ANDROID移动互联网应用开发

傅晓，河海大学计算机与信息学院
2019年3月

ANDROID事件处理

- 在 Android 系统中，从用户与应用的交互中截获事件的方法不止一种。如考虑截获用户界面内的事件，则可从用户与之交互的特定视图对象中捕获事件。为此，View 类提供了多种方法。
- 在您将用于构建布局的各种 View 类中，您可能会注意到几种看起来适用于 UI 事件的公共回调方法。当该对象上发生相应的操作时，Android 框架会调用这些方法。
- 例如，在触摸一个视图对象（例如“按钮”）时，对该对象调用 onTouchEvent() 方法。不过，为了截获此事件，您必须扩展 View 类并重写该方法。 然而，为了处理此类事件而扩展每个视图对象并不现实。

ANDROID事件处理

- 正因如此，View 类还包含一系列嵌套接口以及您可以更加轻松定义的回调。这些接口称为事件侦听器，是您捕获用户与 UI 之间交互的票证。
- 尽管您通常会使用事件侦听器来侦听用户交互，但有时您确实需要扩展 View 类以构建自定义组件。也许，您想扩展 Button 类来丰富某些内容的样式。 在这种情况下，您将能够使用该类的事件处理程序为类定义默认事件行为。
- 事件侦听器是 View 类中包含一个回调方法的接口。 当用户与 UI 项目之间的交互触发已注册此视图的侦听器时，Android 框架将调用这些方法。

ANDROID事件处理

- Android事件侦听器的回调方法：
- **onClick()**：在 View.OnClickListener 中。 当用户触摸项目（处于触摸模式下）时，或者使用导航键或轨迹球聚焦于项目，然后按适用的“Enter”键或按下轨迹球时，将调用此方法。
- **onLongClick()**：在 View.OnLongClickListener 中。 当用户触摸并按住项目（处于触摸模式下）时，或者使用导航键或轨迹球聚焦于项目，然后按住适用的“Enter”键或按住轨迹球（持续一秒钟）时，将调用此方法。

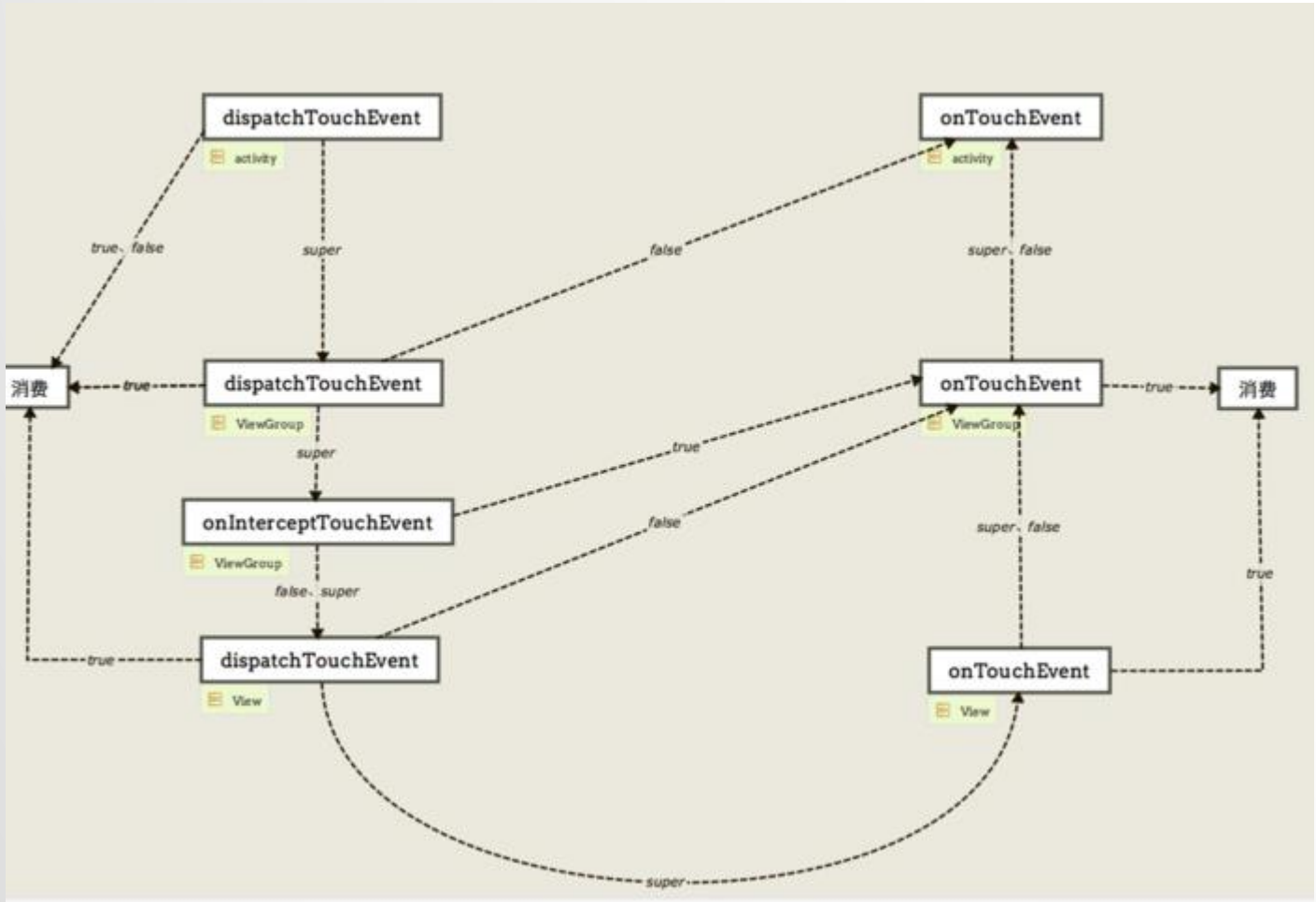
ANDROID事件处理

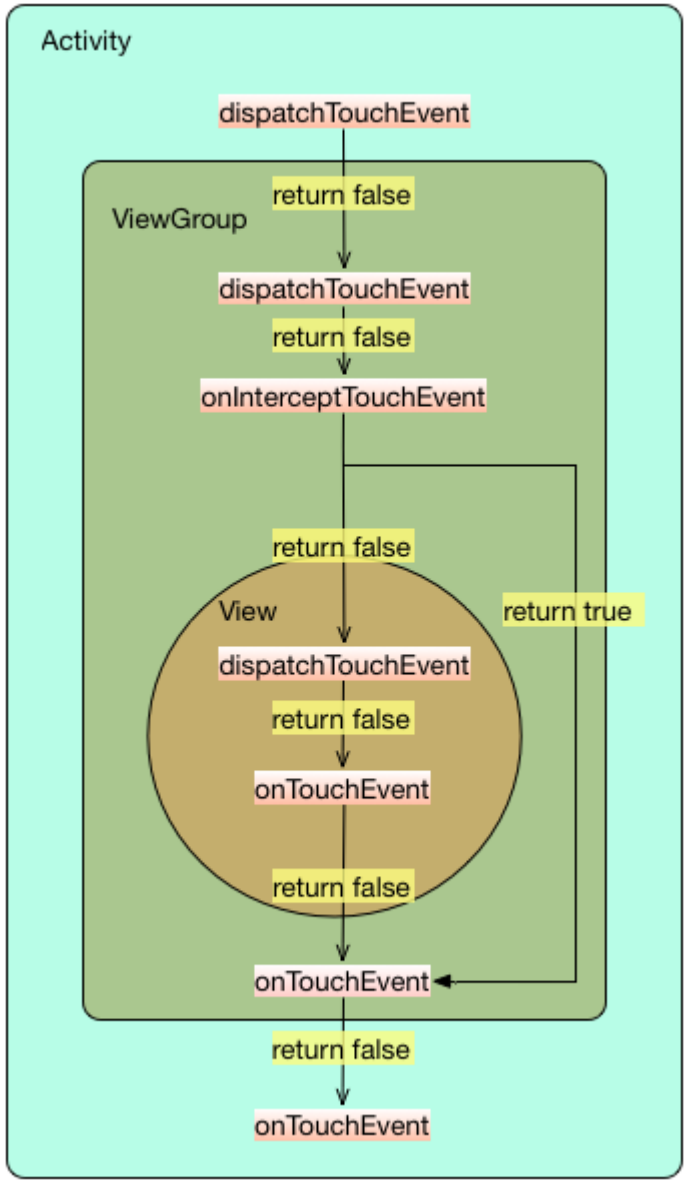
- **onFocusChange()**：在 View.OnFocusChangeListener 中。 当用户使用导航键或轨迹球导航到或远离项目时，将调用此方法。
- **onKey()**：在 View.OnKeyListener 中。 当用户聚焦于项目并按下或释放设备上的硬按键时，将调用此方法。
- **onTouch()**：在 View.OnTouchListener 中。 当用户执行可视为触摸事件的操作时，其中包括按下、释放或屏幕上的任何移动手势（在项目边界内），将调用此方法。
- **onCreateContextMenu()**：在 View.OnCreateContextMenuListener 中。 当（因持续“长按”而）生成上下文菜单时，将调用此方法。请参见菜单开发者指南中有关上下文菜单的阐述。

ANDROID事件处理

- onClick() 回调没有返回值，但是其他某些事件侦听器方法必须返回布尔值。具体原因取决于事件。
- 对于这几个事件侦听器，必须返回布尔值的原因如下：
- **onLongClick()**：此方法返回一个布尔值，表示您是否已处理完事件，以及是否应该将它继续传下去。 也就是说，返回 true 表示您已经处理事件且事件应就此停止；如果您尚未处理事件和/或事件应该继续传递给其他任何点击侦听器，则返回 false。
- **onKey()**：此方法返回一个布尔值，表示您是否已处理完事件，以及是否应该将它继续传下去。 也就是说，返回 true 表示您已经处理事件且事件应就此停止；如果您尚未处理事件和/或事件应该继续传递给其他任何按键侦听器，则返回 false。
- **onTouch()**： 此方法返回一个布尔值，表示侦听器是否处理完此事件。重要的是，此事件可以拥有多个分先后顺序的操作。 因此，如果在收到关闭操作事件时返回 false，则表示您并未处理完此事件，而且对其后续操作也不感兴趣。 因此，您无需执行事件内的任何其他操作，如手势或最终操作事件。

