SQLiteOpenHelper {  
  
 *//创建日志记录表* **private static final** String ***sqlCreateLog*** =  
 **"create table if not exists log("**+  
 **"logid integer not null primary key,"**+  
 **"logtext text,"**+  
 **"logtime integer);"**;  
 *//添加日志记录数据* **private static final** String ***sqlInsertLog*** =  
 **"insert into log(logtext,logtime) values(?,?);"**;  
 *//构造函数* **public** CSqliteEngine(@Nullable Context context, @Nullable String name, @Nullable SQLiteDatabase.CursorFactory factory, **int** version) {  
 **super**(context, name, factory, version);  
 }  
  
 @Override  
 **public void** onCreate(SQLiteDatabase db) {  
 db.execSQL(***sqlCreateLog***);  
 db.execSQL(***sqlInsertLog***,**new** Object[]{**"创建数据库"**,System.*currentTimeMillis*()});  
 }  
  
 @Override  
 **public void** onUpgrade(SQLiteDatabase db, **int** oldVersion, **int** newVersion) {  
 db.execSQL(***sqlInsertLog***,**new** Object[]{**"更新数据库"**, System.*currentTimeMillis*()});  
  
 }  
}

MainActivity

@Override  
 **public void** onClick(View v) {  
 **if**(v.getId() == R.id.***btn1***){  
 *//打开数据库* CSqliteEngine dbe = **new** CSqliteEngine(**this**,***dbFile***,**null**,3);  
 SQLiteDatabase db = dbe.getWritableDatabase();  
 *//显示日志* Cursor records = db.rawQuery(**"select \* from log;"**,**null**);  
 StringBuffer sb = **new** StringBuffer();  
 **if**(records.moveToFirst()){  
 **do** {  
 sb.append(records.getString(0)).append(**","**);  
 sb.append(records.getString(1)).append(**","**);  
 sb.append(records.getString(2)).append(**","**);  
 sb.append(**"\n"**);  
 }**while** (records.moveToNext());  
 }  
 **txt1**.setText(sb.toString());  
 db.close();  
 dbe.close();  
 }**else if**(v.getId() ==R.id.***btn2***){  
 SQLiteDatabase db = **null**;  
 **try** {  
 *//打开数据库* db = openOrCreateDatabase(***dbFile***,***MODE\_PRIVATE***,**null**);  
 *//创建一个新的表* StringBuffer sql = **new** StringBuffer();  
 sql.append(**"create table if not exists user\_main("**);  
 sql.append(**"userid integer not null primary key,"**);  
 sql.append(**"username text not null unique,"**);  
 sql.append(**"userpwd text not null,"**);  
 sql.append(**"islocked integer not null default 0,"**);  
 sql.append(**"sex integer not null default 0,"**);  
 sql.append(**"email text"**);  
 sql.append(**");"**);  
 db.execSQL(sql.toString());  
 **txt1**.setText(**"数据表已建立"**);  
  
 ContentValues values = **new** ContentValues();  
 values.put(**"username"**,**"user03"**);  
 values.put(**"userpwd"**,**"123456"**);  
 values.put(**"islocked"**,**"0"**);  
 values.put(**"sex"**,1);  
 values.put(**"email"**,**"user03@aaa.bbb"**);  
 db.insert(**"user\_main"**,**null**,values);  
 *//uesr04* values.clear();  
 values.put(**"username"**,**"user04"**);  
 values.put(**"userpwd"**,**"123456"**);  
 values.put(**"islocked"**,**"0"**);  
 values.put(**"sex"**,2);  
 values.put(**"email"**,**"user04@aaa.bbb"**);  
 db.insert(**"user\_main"**,**null**,values);  
 *//* **txt1**.setText(**"记录已添加"**);  
 }**catch** (Exception ex){  
 **txt1**.setText(ex.getMessage());  
 }**finally** {  
 **if** (db != **null**){  
 db.close();  
 }  
 }  
 }**else if**(v.getId() == R.id.***btn3***){  
 SQLiteDatabase db = **null**;  
 **try** {  
 db = openOrCreateDatabase(***dbFile***,***MODE\_PRIVATE***,**null**);  
 *//查询数据* String sql = **"select \* from user\_main where username like '%user%' and sex=2;"**;  
 Cursor records = db.rawQuery(sql,**null**);  
 showData(records);  
 records.close();  
 }**catch**(Exception ex){  
 **txt1**.setText(ex.getMessage());  
 }**finally** {  
 **if** (db!= **null**){  
 db.close();  
 }  
 }  
 }**else if**(v.getId() == R.id.***btn4***){  
 SQLiteDatabase db = **null**;  
 **try** {  
 *//打开数据库* db = openOrCreateDatabase(***dbFile***,***MODE\_PRIVATE***,**null**);  
 *//更新数据库* ContentValues values = **new** ContentValues();  
 values.put(**"userpwd"**,**"000000"**);  
 values.put(**"sex"**,1);  
 **int** result = db.update(**"user\_main"**,values,**"userpwd=?"**,**new** String[]{**"123456"**});  
 **txt1**.setText(String.*valueOf*(result));  
 }**catch**(Exception ex){  
 **txt1**.setText(ex.getMessage());  
 }**finally** {  
 **if**(db != **null**){  
 db.close();  
 }  
 }  
 }**else if** (v.getId() == R.id.***btn5***){  
  
