广播接收者BroadCastReceiver　brdksrsw

一个广播可以有一个发送者多个接收者，而接收广播的会因为权限或优先级不同而接收不到或者接收的数据不相同

安卓应用程序里面的电台：系统内置的一个服务，会把事件（电量不足、电量充满、开机启动完成、拨打电话、短信到来、SD卡挂载）作为一个广播消息发送给其他的接受者。谷歌开发广播接受者，为了方便程序员开发应用，监听安卓系统或应用程序发出的广播

广播接收者的生命周期比较短，只限于onReceive()方法，

方法执行完毕后，该生命周期结束，等待下次广播到来

不要在onReceive()中做耗时的操作，可以开启服务，或者是发送通知

注册广播接收者：

1、定义一个普通类继承extends　BroadcastReceiver

2、重写onReceive(Context context, Intent intent)方法的不同情况：

一、向外拨打电话的广播监听改变最终外拨电话的号码

从上一个发送广播的有数据一同发送出来，接收者就要考虑获取数据，获取到数据之后要考虑到转发给下一个接收者

```
String number = this.getResultData();
```

将修改后的数据放入广播中，这样下一个接收者才能收到数据---------接收转发广播，转发给下一个广播接收者

```
this.setResultData(接收到处理后的数据);
```

注意：如果是系统的动作行为，那么只需要获取的行为数据就可以了

二、短信获取拦截

1、获取数据：

```
Bundle bundle = intent.getExtras();
```

2、获取短信专用

```
Object[] objs = (Object[]) bundle.get("pdus");
```

3、将Object转成SmsMessage

```
SmsMessage smsMessage = SmsMessage.createFromPdu((byte[])obj);
```

4、获取短信的号码

```
smsMessage.getOriginatingAddress();
```

5、获取短信的内容

```
smsMessage.getMessageBody();
```

6、判断号码并放弃传递

```
if("15555215556".equals(address)){
```

就拦截，就是不让其再转发到系统内置的短信应用，放弃广播传递

```
this.abortBroadcast();  广播接收者放弃传递
```

三、SD卡、应用的卸载安装监听广播接收

应用和SD卡的卸载安装监听

1、获取SD卡卸载发出的广播的动作

```
String action = intent.getAction();
```

2、根据获取到的动作判断广播的类型做出相应的处理

```
if("android.intent.action.PACKAGE_ADDED".equals(action)){
    处理。。。。。
}else if("android.intent.action.PACKAGE_REMOVED".equals(action)){
    处理。。。。。
```

应用和SD卡的卸载安装监听广播的清单注册：

```
        应用的安装卸载广播注册
<receiver android:name="cn.itcast.receiver.AppStateReceiver">
    <intent-filter>                <!-- 安装 -->
    <action android:name="android.intent.action.PACKAGE_ADDED"/>
    <!-- 卸载 -->
    <action android:name="android.intent.action.PACKAGE_REMOVED"/>
    <!-- 附加: sd->file,app->package -->
    <data android:scheme="package"/>
    </intent-filter>
</receiver>
```

```
        SD卡安装卸载广播接收注册
<receiver android:name="cn.itheima.receiver.SdcardStateReceiver">
    <intent-filter>                <!-- 安装SD卡 -->
        <action android:name="android.intent.action.MEDIA_MOUNTED"/>
                                   <!-- 卸载SD卡 -->
        <action android:name="android.intent.action.MEDIA_UNMOUNTED"/>
        <!-- 需要前缀 -->
        <data android:scheme="file"/>
    </intent-filter>
</receiver>
```

开机广播接收监听- - - - - >只要设置一开机解锁后就显示某个界面

注册广播意图，监听开机完成广播动作 <action android:name="android.intent.action.BOOT_COMPLETED"/>

1、广播接收者类中重写方法onReceive(Context context, Intent intent)

2、创建显示意图，通过意图跳转到相应的界面

当处于没有界面的时候可以强行创建任务栈，以便于界面可以展示出来

Intent myIntent = new Intent();

显示意图打开界面

myIntent.setClass(context,MainActivity.class);

3、由于这只是广播接收者没有界面显示，而要打开一个界面需要强制创建任务栈以便于新打开的界面装入其中

myIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);

4、开启意图

context.startActivity(myIntent);

```
public void onReceive(Context context, Intent intent) {
    System.out.println("我监听到了开机完毕了。。。");
    //当处于没有界面的时候可以强行创建任务栈，以便于界面可以展示出来
    Intent myIntent = new Intent();
    //显示意图打开界面
    myIntent.setClass(context,MainActivity.class);
    //由于是广播没有界面所以要，强行创建任务栈，以便于MainActivity装入其中
    myIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    context.startActivity(myIntent);
```

3、在清单文件中注册广播：

<receiver android:name="cn.itheima.receiver.IpReceiver"> 注册继承了BroadcastReceiver的类为广播接收类

<intent-filter android:priority="1000">

配置广播接收的优先级，默认为0，priority取值范围是:-1000到+1000,数字大，优先级高 如果优先级相同，按注册的顺序一旦配置了优先级，按优先级

action:name表示监听接收广播的动作

android.intent.action.NEW_OUTGOING_CALL:外拨电话的广播

```
            <action android:name="android.intent.action.NEW_OUTGOING_CALL"/>
        </intent-filter>
</receiver>
```

4、配置相应权限

广播接收者相关权限和广播动作：

安卓4.0版本之后为了安全考虑，要求应用程序必须要有界面，必须被用户运行过一次，广播接受者才会生效

配置应用的优先级，默认为0,priority取值范围是:-1000到+1000,数字大，优先级高
如果优先级相同，按注册的顺序
intent-filter android:priority="1000"