ANDROID事件处理

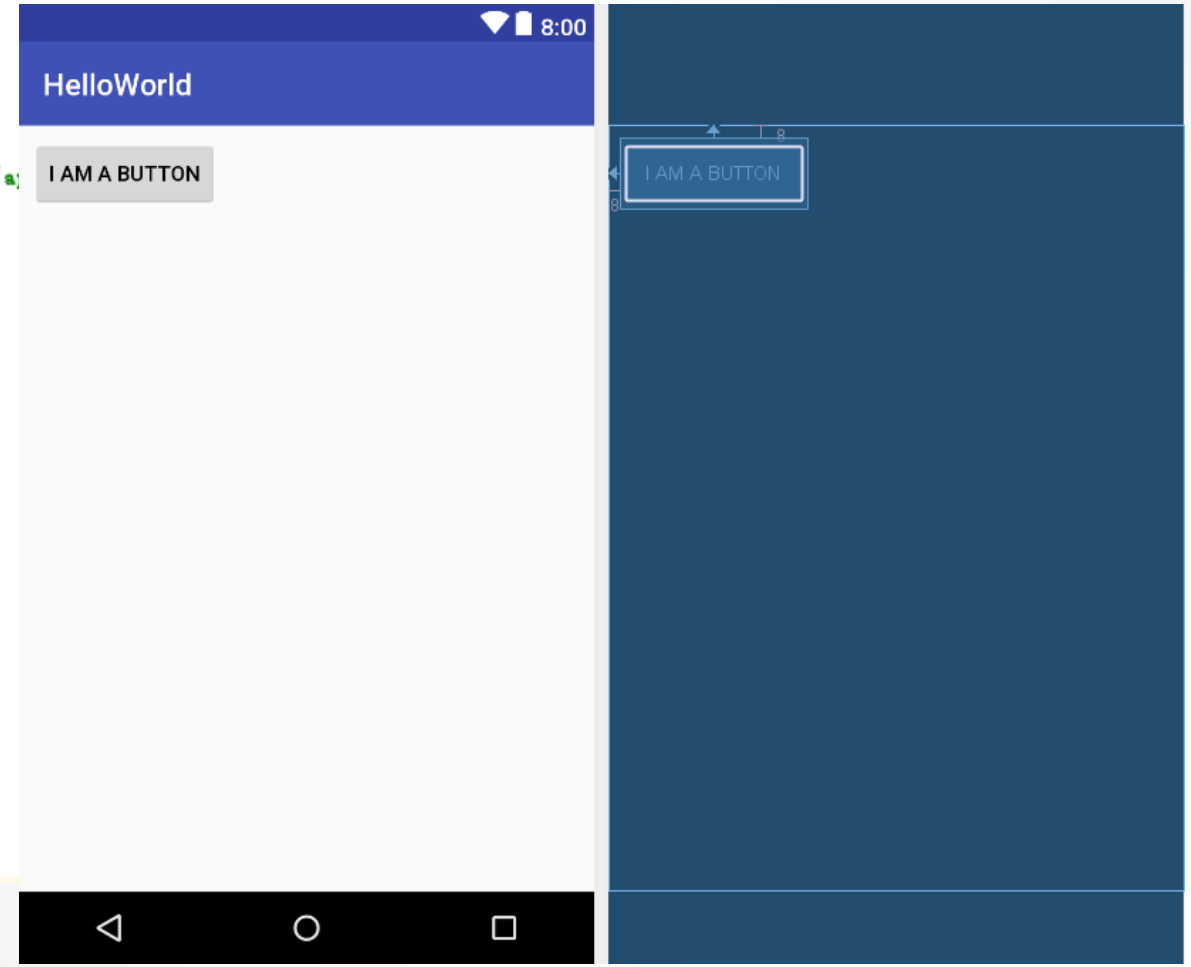




ANDROID事件处理

- 使用匿名内部类实现事件侦听：

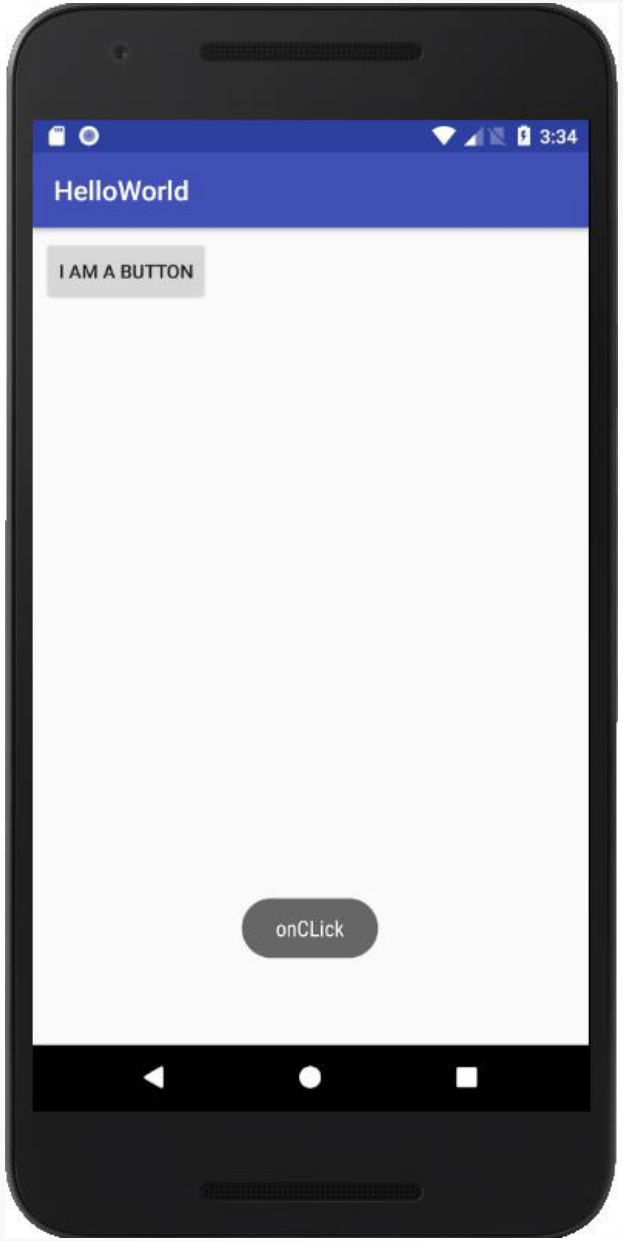
```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/a
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:layout_editor_absoluteY="81dp">

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="i am a button"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```



ANDROID事件处理

- 使用匿名内部类实现事件侦听：

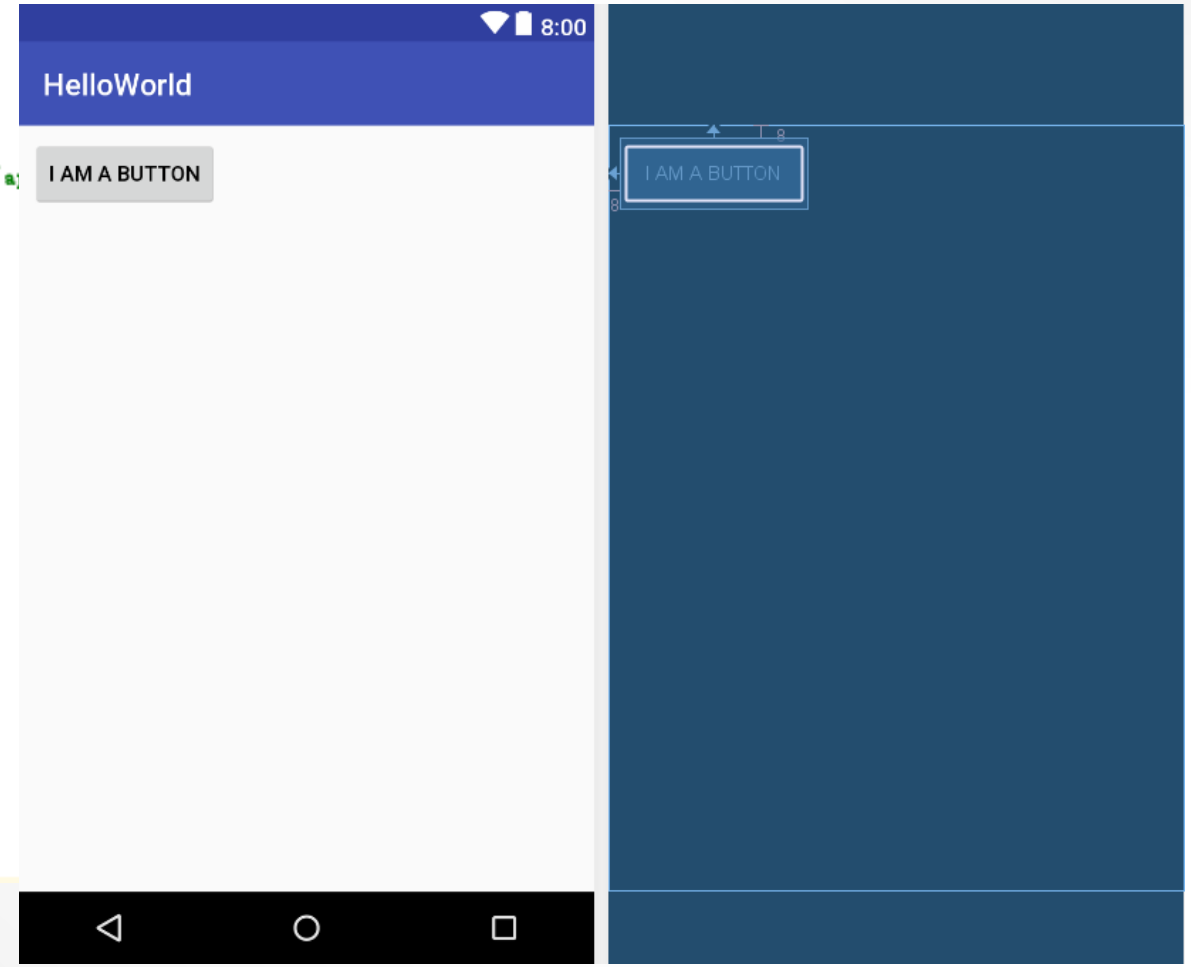
```
package com.example.bishop.helloworld;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button a = (Button)findViewById(R.id.button1);
        a.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Toast.makeText( context: MainActivity.this, text: "onCLick",Toast.LENGTH_LONG).show();
            }
        });
    }
}
```



ANDROID事件处理

- 使用Activity类实现事件侦听：

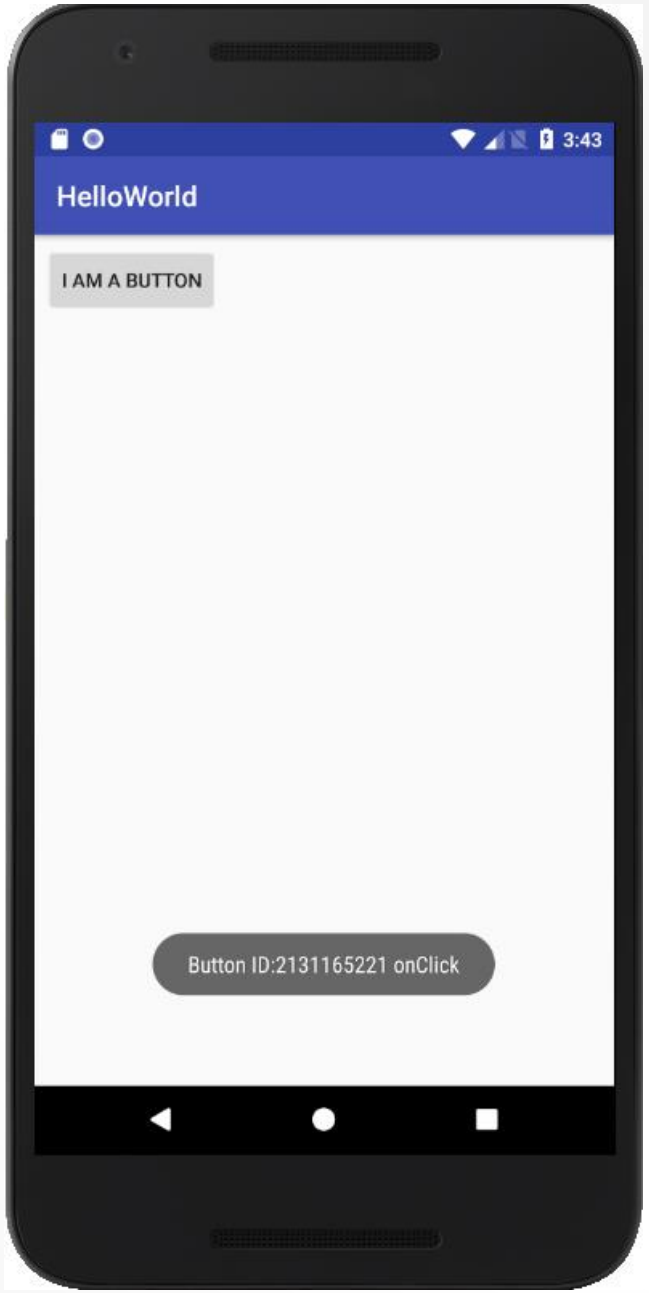
```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/a
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:layout_editor_absoluteY="81dp">

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="i am a button"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```



