广播

https://developer.android.google.cn/develop/background-work/background-tasks/broadcasts?hl=zh_cn

安卓应用可以通过Android系统发送或接收广播，Android系统会在发生各种系统事件时发送广播，比如开机、网络连接、电池电量变化等。应用也可以发送自定义广播。

接收广播

应用可以通过两种方式接收广播：通过清单声明的接收器和上下文注册的接收器。 - Android 8.0 及以上版本不支持清单声明的接收器。

上下文注册的接收器

- `BroadcastReceiver` - 广播接收器，需要继承并重写`onReceive`方法。
  - `setResultExtras` - 传递数据
  - `abortBroadcast` - 终止广播
  - `getResultExtras` - 获取数据
- `registerReceiver` - 注册接收器
- `unregisterReceiver` - 注销接收器
- `sendBroadcast` - 发送广播
- `sendOrderedBroadcast` - 发送有序广播
- `LocalBroadcastManager` - 本地广播管理器，用于发送本地广播，只能在应用内部使用，无法发送给其他应用。

接收系统广播

接收网络切换的广播。

```
public class MyReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        // 处理广播
        Log.v("MyReceiver", intent.getAction());
    }
}

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

    // 注册接收器
    IntentFilter filter = new IntentFilter();
    filter.addAction("WifiManager.NETWORK_STATE_CHANGED_ACTION"); // 网络状态变化
```

```
    filter.addAction("android.net.conn.CONNECTIVITY_CHANGE"); // 网络连接变化
    filter.addAction("Telephony.Sms.Intents.SMS_RECEIVED_ACTION"); // 短信接收
    filter.addAction("Intent.ACTION_BATTERY_CHANGED"); // 电池电量变化

    registerReceiver(new MyReceiver(), filter);
}
```

发送自定义广播

- 定义广播名（action）
- 继承BroadcastReceiver，重写onReceive方法
- registerReceiver注册广播接收器
- sendBroadcast发送广播

```
// 定义广播名
public static final String MY_BROADCAST = "com.example.myapplication.MY_BROADCAST";

public class MyReceiver extends BroadcastReceiver { // 广播接收器

    @Override
    public void onReceive(Context context, Intent intent) {
        // 处理广播
        Log.v("MyReceiver", intent.getAction());
    }
}

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

    // 注册接收器
    IntentFilter filter = new IntentFilter();
    filter.addAction(MY_BROADCAST); // 接收自定义广播

    registerReceiver(new MyReceiver(), filter);

    // 点击按钮发送广播
    findViewById(R.id.button).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // 发送广播
            Intent intent = new Intent();
            intent.setAction(MY_BROADCAST);
            sendBroadcast(intent);
        }
    });
}
```

发送有序广播

有序广播可以设置优先级，优先级高的接收器先接收广播，优先级高的可以终止广播，优先级低的接收器无法接收广播。

- setPriority - 设置优先级

```
// 定义广播名
public static final String MY_BROADCAST = "com.example.myapplication.MY_BROADCAST";

public class MyReceiver extends BroadcastReceiver { // 广播接收器

    @Override
```

```
    public void onReceive(Context context, Intent intent) {
        // 处理广播
        Log.v("MyReceiver", intent.getAction() + " " + intent.getStringExtra("key"));

        // 发送数据
        Bundle bundle = new Bundle();
        bundle.putString("key1", "value1");
        setResultExtras(bundle);

        // 终止广播
        abortBroadcast();
    }
}
public class MyReceiver2 extends BroadcastReceiver { // 广播接收器

    @Override
    public void onReceive(Context context, Intent intent) {
        // 处理广播
        Log.v("MyReceiver2", intent.getAction() + " " + intent.getStringExtra("key") + " " + g
    }
}

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

    // 注册接收器
    IntentFilter filter = new IntentFilter();
    filter.addAction(MY_BROADCAST); // 接收自定义广播
    filter.setPriority(100); // 设置优先级

    registerReceiver(new MyReceiver(), filter);

    IntentFilter filter1 = new IntentFilter();
    filter1.addAction(MY_BROADCAST); // 接收自定义广播
    filter1.setPriority(200); // 设置优先级
    registerReceiver(new MyReceiver2(), filter1);

    // 点击按钮发送广播
    findViewById(R.id.button).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // 发送广播
            Intent intent = new Intent(MY_BROADCAST);
            intent.putExtra("key", "value");
            sendBroadcast(intent);
        }
    });
}
```

