22

Android Services

Victor Matos
Cleveland State University

Notes are based on:
Android Developers
http://developer.android.com/index.html

Services

Android Services

A Service is an application component that runs in the background, not interacting with the user, for an **indefinite** period of time.

Note that services, like other application objects, run in the main thread of their hosting process. This means that, if your service is going to do any CPU intensive (such as MP3 playback) or blocking (such as networking) operations, it *should spawn its own thread in which to do that work.*

Each service class must have a corresponding <service> declaration in its package's AndroidManifest.xml.

Services can be started with Context.startService() and Context.bindService().

Services

Android Services

Multiple calls to **Context.startService()** do not nest (though they do result in multiple corresponding calls to the onStart() method of the Service class), so no matter how many times it is started a service will be stopped once **Context.stopService()** or **stopSelf()** is called.

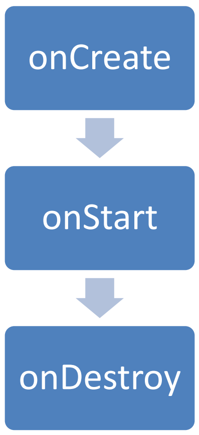
A service can be started and allowed to run until someone stops it or it stops itself. Only one **stopService()** call is needed to stop the service, no matter how many times **startService()** was called.

Services

Service Life Cycle

Like an activity, a service has lifecycle methods that you can implement to monitor changes in its state. But they are fewer than the activity methods — only three — and they are public, not protected:

1. **void onCreate ()**
2. **void onStart (Intent intent)**
3. **void onDestroy ()**



Services

Service Life Cycle

The entire lifetime of a service happens between the time onCreate() is called and the time onDestroy() returns.

Like an activity, a service does its initial setup in onCreate(), and releases all remaining resources in onDestroy().

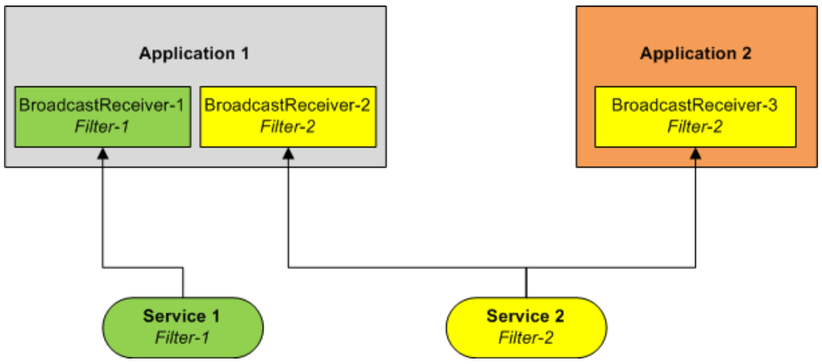
For example, a music playback service could create the thread where the music will be played in *onCreate*(), and then stop the thread in *onDestroy*().

Services

Broadcast Receiver Lifecycle

A Broadcast Receiver is an application class that listens for Intents that are broadcast, rather than being sent to a single target application/activity.

The system delivers a broadcast Intent to all interested broadcast receivers, which handle the Intent *sequentially*.



Services

Registering a Broadcast Receiver

- You can either *dynamically* register an instance of this class with **Context.registerReceiver()**
- or statically publish an implementation through the **<receiver>** tag in your AndroidManifest.xml.

Services

Broadcast Receiver Lifecycle

A broadcast receiver has a single callback method:

void onReceive (Context curContext, Intent broadcastMsg)

1. When a broadcast message arrives for the receiver, Android calls its onReceive() method and passes it the Intent object containing the message.
2. The broadcast receiver is considered to be active only while it is executing this method.
3. When onReceive() returns, it is inactive.

Services

Services, BroadcastReceivers and the AdroidManifest

The manifest of applications using Android Services must include:

1. A **<service>** entry for each service used in the application.

2. If the application defines a **BroadcastReceiver** as an independent class, it must include a **<receiver>** clause identifying the component. In addition an **<intent-filter>** entry is needed to declare the actual filter the service and the receiver use.