ANDROID事件处理

- 使用Activity类实现事件侦听：

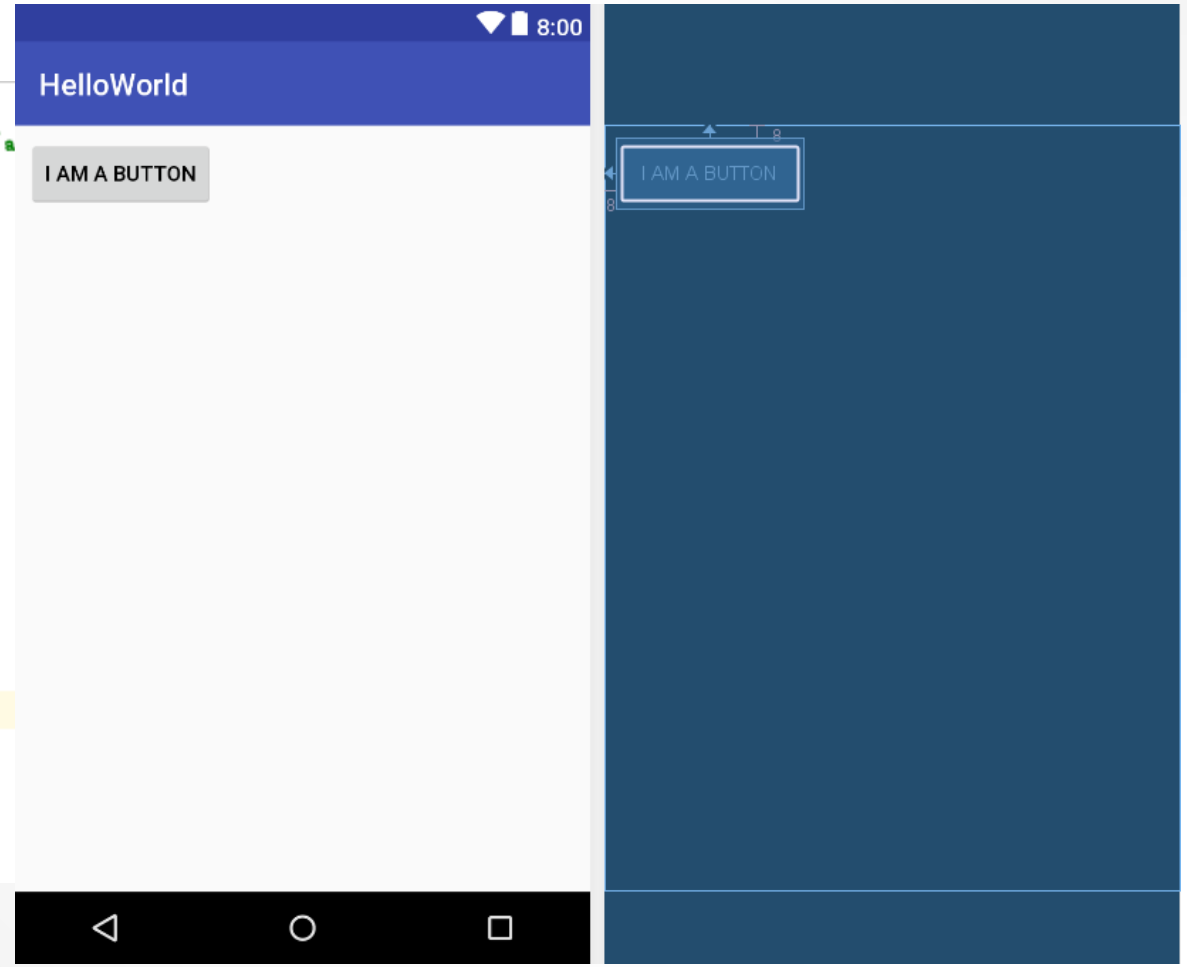
```java
package com.example.bishop.helloworld;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button a = (Button)findViewById(R.id.button1);
        a.setOnClickListener(this);
    }
     public void onClick(View v){
        Toast.makeText( context: this, text: "Button ID:"+v.getId()+" onClick",Toast.LENGTH_LONG).show();
        return;
     }
}
```



ANDROID事件处理

- 使用布局文件实现事件侦听：

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/a
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:layout_editor_absoluteY="81dp">

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="i am a button"
        android:onClick="onClick"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```



ANDROID事件处理

- 使用布局文件实现事件侦听：

```
package com.example.bishop.helloworld;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button a = (Button)findViewById(R.id.button1);
    }
     public void onClick(View v){
        Toast.makeText( context: this, text: "Button ID:"+v.getId()+" onClick",Toast.LENGTH_LONG).show();
        return;
     }
}
```

ANDROID事件处理

- 3种实现方法各有利弊：
- 使用匿名类需要编写额外代码，安全性最高。
- 使用Activity 类更为方便，可以避免加载额外的类和分配对象，但需要对View的ID进行识别，可能引入安全风险。
- 使用布局文件最为简单，但无法实现比较复杂的功能。
- 需要根据项目实际情况进行选择。

ANDROID SERVICE

- Service 是一个可以在后台执行长时间运行操作而不提供用户界面的应用组件。
- 服务可由其他应用组件启动，而且即使用户切换到其他应用，服务仍将在后台继续运行。
- 此外，组件可以绑定到服务，以与之进行交互，甚至是执行进程间通信 (IPC)。 例如，服务可以处理网络事务、播放音乐，执行文件 I/O 或与内容提供程序交互，而所有这一切均可在后台进行。

ANDROID SERVICE

- 服务分为两种状态：
- **Started（启动）**：当应用组件（如 Activity）通过调用 startService() 启动服务时，服务即处于“启动”状态。一旦启动，服务即可在后台无限期运行，即使启动服务的组件已被销毁也不受影响。 已启动的服务通常是执行单一操作，而且不会将结果返回给调用方。例如，它可能通过网络下载或上传文件。 操作完成后，服务会自行停止运行。
- **Bound（绑定）**：当应用组件通过调用 bindService() 绑定到服务时，服务即处于“绑定”状态。绑定服务提供了一个客户端-服务器接口，允许组件与服务进行交互、发送请求、获取结果，甚至是利用进程间通信 (IPC) 跨进程执行这些操作。 仅当与另一个应用组件绑定时，绑定服务才会运行。 多个组件可以同时绑定到该服务，但全部取消绑定后，该服务即会被销毁。

ANDROID SERVICE

- Android Service的回调方法：
- **onStartCommand()**：当另一个组件（如 Activity）通过调用 startService() 请求启动服务时，系统将调用此方法。一旦执行此方法，服务即会启动并可在后台无限期运行。 如果您实现此方法，则在服务工作完成后，需要由您通过调用 stopSelf() 或 stopService() 来停止服务。（如果您只想提供绑定，则无需实现此方法。）
- **onBind()**：当另一个组件想通过调用 bindService() 与服务绑定（例如执行 RPC）时，系统将调用此方法。在此方法的实现中，您必须通过返回 IBinder 提供一个接口，供客户端用来与服务进行通信。请务必实现此方法，但如果您并不希望允许绑定，则应返回 null。

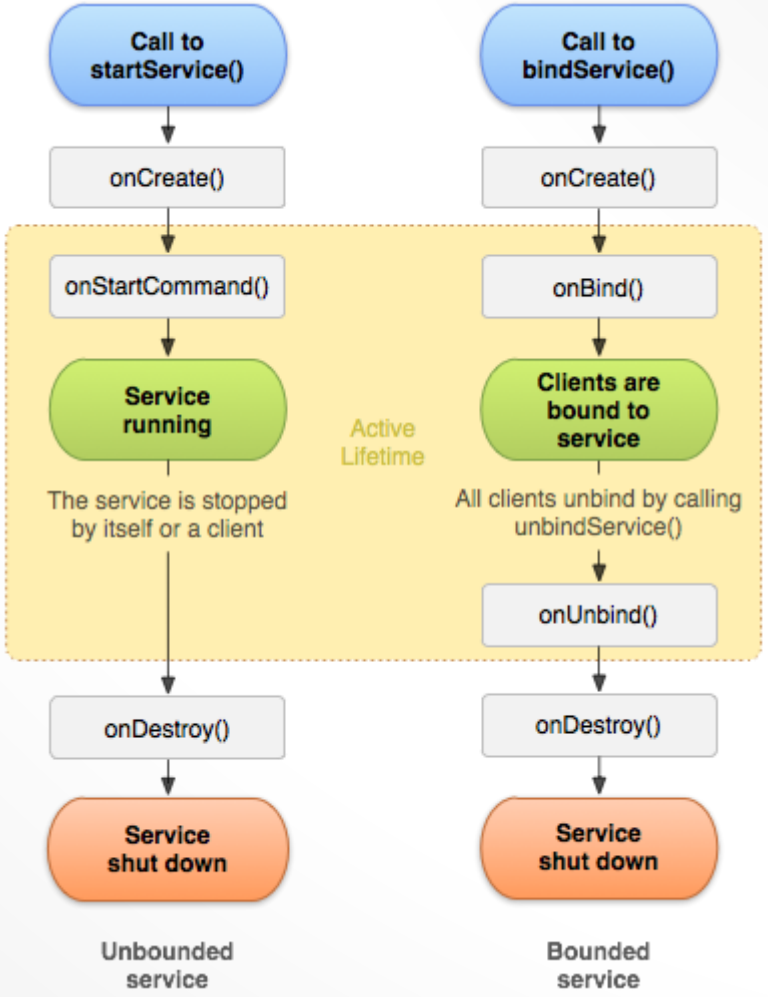
ANDROID SERVICE

- **onUnbind()**：当组件中断所有与服务绑定的连接时，系统调用该方法。
- **onRebind()**：当新的组件与服务绑定，且此前它已经通过onUnbind(Intent)通知断开连接时，系统调用该方法。
- **onCreate()**：首次创建服务时，系统将调用此方法来执行一次性设置程序（在调用 onStartCommand() 或 onBind() 之前）。如果服务已在运行，则不会调用此方法。
- **onDestroy()**：当服务不再使用且将被销毁时，系统将调用此方法。服务应该实现此方法来清理所有资源，如线程、注册的侦听器、接收器等。这是服务接收的最后一个调用。

ANDROID SERVICE

- Android Service的生命周期：

```
public class ExampleService extends Service {
    int mStartMode;       // indicates how to behave if the service is killed
    IBinder mBinder;      // interface for clients that bind
    boolean mAllowRebind; // indicates whether onRebind should be used

    @Override
    public void onCreate() {
        // The service is being created
    }
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        // The service is starting, due to a call to startService()
        return mStartMode;
    }
    @Override
    public IBinder onBind(Intent intent) {
        // A client is binding to the service with bindService()
        return mBinder;
    }
    @Override
    public boolean onUnbind(Intent intent) {
        // All clients have unbound with unbindService()
        return mAllowRebind;
    }
    @Override
    public void onRebind(Intent intent) {
        // A client is binding to the service with bindService(),
        // after onUnbind() has already been called
    }
    @Override
    public void onDestroy() {
        // The service is no longer used and is being destroyed
    }
}
```



ANDROID SERVICE

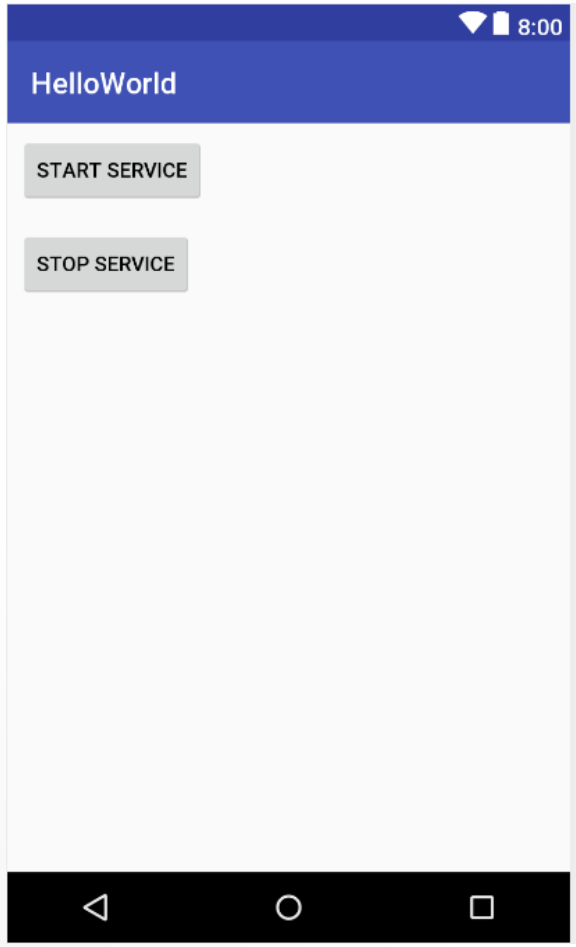
- 服务的整个生命周期从调用 onCreate() 开始起，到 onDestroy() 返回时结束。与 Activity 类似，服务也在 onCreate() 中完成初始设置，并在 onDestroy() 中释放所有剩余资源。例如，音乐播放服务可以在 onCreate() 中创建用于播放音乐的线程，然后在 onDestroy() 中停止该线程。
- 无论服务是通过 startService() 还是 bindService() 创建，都会为所有服务调用 onCreate() 和 onDestroy() 方法。