 SQLiteDatabase db = **null**;  
 **try** {  
 *//打开数据库* db = openOrCreateDatabase(***dbFile***,***MODE\_PRIVATE***,**null**);  
 *//删除记录* **int** result = db.delete(**"user\_main"**,**"username='user03'"**,**null**);  
 **txt1**.setText(String.*valueOf*(result));  
 }**catch**(Exception ex){  
 **txt1**.setText(ex.getMessage());  
 }**finally** {  
 **if**(db != **null**){  
 db.close();  
  
 }  
 }  
 }**else if**(v.getId() == R.id.***btn6***){  
 SQLiteDatabase db = **null**;  
 **try** {  
 *//打开数据库* db = openOrCreateDatabase(***dbFile***,***MODE\_PRIVATE***,**null**);  
 *//创建user\_permission表* StringBuilder sb = **new** StringBuilder();  
 sb.append(**"create table if not exists user\_permission (\n"**+  
 **"upid integer not null primary key,\n"**+  
 **"userid integer not null references user\_main(userid),\n"**+  
 **"permission integer not null\n"**+  
 **");"**);  
 db.execSQL(sb.toString());  
 *//添加记录* String sql = **"insert into user\_permission(userid,permission) values(1,1);"**;  
 db.execSQL(sql);  
 sql = **"insert into user\_permission(userid,permission) values(2,1001);"**;  
 db.execSQL(sql);  
 sql = **"insert into user\_permission(userid,permission) values(2,1002);"**;  
 db.execSQL(sql);  
 sql = **"insert into user\_permission(userid,permission) values(3,1001);"**;  
 db.execSQL(sql);  
 sql = **"insert into user\_permission(userid,permission) values(3,1003);"**;  
 db.execSQL(sql);  
 sql = **"insert into user\_permission(userid,permission) values(4,1001);"**;  
 db.execSQL(sql);  
 sql = **"insert into user\_permission(userid,permission) values(4,1002);"**;  
 db.execSQL(sql);  
  
 **txt1**.setText(**"OK"**);  
 }**catch**(Exception ex){  
 **txt1**.setText(ex.getMessage());  
 }**finally** {  
 **if**(db != **null**){  
 db.close();  
  
 }  
 }  
 }**else if**(v.getId() == R.id.***btn7***){  
 SQLiteDatabase db = **null**;  
 **try** {  
 *//打开数据库* db = openOrCreateDatabase(***dbFile***,***MODE\_PRIVATE***,**null**);  
 *//  
// String sql = "select U.\*,P.upid,P.permission from"+  
// " user\_main as U join user\_permission as P on U.userid=P.userid;";* StringBuilder sb = **new** StringBuilder();  
 sb.append(**"create view if not exists v\_user \n"**+  
 **"as \n"**+  
 **"select U.\*,P.upid,P.permission \n"**+  
 **"from user\_main as U join user\_permission as P \n"**+  
 **"on U.userid=P.userid;\n"**);  
 db.execSQL(sb.toString());  
 String sql = **"select userid,username,permission from v\_user;"**;  
 Cursor records = db.rawQuery(sql,**null**);  
 showData(records);  
 records.close();  
 }**catch**(Exception ex){  
 **txt1**.setText(ex.getMessage());  
 }**finally** {  
 **if**(db != **null**){  
 db.close();  
  
 }  
 }  
 }  
  
  
 }  
  
  
  
  
  
 **private void** showData(Cursor records){  
 **if** (records == **null** || records.getCount()==0){  
 **txt1**.setText(**"没有满足条件的记录"**);  
 **return**;  
 }  
 StringBuffer sb = **new** StringBuffer();  
 **int** fieldCount = records.getColumnCount();  
 *//显示字段名* **for**(**int** i =0;i<fieldCount;i++){  
 sb.append(records.getColumnName(i));  
 sb.append(**" | "**);  
  
 }  
 sb.append(**"\n"**);  
 *//显示记录* **if**(records.moveToFirst()){  
 **do**{  
 **for**(**int** i =0;i<fieldCount;i++){  
 sb.append(records.getString(i));  
 sb.append(**" | "**);  
  
 }  
 sb.deleteCharAt(sb.length()-1);  
 sb.append(**"\n"**);  
  
 }**while**(records.moveToNext());  
 }  
 **txt1**.setText(sb.toString());  
  
  
 }

# RelativeLayout