外播电话权限
android.permission.PROCESS_OUTGOING_CALLS

接收短信需要权限
android.permission.RECEIVE_SMS

监听开机需要的权限
android.permission.RECEIVE_BOOT_COMPLETED

监听打电话的动作
android.intent.action.NEW_OUTGOING_CALL

监听短信接收的（动作）广播
android.provider.Telephony.SMS_RECEIVED"

监听SD卡的安装动作
android.intent.action.MEDIA_MOUNTED
监听SD卡的卸截动作
android.intent.action.MEDIA_UNMOUNTED
监听SD卡意图数据
data android:scheme="file"

监听开机完成动作
android.intent.action.BOOT_COMPLETED

```
<!-- 读取电话状态需要权限 -->
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<!-- 写SD卡需要权限 -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<!-- 录音需要权限 -->
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
```

自定义发送广播：

发送广播者

只需要设定要发送的广播的动作和意图中的数据然后发送就可以了，那么发送广有两种

发送有序广播：sendOrderedBroadcast(intent,null);

发送无序广播：sendBroadcast(intent);

区别

无序广播：接收者几乎是在同一时间接收到广播，广播在发送过程中不能被修改，不会被拦截，效率高无接收主次之分
有序广播：按接收者的优先级依次接收或有先级低的可能接接不到，效率低，数据可被拦截改变等。
具体代码：
1、创建意图
Intent intent = new Intent();
2、设置广播的动作
intent.setAction("www.cctv.com"); 这个动作一定是要有接收者才有意义 即广播接收者的注册中有这个自定义广播意图的动作才有效
3、将意图中绑定数据，这个数据可有可无根据需求来确定
intent.putExtra("NAME","中央电视台");
4、 发送无序广播
参数说明：无序广播，参数为意图 有序广播参数一：意图，参数二：接收者的权限，如果写null的话，会优先级的高低依次接收
this.sendBroadcast(intent); 或者发送有序广播 this.sendOrderedBroadcast(intent,null);

参数	含义
intent	意图
receiverPermission	接收的权限，如不需要权限传null
resultReceiver	最终的接收者
scheduler	自定义的handler来执行最终接受者的回调，如果为null，则在主线程中执行。
initialCode	初始码，通常为Activity.RESULT_OK
initialData	初始化的数据
initialExtras	初始化的结果的额外值，通常为null。

应用内 内部广播的发送

未来意图广播

使用服务注册广播接收者（特殊广播）

一、用代码注册普通广播：
创建一个广播接收的类，定义相关广播事件
二、在主界面中用代码注册广播
1、要创建广播类对象 在服务中自定义广播类 获取广播类对象
2、使用意图 注册广播 `IntentFilter filter = new IntentFilter();`
3、通过意图添加广播动作
`filter.addAction("");filter.addAction("");` 这个动作就是用清单文件注册时的广播意图

动作

4、注册广播

参一:广播类对象　　参二：意图对象

```
this.registerReceiver(广播类对象, 意图对象);
```

取消广播注册　　反注册

```
this.unregisterReceiver(广播类对象);
```

通过服务注册一个长久运行的广播

1、创建广播类，定义相关广播　　这个广播类必须继承BroadcastReceiver可以是服务的内部类

2、创建服务类，并在清单文件中注册服务，并在服务类中创建广播类对象

3、在服务类的onCreate()方法中注册广播

```
IntentFilter  filter  =  new  IntentFilter();
```

添加广播动作

```
filter.addAction(Intent.ACTION_SCREEN_OFF);
filter.addAction(Intent.ACTION_SCREEN_ON);
```

注册广播

```
this.registerReceiver(广播类对象, 意图对象);
```

4、在服务类的 onDestroy方法中取消广播注册

```
this.unregisterReceiver(scr);
```

5、在主界面开启服务关闭服务

开启服务

```
Intent  intent  =  new  Intent(this,服务类.class);
    this.startService(意图对象);
```

关闭服务

```
Intent  intent  =  new  Intent(this,服务类.class);
    this.stopService(意图对象);
```

```
<service android:name="com.liu.ser.PinmuService"></service>
```

清单中只需要注册服务就可以了

```
public class PinmuService extends Service {
    private ScrReceiver scr;
    @Override
    public void onCreate() {
        if(scr==null){
            scr = new ScrReceiver();
            IntentFilter filter = new IntentFilter();
            filter.addAction(Intent.ACTION_SCREEN_OFF);
            filter.addAction(Intent.ACTION_SCREEN_ON);
            this.registerReceiver(scr, filter);
        }
        super.onCreate();
    }
    @Override
    public void onDestroy() {
        if(scr!=null){
            this.unregisterReceiver(scr);
        }

        super.onDestroy();
    }
```

注册 手机屏幕开关意图活动 的广播

```
    Intent intent = new Intent(this,PinmuService.class);
    this.startService(intent);
    but.setText("停止");
    SharedPreferences sp = this.getSharedPreferences("fuwu", Context.MODE_APPEND);
    Editor edit = sp.edit();
    edit.putString("fu", "停止");
    edit.commit();
else{
    Intent intent = new Intent(this,PinmuService.class);
    this.stopService(intent);
    but.setText("开启");
    SharedPreferences sp = this.getSharedPreferences("fuwu", Context.MODE_APPEND);
    Editor edit = sp.edit();
    edit.putString("fu", "开启");
    edit.commit();
}
```

开启服务

停止服务

```
public class ScrReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context arg0, Intent arg1) {
        Log.w("mylog", "锁屏了");
    }

}
```

广播事件