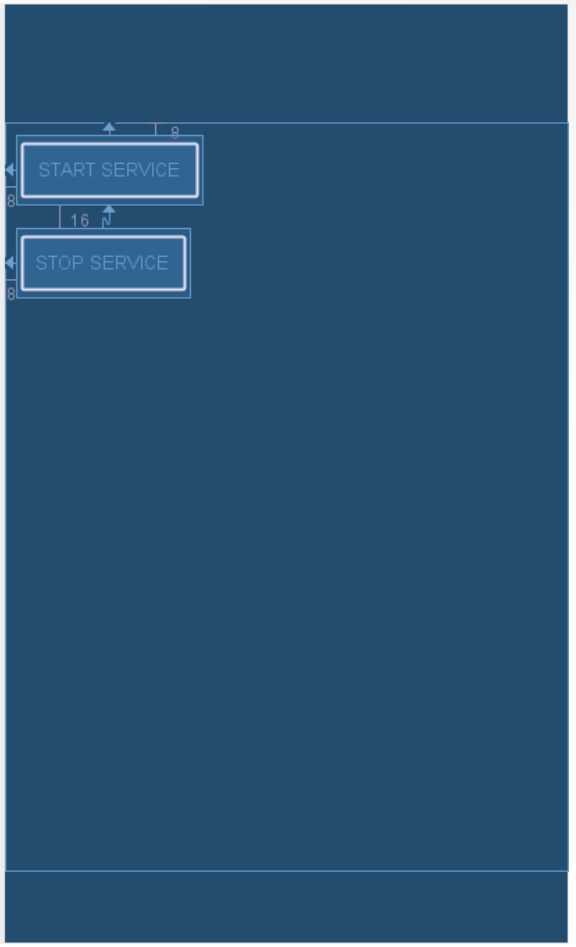
ANDROID SERVICE

- 服务的有效生命周期从调用 onStartCommand() 或 onBind() 方法开始。每种方法均有 {Intent 对象，该对象分别传递到 startService() 或 bindService()。
- 对于启动服务，有效生命周期与整个生命周期同时结束（即便是在 onStartCommand() 返回之后，服务仍然处于活动状态）。对于绑定服务，有效生命周期在 onUnbind() 返回时结束。

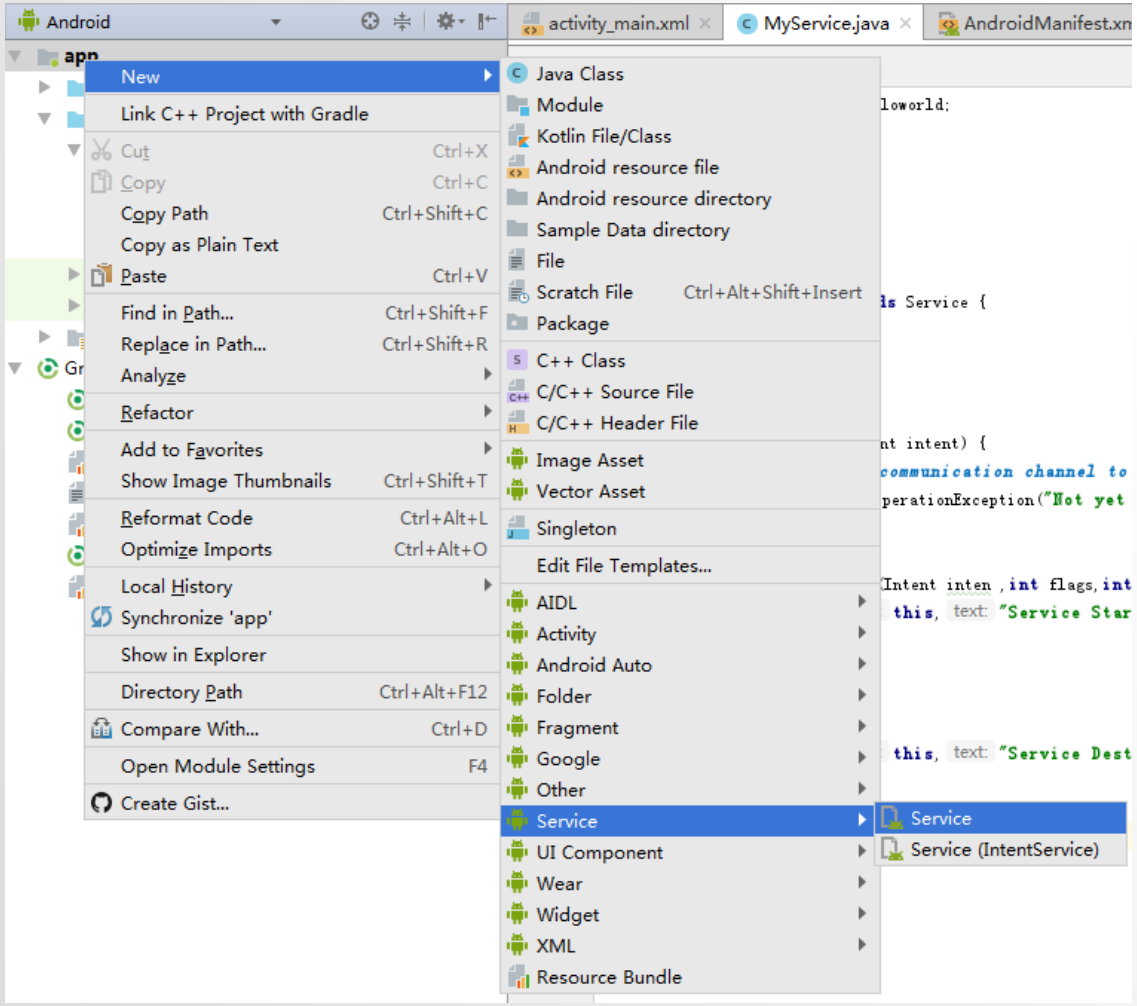
ANDROID SERVICE

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:layout_editor_absoluteY="81dp">

    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="16dp"
        android:text="stop service"
        android:onClick="stopService"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/button2" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:onClick="startService"
        android:text="Start service"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```





ANDROID SERVICE



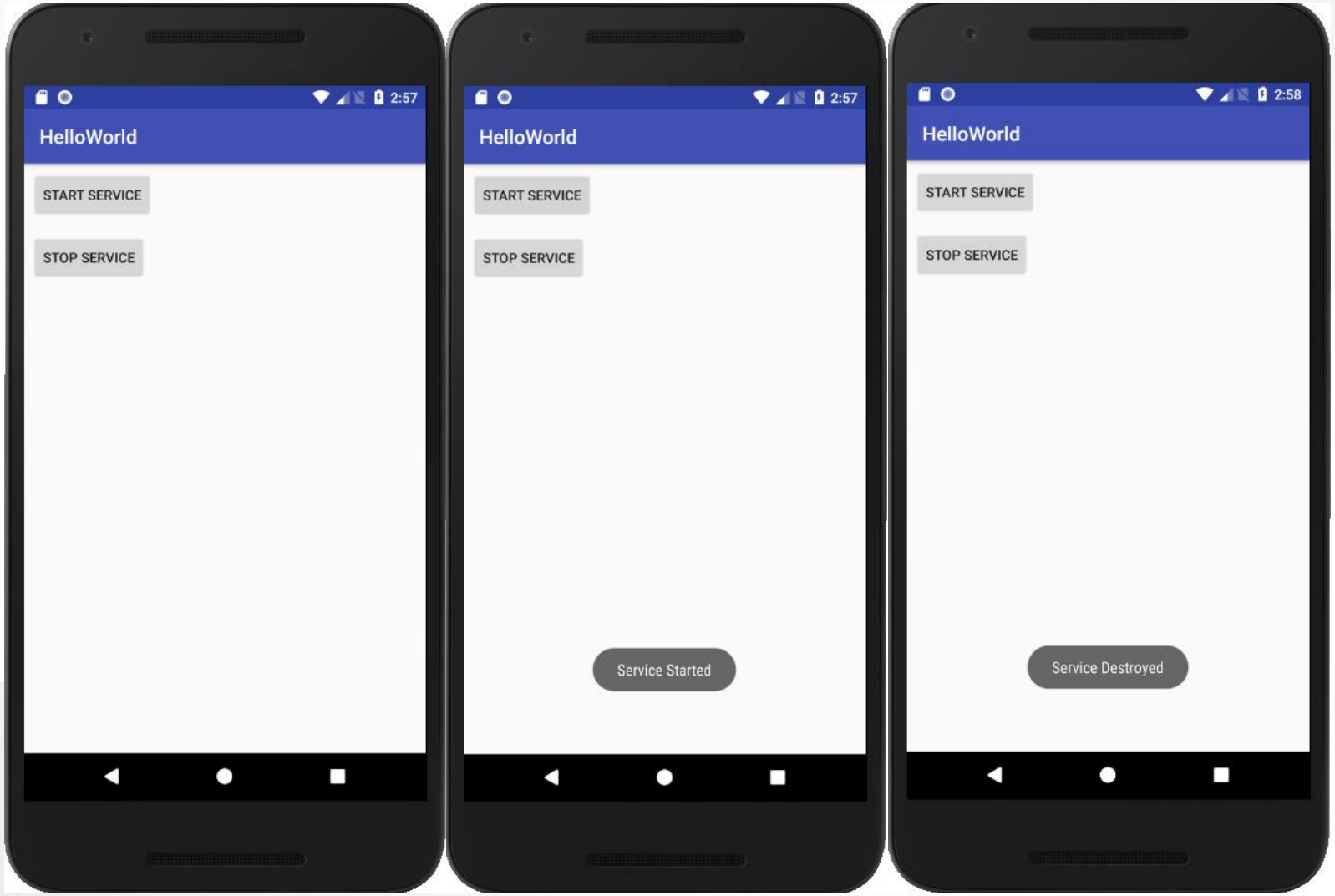

```java
package com.example.bishop.helloworld;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.widget.Toast;

public class MyService extends Service {
    public MyService() {
    }

    @Override
    public IBinder onBind(Intent intent) {
        // TODO: Return the communication channel to the service.
        throw new UnsupportedOperationException("Not yet implemented");
    }
    @Override
    public int onStartCommand(Intent inten ,int flags,int startId){
        Toast.makeText( context: this, text: "Service Started",Toast.LENGTH_LONG).show();
        return START_STICKY;
    }
    public void onDestroy(){
        super.onDestroy();
        Toast.makeText( context: this, text: "Service Destroyed",Toast.LENGTH_LONG).show();
    }
}
```

ANDROID SERVICE

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.bishop.helloworld">

    <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=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service
            android:name=".MyService"
            android:enabled="true"
            android:exported="true"></service>
    </application>
</manifest>
```

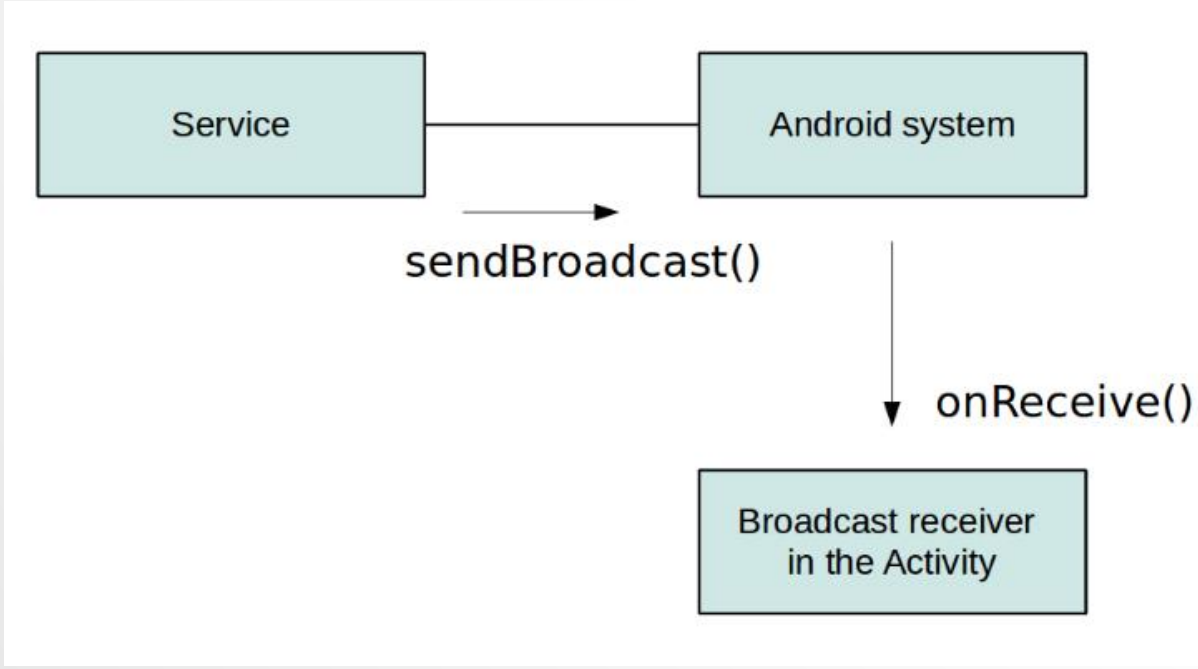
ANDROID SERVICE