发送本地广播

- `LocalBroadcastManager` - 本地广播管理器，用于发送本地广播，只能在应用内部使用，无法发送给其他应用。

```
// 定义广播名
public static final String MY_BROADCAST = "com.example.myapplication.MY_BROADCAST";

public class MyReceiver extends BroadcastReceiver { // 广播接收器

    @Override
    public void onReceive(Context context, Intent intent) {
        // 处理广播
        Log.v("MyReceiver", intent.getAction() + " " + intent.getStringExtra("key"));
    }
```

```
}

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

    // 注册接收器
    IntentFilter filter = new IntentFilter();
    filter.addAction(MY_BROADCAST); // 接收自定义广播

    LocalBroadcastManager.getInstance(this).registerReceiver(new MyReceiver(), filter);

    // 点击按钮发送广播
    findViewById(R.id.button).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // 发送广播
            Intent intent = new Intent(MY_BROADCAST);
            intent.putExtra("key", "value");
            LocalBroadcastManager.getInstance(MainActivity.this).sendBroadcast(intent);
        }
    });
}
```

通知

https://developer.android.google.cn/develop/ui/views/notifications?hl=zh-cn

通知是Android在状态栏显示的一条消息，用于向用户显示重要的信息。通知可以包含标题、内容、图标、声音、震动等信息。

- `NotificationManager` - 用于创建渠道和显示通知，
  - `createNotificationChannel` - 创建通知渠道
  - `notify` - 显示通知
- `NotificationChannel` - 通知渠道，用于设置通知的属性，构造`NotificationChannel(String id, String name, int importance)`
  - `id` - 通知渠道的唯一标识符
  - `name` - 通知渠道的名称
  - `importance` - 通知渠道的重要性，有以下几种：
    - `NotificationManager.IMPORTANCE_MIN` - 低，无声且不会出现在通知栏中
    - `NotificationManager.IMPORTANCE_LOW` - 低，无声但会出现在通知栏中
    - `NotificationManager.IMPORTANCE_DEFAULT` - 默认，有声音
    - `NotificationManager.IMPORTANCE_HIGH` - 高，有声音且会出现在通知栏中
- `Notification` - 通知，不会直接实例化，使用下面的类
- `NotificationCompat.Builder` - 通知构造器，构造`NotificationCompat.Builder(Context context, String channelId)`
  - `setSmallIcon` - 设置通知的小图标，必选
  - `setLargeIcon` - 设置通知的大图标
  - `setContentTitle` - 设置通知的标题
  - `setContentText` - 设置通知的内容
  - `setContentIntent` - 设置通知的点击事件，点击通知时会启动指定的活动
  - `setAutoCancel` - 设置通知是否自动取消，点击通知后会自动消失
  - `setSound` - 设置通知的声音
  - `setVibrate` - 设置通知的震动
  - `setProgress` - 设置通知的进度条
  - `addAction` - 添加通知的动作，点击动作时会触发指定的意图
  - `setCustomContentView` - 设置通知的自定义`View`
  - `setStyle` - 设置通知的样式，有以下几种：

        - `Notification.BigTextStyle` - 大文本样式，用于显示长文本
        - `Notification.BigPictureStyle` - 大图片样式，用于显示大图
    - `build` - 构建通知，返回`Notification`对象
- `PendingIntent` - 用于启动活动或服务，构造`PendingIntent.getActivity(Context context, int requestCode, Intent intent, int flags)`
- `RemoteViews` - 用于设置通知的自定义`View`，构造`RemoteViews(String packageName, int layoutId)`
    - `setTextViewText` - 设置`TextView`的文本
    - `setImageViewResource` - 设置`ImageView`的图片
    - `setProgressBar` - 设置进度条
    - `setOnClickPendingIntent` - 设置`View`的点击事件，点击`View`时会启动指定的活动
    - `setViewVisibility` - 设置`View`的可见性

创建和管理通知渠道

从Android 8.0（API级别26）开始，通知需要在通知渠道中注册，通知渠道是通知的分类，通知必须属于通知渠道。

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

    // 创建通知渠道
    NotificationManager manager = getSystemService(NotificationManager.class);

    NotificationChannel channel = new NotificationChannel("低渠道", "渠道名 低", NotificationMana
    channel.setDescription("渠道描述 这是一个低渠道");
    manager.createNotificationChannel(channel);

    NotificationChannel channel1 = new NotificationChannel("默认渠道", "渠道名 默认", Notificatio
    channel1.setDescription("渠道描述 这是一个默认渠道");
    manager.createNotificationChannel(channel1);

    NotificationChannel channel2 = new NotificationChannel("高渠道", "渠道名 高", NotificationMar
    channel2.setDescription("渠道描述 这是一个高渠道");
    manager.createNotificationChannel(channel2);
}
```

显示通知

简单显示

```
// 使用默认渠道构造通知
NotificationCompat.Builder builder = new NotificationCompat.Builder(this, "默认渠道");