```java
package com.example.bishop.helloworld;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void startService(View view){
        startService(new Intent(getBaseContext(),MyService.class));
    }
    public void stopService(View view){
        stopService(new Intent(getBaseContext(),MyService.class));
    }
}
```



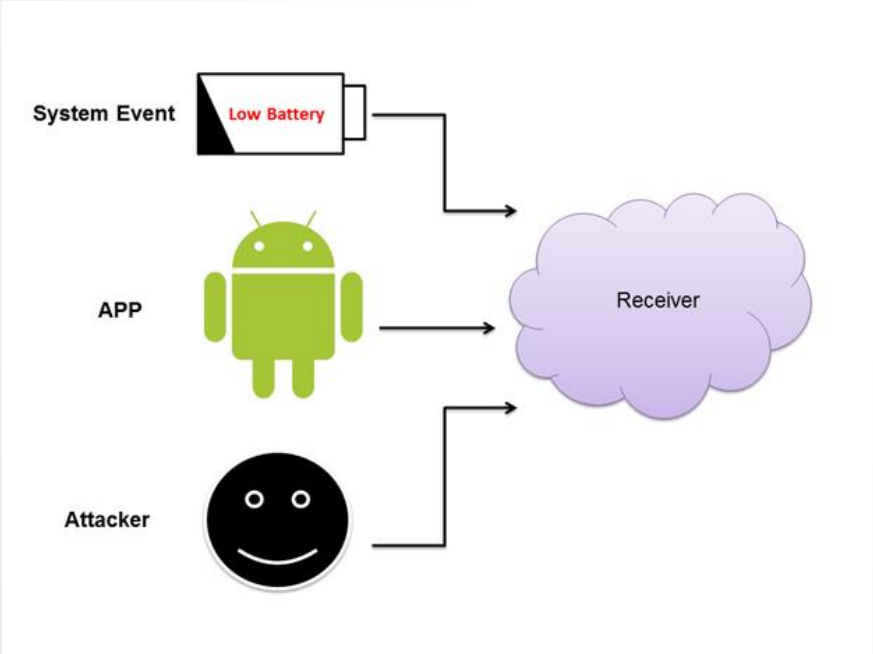
BROADCAST RECEIVER

- Broadcast Receiver本质上是一个全局的监听器，属于Android四大组件之一。
- Android广播分为两个方面：广播发送者、广播接收者（BroadcastReceiver）。



BROADCAST RECEIVER

- Broadcast Receiver用于监听（接收）应用发出的广播消息，并做出响应。
- 不同组件之间通信（包括应用内 / 不同应用之间）；
- Android系统在特定情况下与App之间的消息通信;
- 多线程通信。

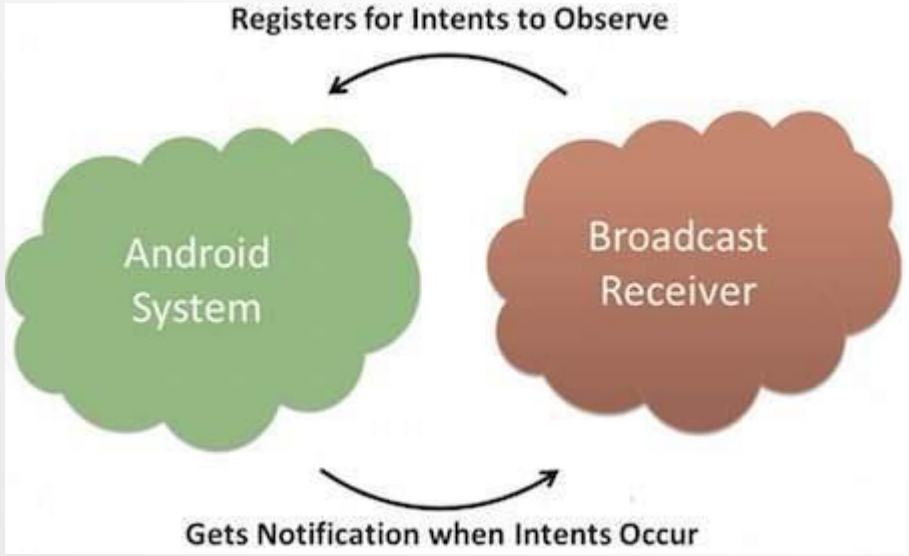


BROADCAST RECEIVER

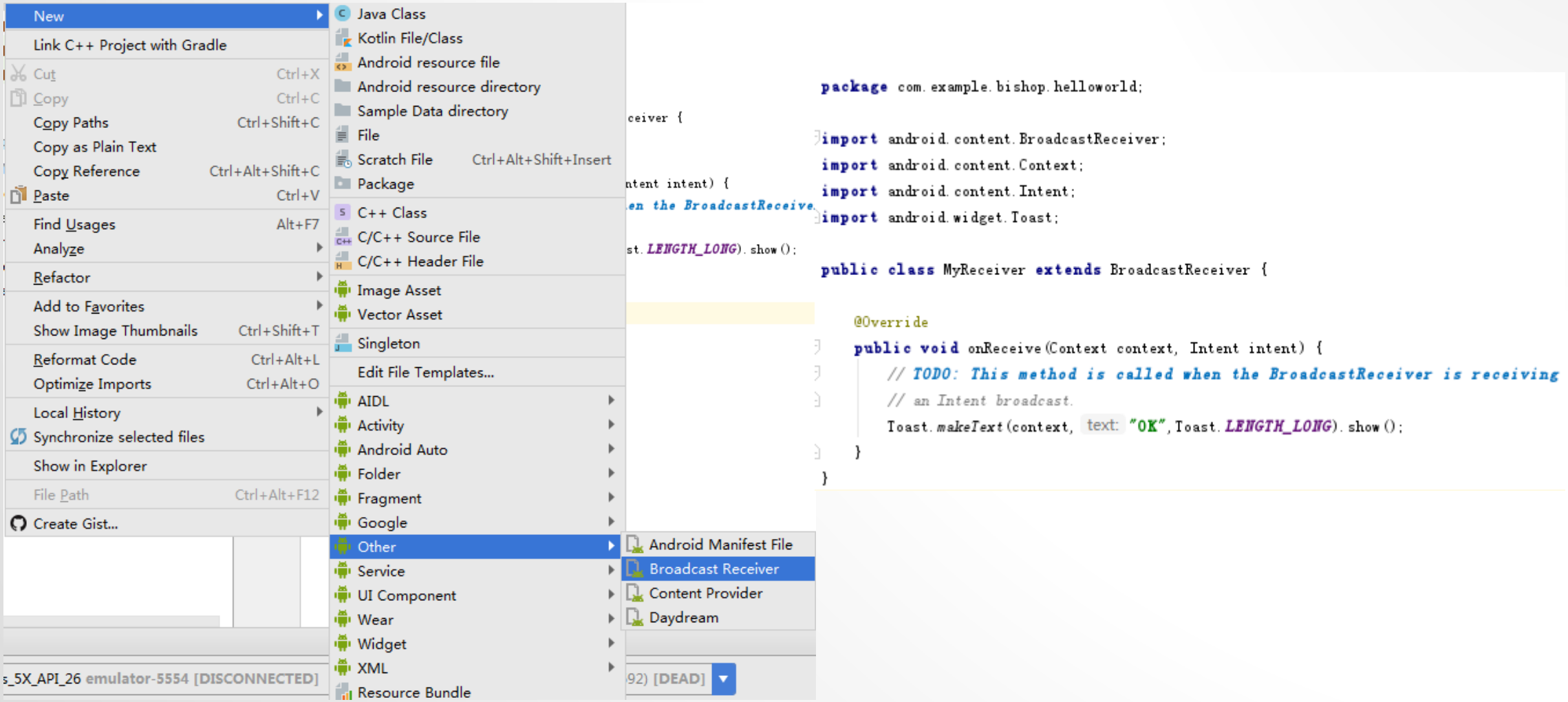
- 自定义广播接收者BroadcastReceiver子类，并复写onRecvice（）方法；
- 通过Binder机制向AMS（Activity Manager Service）进行注册；
- 广播发送者通过Binder机制向AMS发送广播；
- AMS查找符合相应条件（IntentFilter/Permission等）的BroadcastReceiver，将广播发送到BroadcastReceiver（一般情况下是Activity）相应的消息循环队列中；
- 消息循环执行拿到此广播，回调BroadcastReceiver中的onReceive()方法。

BROADCAST RECEIVER

- 广播接收器接收到相应广播后，会自动回调onReceive()方法。
- 一般情况下，onReceive方法会涉及与其他组件之间的交互，如发送Notification、启动service等。
- 默认情况下，广播接收器运行在UI线程，因此，onReceive方法不能执行耗时操作，否则将导致ANR（Application Not Responding）。



BROADCAST RECEIVER



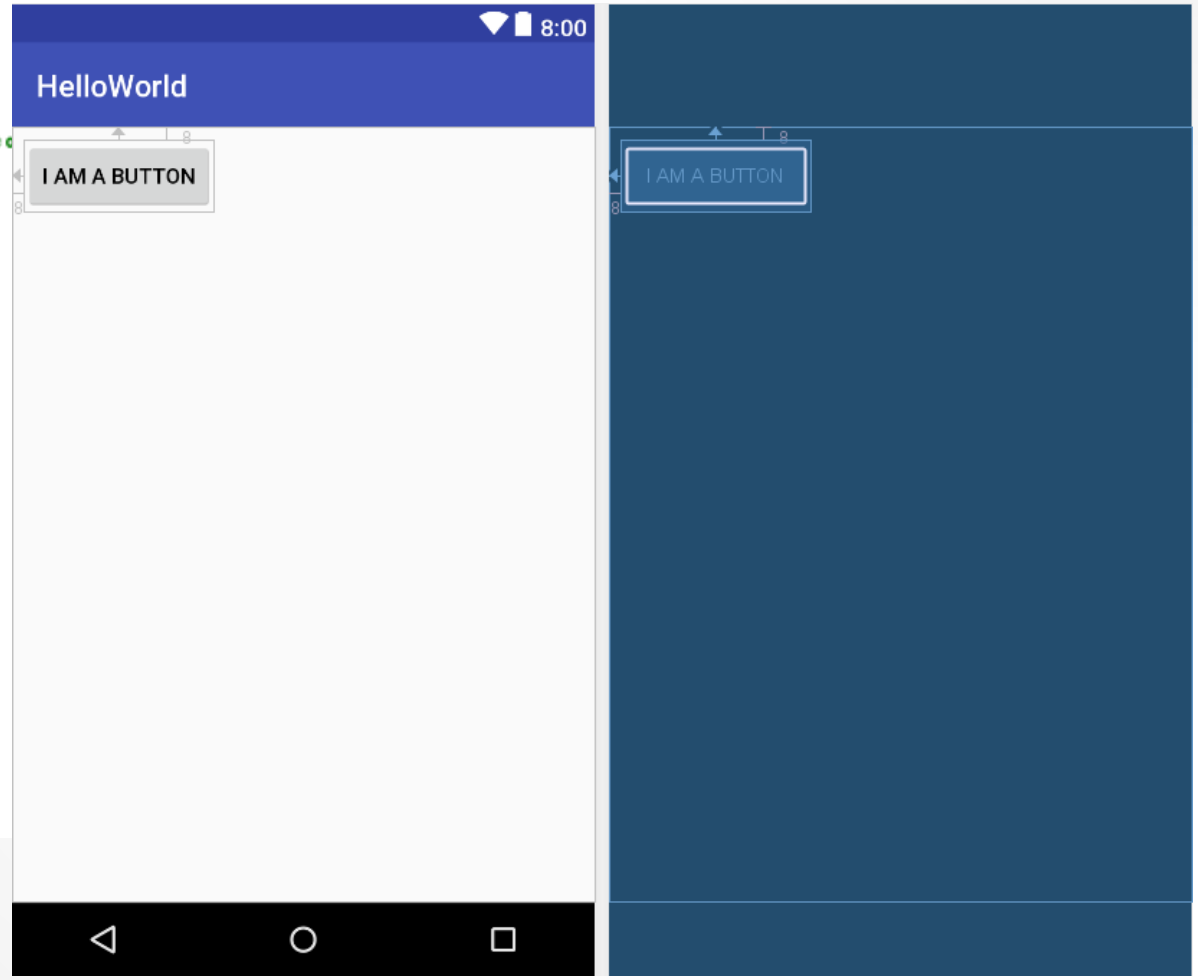
BROADCAST RECEIVER

- 注册的方式分为两种：
- 1，静态注册，在AndroidManifest.xml里通过标签声明。

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.bishop.helloworld">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="HelloWorld"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <receiver android:name=".MyReceiver">
            <intent-filter>
                <action android:name="com.example.DEMO"/>
            </intent-filter>
        </receiver>
    </application>
</manifest>
```

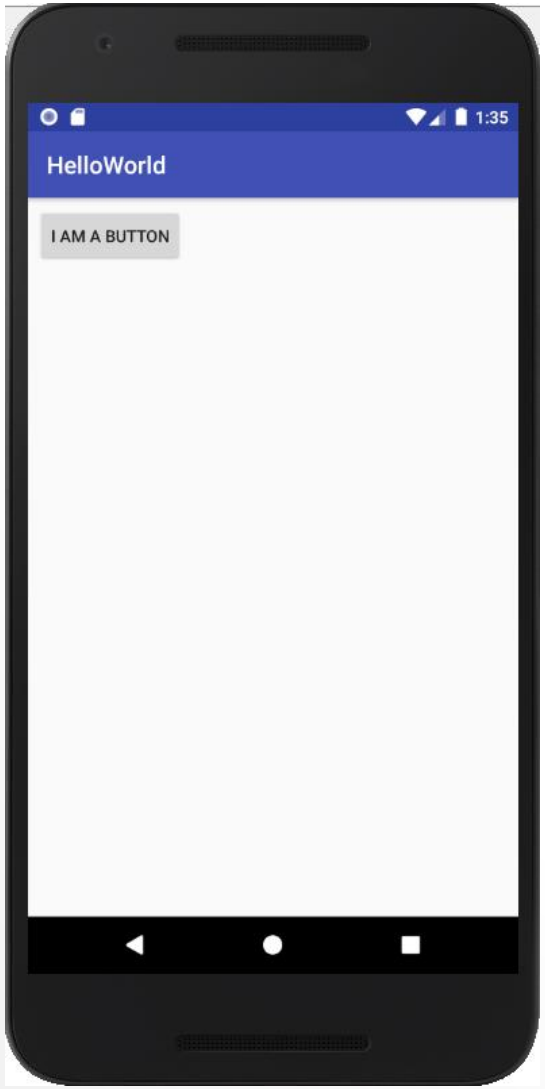
BROADCAST RECEIVER

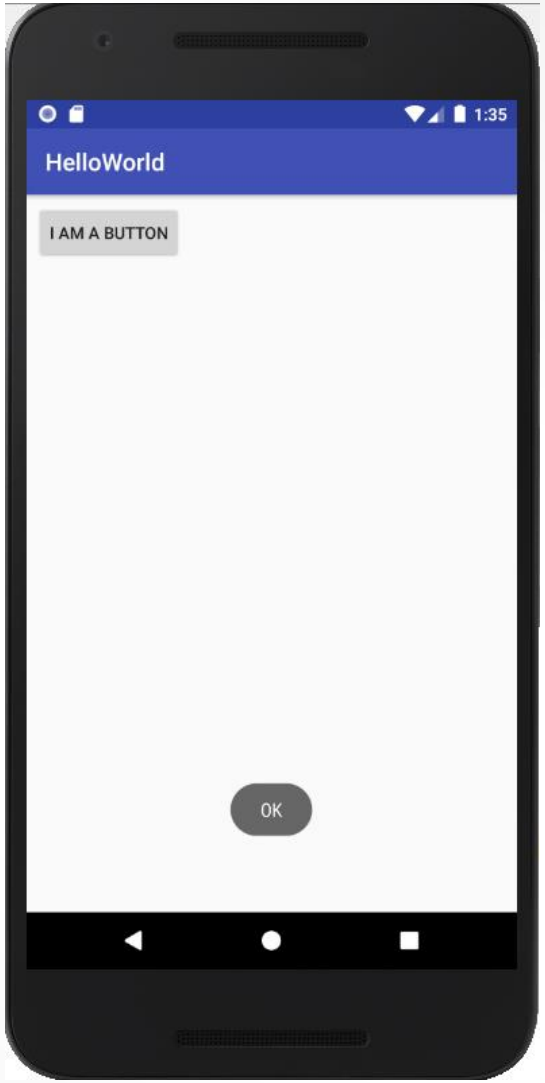
```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.co
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:layout_editor_absoluteY="81dp">

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="i am a button"
        android:onClick="onClick"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
```



BROADCAST RECEIVER

```java
package com.example.bishop.helloworld;

import android.content.IntentFilter;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.content.Intent;

public class MainActivity extends AppCompatActivity {
    //MyReceiver receiver = new MyReceiver();
    //IntentFilter filter = new IntentFilter();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //filter.addAction(Intent.ACTION_BATTERY_LOW);
        //registerReceiver(receiver, filter);
    }
    @Override
    protected void onDestroy(){
        super.onDestroy();
        //unregisterReceiver(receiver);
    }

     public void onClick(View v){
        Intent intent= new Intent( action: "com.example.DEMO");
        sendBroadcast(intent);
     }
}
```





BROADCAST RECEIVER

- 2，动态注册，在代码中通过调用Context的registerReceiver（）方法进行动态注册BroadcastReceiver。

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.bishop.helloworld">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="HelloWorld"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

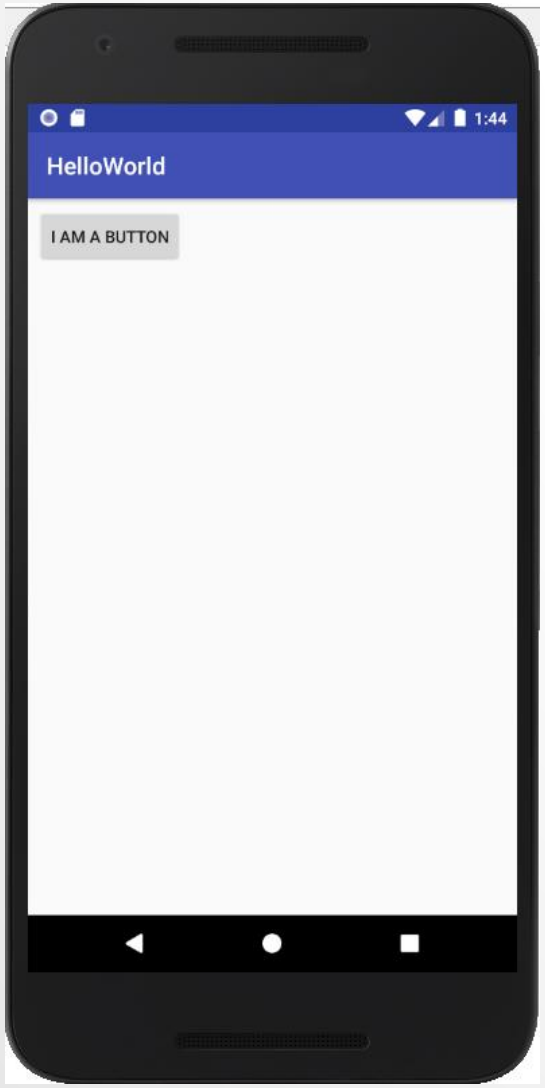
```java
package com.example.bishop.helloworld;

import android.content.IntentFilter;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.content.Intent;

public class MainActivity extends AppCompatActivity {
     MyReceiver receiver = new MyReceiver();
     IntentFilter filter = new IntentFilter();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        filter.addAction("com.example.DEMO");
        registerReceiver(receiver,filter);
    }
    @Override
    protected void onDestroy(){
        super.onDestroy();
        unregisterReceiver(receiver);
    }

     public void onClick(View v){
        Intent intent= new Intent( action: "com.example.DEMO");
        sendBroadcast(intent);
     }
}
```

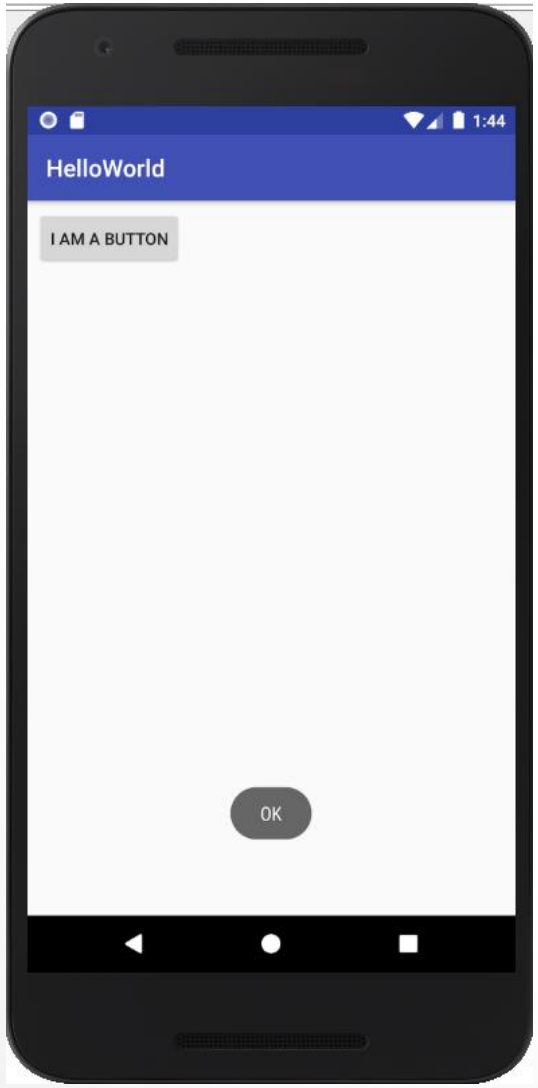
BROADCAST RECEIVER

- Android Oreo已限制App在后台接收Intent广播信息，需要修改build.gradle中targetSdkVersion的值，使其小于等于25。