// 启动活动
Intent intent = new Intent(this, MainActivity.class);
PendingIntent pendingIntent = PendingIntent.getActivity(this, 0, intent, 0);

// 设置通知的属性
builder.setConentTitle("通知标题")
        .setContentText("通知内容")
        .setSmallIcon(R.drawable.ic_launcher_foreground)
        .setContentIntent(pendingIntent)
        .setAutoCancel(true);

// 显示通知
manager.notify(1, builder.build());
```

添加按钮

给通知添加按钮，点击按钮时会触发指定的意图。

```
public String action = "com.example.myapplication.MY_ACTION";
public class MyReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        // 处理广播
        Log.v("MyReceiver", intent.getAction());
    }
}

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

    // 注册接收器
    IntentFilter filter = new IntentFilter();
    filter.addAction(action);
    registerReceiver(new MyReceiver(), filter);

    // 获取通知管理器
    NotificationManager manager = getSystemService(NotificationManager.class);

    // 设置通知渠道
    NotificationChannel channel = new NotificationChannel("默认渠道", "渠道名 默认", NotificationI
    channel.setDescription("渠道描述 这是一个默认渠道");
    manager.createNotificationChannel(channel);

    // 构造通知
    NotificationCompat.Builder builder = new NotificationCompat.Builder(this, "默认渠道");

    // 启动活动
    Intent intent = new Intent(this, MainActivity.class);
    PendingIntent pendingIntent = PendingIntent.getActivity(this, 0, intent, 0);

    // 发送广播
    Intent intent1 = new Intent(action);
    PendingIntent pendingIntent1 = PendingIntent.getBroadcast(this, 0, intent1, 0);

    // 设置通知的属性
    builder.setConentTitle("通知标题")
            .setContentText("通知内容")
            .setSmallIcon(R.drawable.ic_launcher_foreground)
            .setContentIntent(pendingIntent)
            .setAutoCancel(true)
            .addAction(R.drawable.ic_launcher_foreground, "启动活动", pendingIntent)
            .addAction(R.drawable.ic_launcher_foreground, "发送广播", pendingIntent1);

    // 点击按钮
    findViewById(R.id.button).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // 显示通知
            manager.notify(1, builder.build());
        }
    });
}
```

自定义View

创建布局文件

```
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="播放器"/>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="开始"
            android:id="@+id/btn_start"/>

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="暂停"
            android:id="@+id/btn_stop"/>
    </LinearLayout>
```

活动文件

```
    public MediaPlayer mediaPlayer;
    public String actionStart = "com.example.player.start";
    public String actionStop = "com.example.player.stop";
    public class MyReceiver extends BroadcastReceiver{

        @Override
        public void onReceive(Context context, Intent intent) {
            // 播放音乐
            mediaPlayer.start();
        }
    }
    public class MyReceiver1 extends BroadcastReceiver{

        @Override
        public void onReceive(Context context, Intent intent) {
            // 暂停音乐
            mediaPlayer.pause();
        }
    }

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

        // 音乐
        mediaPlayer = new MediaPlayer();
        try {
            mediaPlayer.setDataSource(getResources().openRawResourceFd(R.raw.yyx));
            mediaPlayer.prepare();
        } catch (IOException e) {
            e.toString();
        }

        // 注册接收器
        IntentFilter filter = new IntentFilter(actionStart);
        registerReceiver(new MyReceiver(), filter);
        IntentFilter filter1 = new IntentFilter(actionStop);
        registerReceiver(new MyReceiver1(), filter1);

        // 播放
        Intent intent = new Intent();
        intent.setAction(actionStart);
        // 暂停
        Intent intent1 = new Intent();
```

```
        intent1.setAction(actionStop);

        // 加载View
        RemoteViews remoteViews = new RemoteViews(getPackageName(), R.layout.player);
        PendingIntent pendingIntent = PendingIntent.getBroadcast(this, 0, intent, PendingIntent
        remoteViews.setOnClickPendingIntent(R.id.btn_start, pendingIntent);

        PendingIntent pendingIntent1 = PendingIntent.getBroadcast(this, 0, intent1, PendingInt
        remoteViews.setOnClickPendingIntent(R.id.btn_stop, pendingIntent1);

        // 获取通知管理器
        NotificationManager manager = getSystemService(NotificationManager.class);

        // 创建渠道
        NotificationChannel channel = new NotificationChannel("默认渠道", "渠道名 默认渠道", Notif
        manager.createNotificationChannel(channel);

        // 构建通知并显示
        NotificationCompat.Builder builder = new NotificationCompat.Builder(MainActivity.this,
        builder.setSmallIcon(R.drawable.bomb)
                .setCustomContentView(remoteViews);

        manager.notify(1, builder.build());
    }
```