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 26
    defaultConfig {
        applicationId "com.example.bishop.helloworld"
        minSdkVersion 18
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:26.1.0'
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.1'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.1'
}
```

BROADCAST RECEIVER





BROADCAST RECEIVER

- 动态广播最好在Activity的onResume()注册、onPause()注销。
- 在onResume()注册、onPause()注销是因为onPause()在App死亡前一定会被执行，从而保证广播在App死亡前一定会被注销，从而防止内存泄露。
- 不在onCreate() & onDestory() 或 onStart() & onStop()注册、注销是因为： 当系统因为内存不足要回收Activity占用的资源时，Activity在执行完onPause()方法后就会被销毁，有些生命周期方法onStop()，onDestory()就不会执行。当再回到此Activity时，是从onCreate方法开始执行。
- 假设我们将广播的注销放在onStop()，onDestory()方法里的话，有可能在Activity被销毁后还未执行onStop()，onDestory()方法，即广播仍还未注销，从而导致内存泄露。
- 但是，onPause()一定会被执行，从而保证了广播在App死亡前一定会被注销，从而防止内存泄露。

BROADCAST RECEIVER

注册方式	特点	应用场景
静态注册 (常驻广播)	常驻，不受任何组件的生命周期影响 (应用程序关闭后，如果有信息广播来，程序依旧会被系统调用) 缺点：耗电、占内存	需要时刻监听广播
动态注册 (非常驻广播)	非常驻，灵活，跟随组件的生命周期变化 (组件结束=广播结束，在组件结束前，必须移除广播接收器)	需要特定时刻监听广播

BROADCAST RECEIVER

- Android中内置了多个系统广播（System Broadcast），只要涉及到手机的基本操作（如开机、网络状态变化、拍照等等），都会发出相应的广播。
- 当使用系统广播时，只需要在注册广播接收者时定义相关的action即可，并不需要手动发送广播，当系统有相关操作时会自动进行系统广播。
- 每个广播都有特定的Intent - Filter（包括具体的action），Android系统广播action如下：

BROADCAST RECEIVER

系统操作	action
关闭或打开飞行模式	Intent.ACTION_AIRPLANE_MODE_CHANGED
充电时或电量发生变化	Intent.ACTION_BATTERY_CHANGED
电池电量低	Intent.ACTION_BATTERY_LOW
电池电量充足（即从电量低变化到饱满时会发出广播	Intent.ACTION_BATTERY_OKAY
系统启动完成后(仅广播一次)	Intent.ACTION_BOOT_COMPLETED
检测网络变化	ConnectivityManager.CONNECTIVITY_ACTION
按下照相时的拍照按键(硬件按键)时	Intent.ACTION_CAMERA_BUTTON
屏幕锁屏	Intent.ACTION_CLOSE_SYSTEM_DIALOGS

BROADCAST RECEIVER

设备当前设置被改变时(界面语言、设备方向等)	Intent.ACTION_CONFIGURATION_CHANGED
插入耳机时	Intent.ACTION_HEADSET_PLUG
未正确移除SD卡但已取出来时(正确移除方法:设置–SD卡和设备内存–卸载SD卡)	Intent.ACTION_MEDIA_BAD_REMOVAL
插入外部储存装置（如SD卡)	Intent.ACTION_MEDIA_CHECKING
成功安装APK	Intent.ACTION_PACKAGE_ADDED
成功删除APK	Intent.ACTION_PACKAGE_REMOVED
重启设备	Intent.ACTION_REBOOT
屏幕被关闭	Intent.ACTION_SCREEN_OFF
屏幕被打开	Intent.ACTION_SCREEN_ON
关闭系统时	Intent.ACTION_SHUTDOWN
重启设备	Intent.ACTION_REBOOT

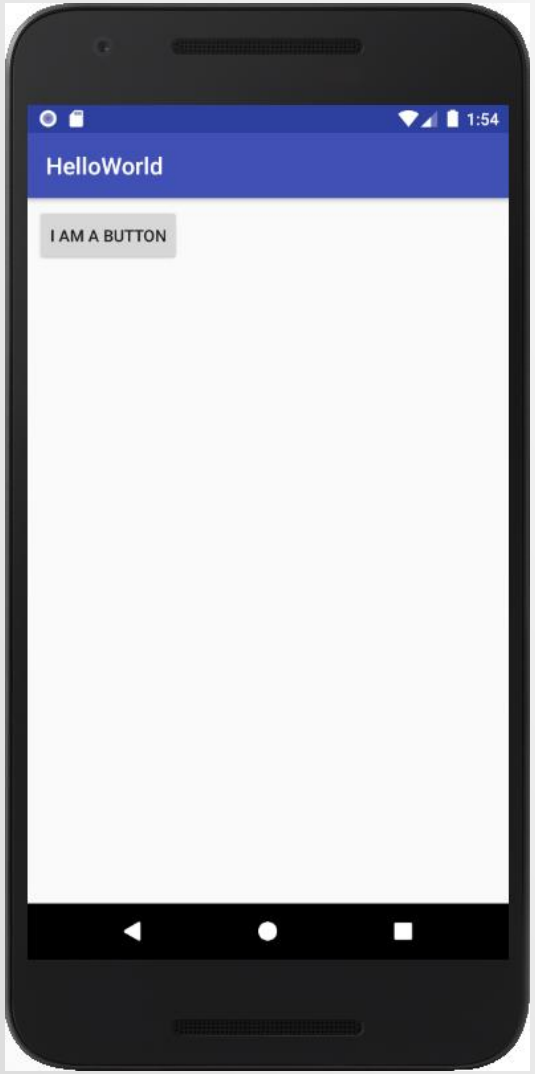
BROADCAST RECEIVER

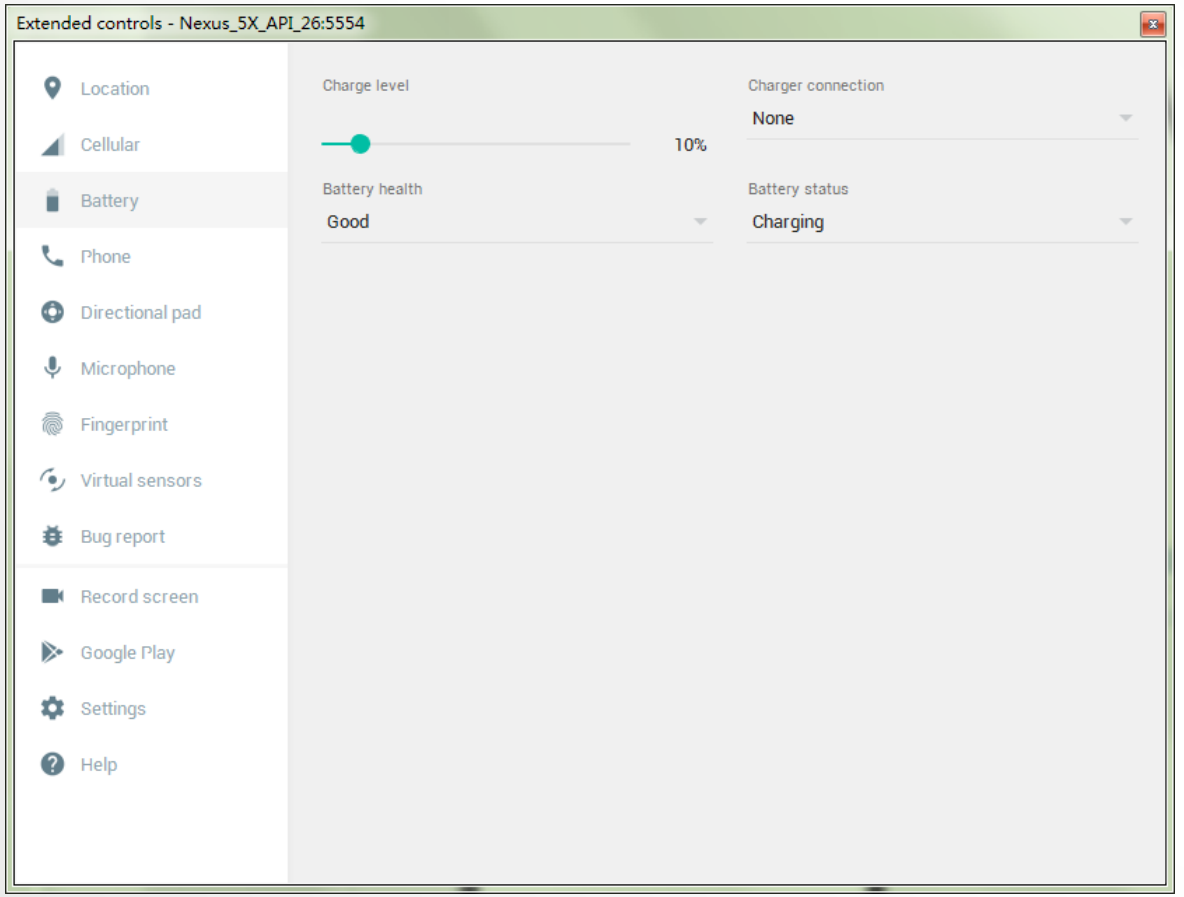
- 静态注册系统广播接收器：

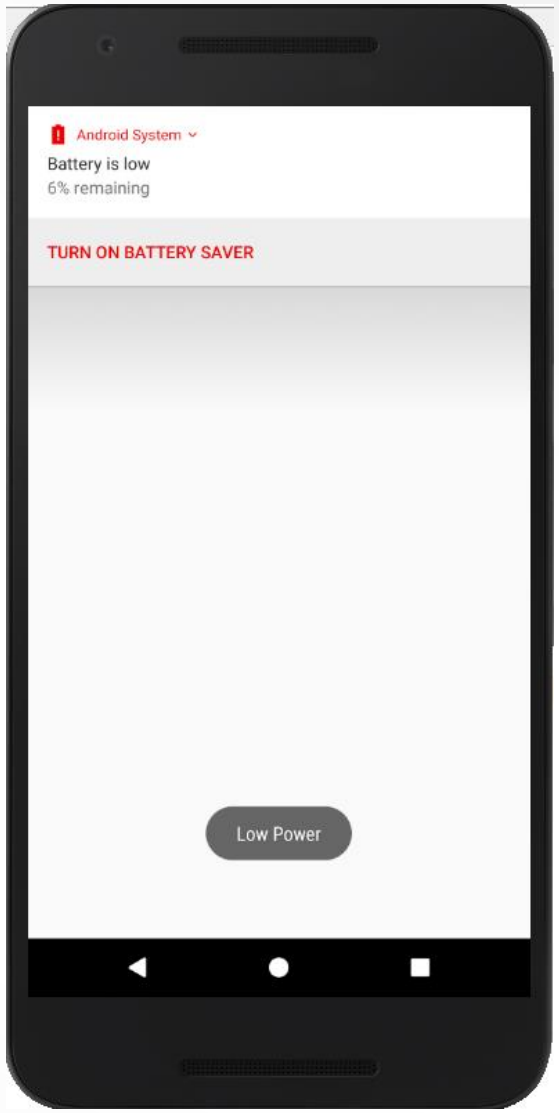
```java
package com.example.bishop.helloworld;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.Toast;

public class MyReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        // TODO: This method is called when the BroadcastReceiver is receiving
        // an Intent broadcast.
        Toast.makeText(context, text: "Low Power", Toast.LENGTH_LONG).show();
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.bishop.helloworld">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="HelloWorld"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <receiver android:name=".MyReceiver">
            <intent-filter>
                <action android:name="android.intent.action.BATTERY_LOW"/>
            </intent-filter>
        </receiver>
    </application>
</manifest>
```

BROADCAST RECEIVER







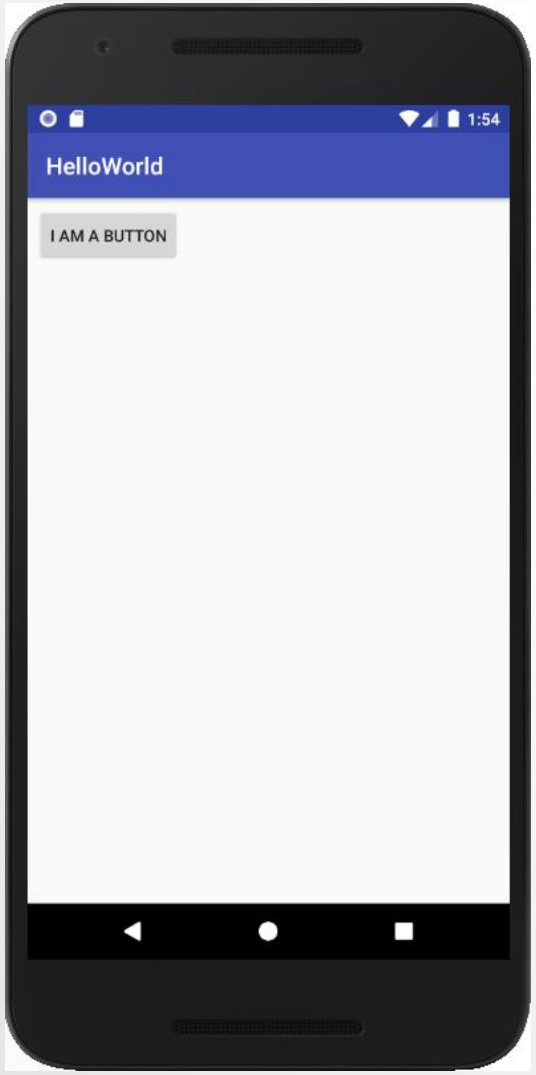
BROADCAST RECEIVER

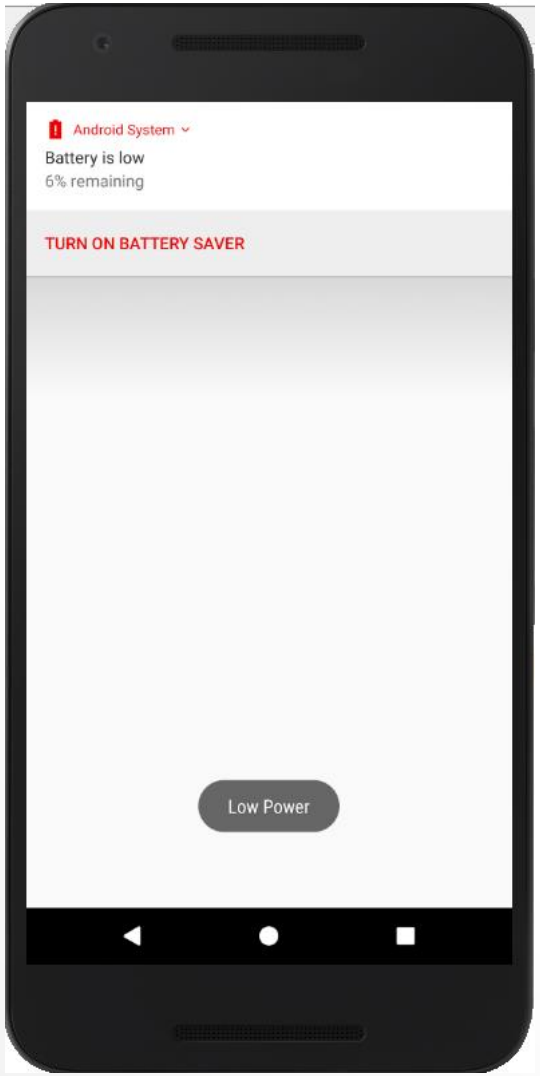
- 动态注册系统广播接收器：

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.bishop.helloworld">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="HelloWorld"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

```java
package com.example.bishop.helloworld;

import android.content.IntentFilter;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.content.Intent;

public class MainActivity extends AppCompatActivity {
     MyReceiver receiver = new MyReceiver();
     IntentFilter filter = new IntentFilter();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        filter.addAction(Intent.ACTION_BATTERY_LOW);
        registerReceiver(receiver, filter);
    }
    @Override
    protected void onDestroy(){
        super.onDestroy();
        unregisterReceiver(receiver);
    }

     public void onClick(View v){
        //Intent intent= new Intent("com.example.DEMO");
        //sendBroadcast(intent);
     }
}
```

BROADCAST RECEIVER



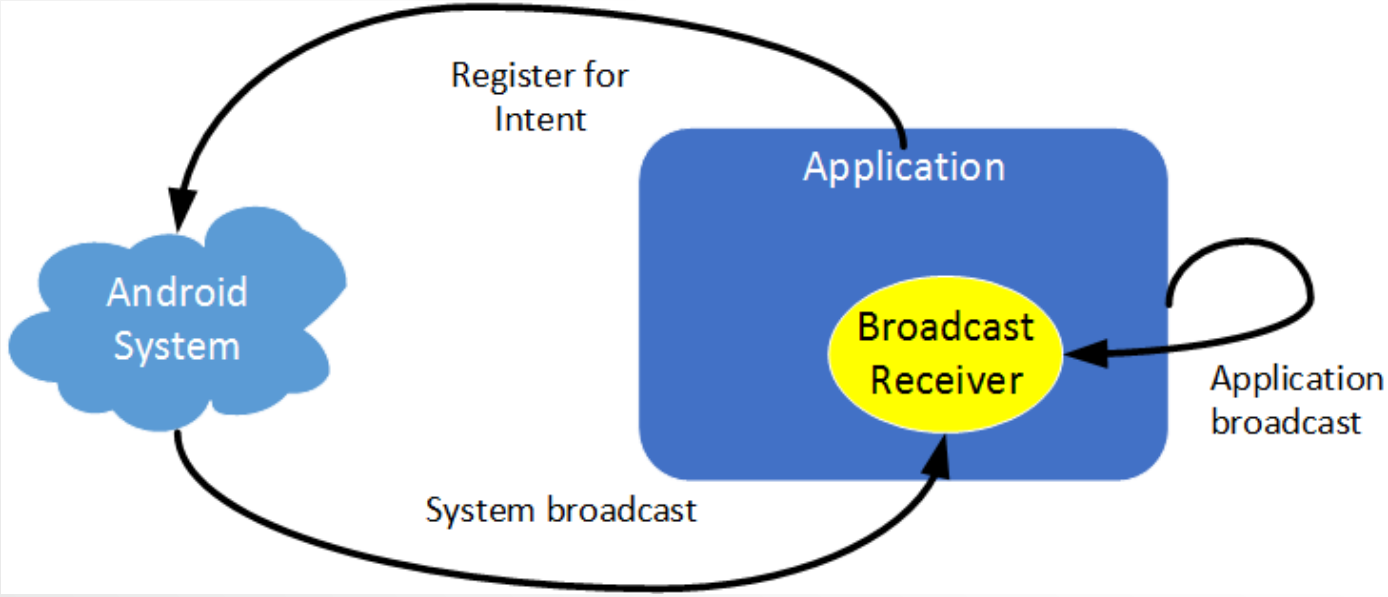


BROADCAST RECEIVER

- Android中的广播可以跨App直接通信（exported对于有intent-filter情况下默认值为true）。可能出现的问题：
- 其他App针对性发出与当前App intent-filter相匹配的广播，由此导致当前App不断接收广播并处理；
- 其他App注册与当前App一致的intent-filter用于接收广播，获取广播具体信息。

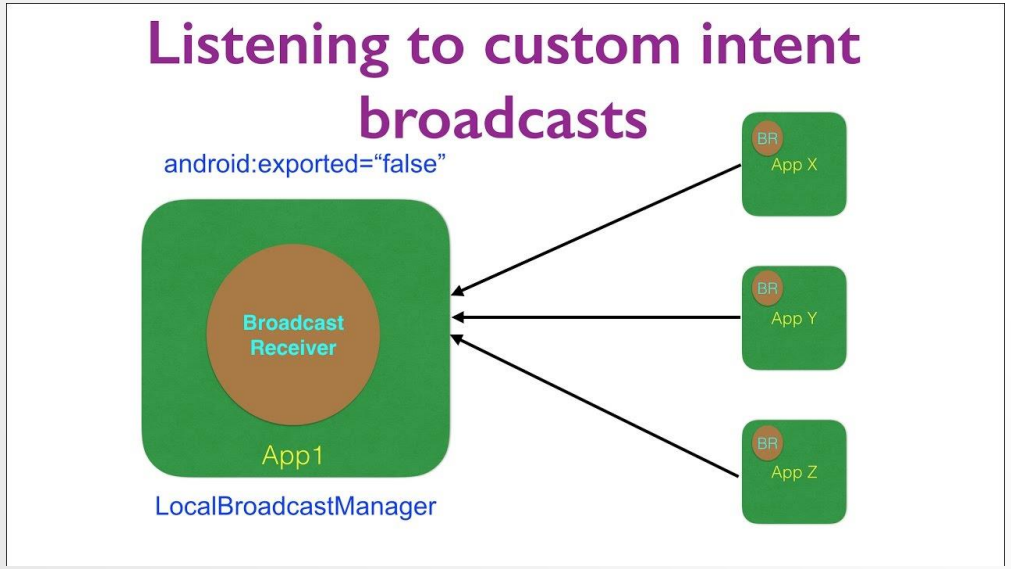
BROADCAST RECEIVER

- 使用App应用内广播（Local Broadcast）：
- App应用内广播可理解为一种局部广播，广播的发送者和接收者都同属于一个App。
- 相比于全局广播（普通广播），App应用内广播优势体现在：安全性高 & 效率高。



BROADCAST RECEIVER

- 注册广播时将exported属性设置为false，使得非本App内部发出的此广播不被接收；
- 在广播发送和接收时，增设相应权限permission，用于权限验证；
- 发送广播时指定该广播接收器所在的包名，此广播将只会发送到此包中的App内与之相匹配的有效广播接收器中。



BROADCAST RECEIVER

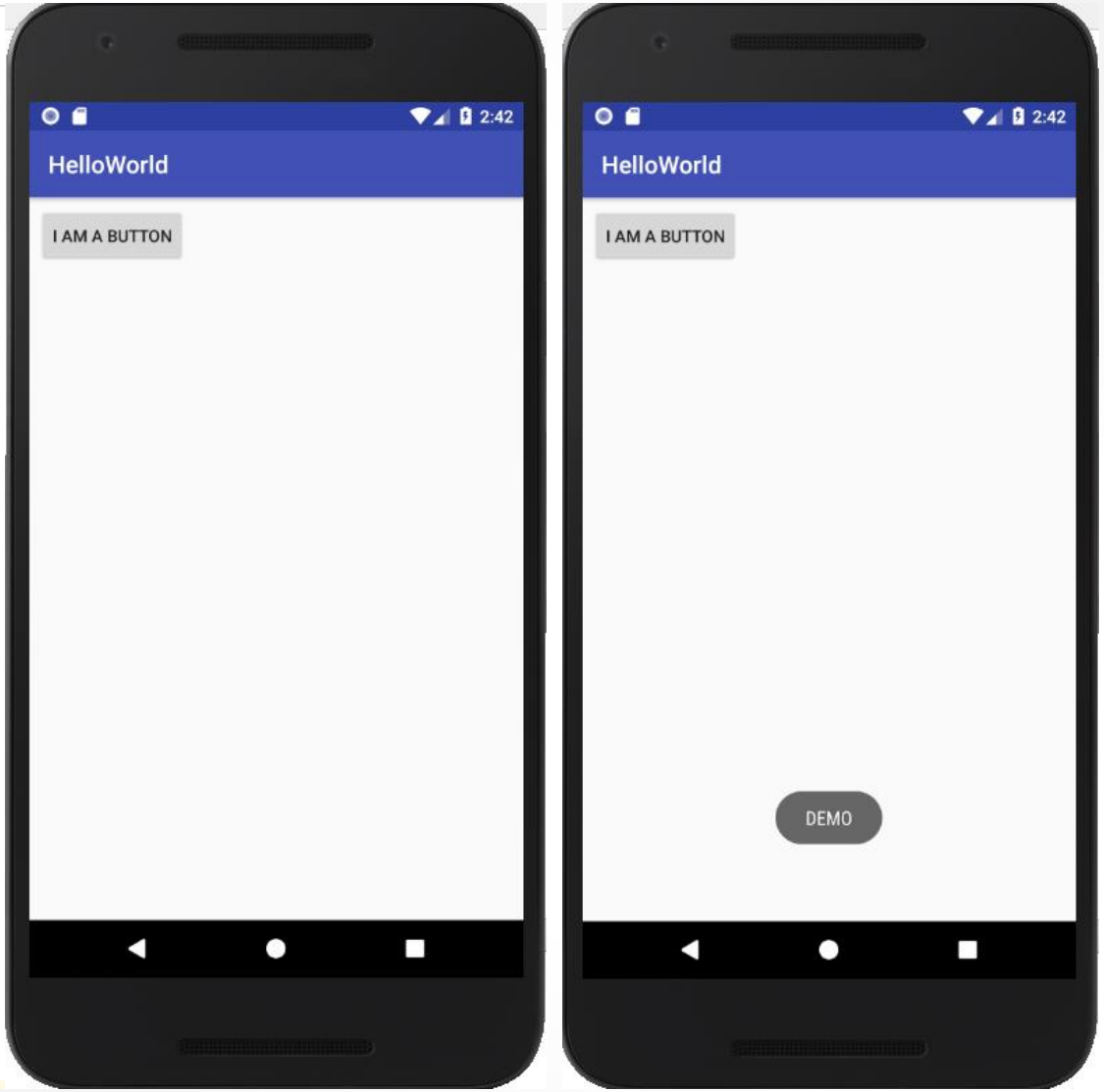
- 使用封装好的LocalBroadcastManager类：
- 使用方式上与全局广播几乎相同，只是注册/取消注册广播接收器和发送广播时将参数的context变成了LocalBroadcastManager的单一实例。

```java
package com.example.bishop.helloworld;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.util.Log;
import android.widget.Toast;

public class MyReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        // TODO: This method is called when the BroadcastReceiver is receiving
        // an Intent broadcast.
        Toast.makeText(context, text: "DEMO", Toast.LENGTH_LONG).show();
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.bishop.helloworld">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="HelloWorld"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

BROADCAST RECEIVER

```java
package com.example.bishop.helloworld;

import android.content.IntentFilter;
import android.support.v4.content.LocalBroadcastManager;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.content.Intent;

public class MainActivity extends AppCompatActivity {
     MyReceiver receiver=new MyReceiver();;
     LocalBroadcastManager manager;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    @Override
    protected void onResume(){
        manager.getInstance(this).registerReceiver(receiver,new IntentFilter( action: "com.example.DEMO"));
        super.onResume();
    }
    @Override
    protected void onPause(){
        manager.getInstance(this).unregisterReceiver(receiver);
        super.onPause();
    }

     public void onClick(View v){
        Intent intent=new Intent( action: "com.example.DEMO");
         manager.getInstance(this).sendBroadcast(intent);
     }
}
```



BROADCAST RECEIVER

- 对于不同注册方式的广播接收器回调OnReceive（Context context，Intent intent）中的context返回值是不一样的：
- 对于静态注册（全局+应用内广播），回调onReceive(context, intent)中的context返回值是：ReceiverRestrictedContext；
- 对于全局广播的动态注册，回调onReceive(context, intent)中的context返回值是：Activity Context；
- 对于应用内广播的动态注册（LocalBroadcastManager方式），回调onReceive(context, intent)中的context返回值是：Application Context；
- 对于应用内广播的动态注册（非LocalBroadcastManager方式），回调onReceive(context, intent)中的context返回值是：Activity Context。

THE END.