See example

Services

Services, BroadcastReceivers and the AdroidManifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="cis493.demos" android:versionCode="1" android:versionName="1.0.0">
    <uses-sdk android:minSdkVersion="4"></uses-sdk>

    <application android:icon="@drawable/icon" android:label="@string/app_name">

        <activity android:name=".MyServiceDriver2">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <service android:name="MyService2" />

        <receiver android:name="MyBroadcastReceiver">
            <intent-filter>
                <action android:name = "matos.action.GOSERVICE2" />
            </intent-filter>
        </receiver>

     </application>
</manifest>
```

Services

Types of Broadcasts

There are two major classes of broadcasts that can be received:

1. **Normal broadcasts** (sent with **Context.sendBroadcast**) are completely asynchronous. All receivers of the broadcast are run in an undefined order, often at the same time.
2. **Ordered broadcasts** (sent with **Context.sendOrderedBroadcast**) are delivered to one receiver at a time. As each receiver executes in turn, it can propagate a result to the next receiver, or it can completely abort the broadcast so that it won't be passed to other receivers. The order receivers run in can be controlled with the **android:priority** attribute of the matching intent-filter; receivers with the same priority will be run in an arbitrary order.

Services

Useful Methods – The Driver

Assume main activity *MyService3Driver* wants to interact with a service called *MyService3*. The main activity is responsible for the following tasks:

1. Start the service called *MyService3*.

```
Intent intentMyService = new Intent(this, MyService3.class);
Service myService = startService(intentMyService);
```

2. Define corresponding receiver's filter and register local receiver

```
IntentFilter mainFilter = new IntentFilter("matos.action.GOSERVICE3");
BroadcastReceiver receiver = new MyMainLocalReceiver();
registerReceiver(receiver, mainFilter);
```

3. Implement local receiver and override its main method

```
public void onReceive(Context localContext, Intent callerIntent)
```

Services

Useful Methods – The Service

Assume main activity *MyService3Driver* wants to interact with a service called *MyService3*. The Service uses its *onStart* method to do the following:

1. Create an Intent with the appropriate broadcast filter (any number of receivers could match it).

```
Intent myFilteredResponse = new Intent("matos.action.GOSERVICE3");
```

2. Prepare the extra data ('myServiceData') to be sent with the intent to the receiver(s)

```
Object msg = some user data goes here;
myFilteredResponse.putExtra("myServiceData", msg);
```

3. Release the intent to all receivers matching the filter

```
sendBroadcast(myFilteredResponse);
```

Services

Useful Methods – The Driver (again)

Assume main activity *MyService3Driver* wants to interact with a service called *MyService3*. The main activity is responsible for cleanly terminating the service. Do the following

1. Assume `intentMyService` is the original Intent used to start the service. Calling the termination of the service is accomplished by the method

```
stopService(new Intent(intentMyService) );
```

2. Use the service's onDestroy method to assure that all of its running threads are terminated and the receiver is unregistered.

```
unregisterReceiver(receiver);
```

Services

Example 1. A very Simple Service

The main application starts a service. The service prints lines on the DDMS **LogCat** until the main activity stops the service. No IPC occurs in the example.

```
// a simple service is started & stopped
package cis493.demos;
import android.app.Activity;
import android.content.ComponentName;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.*;

public class ServiceDriver1 extends Activity {
   TextView txtMsg;
   Button btnStopService;
   ComponentName service;
   Intent intentMyService;
```

Services

Example 1. *cont.*

```
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      txtMsg = (TextView) findViewById(R.id.txtMsg);

      intentMyService = new Intent(this, MyService1.class);
       service = startService(intentMyService);

      btnStopService = (Button) findViewById(R.id.btnStopService);
      btnStopService.setOnClickListener(new OnClickListener() {
         public void onClick(View v) {
         try {
                  stopService((intentMyService)  );
                  txtMsg.setText("After stoping Service: \n" +  service.getClassName());
         } catch (Exception e) {
                  Toast.makeText(getApplicationContext(), e.getMessage(), 1).show();
         }
      }
    } );
  }
}
```

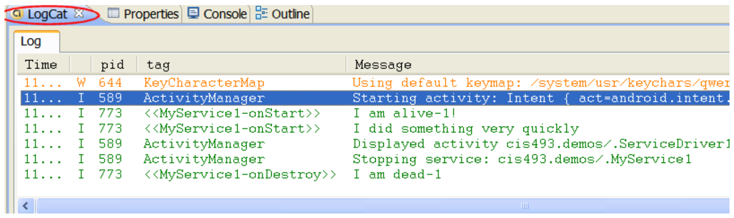
Services

Example 1. *cont.*

```java
//non CPU intensive service running the main task in its main thread
package cis493.demos;
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.util.Log;

public class MyService1 extends Service {
     @Override
     public IBinder onBind(Intent arg0) {
        return null;
     }
     @Override
     public void onCreate() {
        super.onCreate();
        Log.i ("<<MyService1-onStart>>", "I am alive-1!");
     }
     @Override
     public void onStart(Intent intent, int startId) {
        super.onStart(intent, startId);
        Log.i ("<<MyService1-onStart>>", "I did something very quickly");
     }
     @Override
     public void onDestroy() {
        super.onDestroy();
        Log.i ("<<MyService1-onDestroy>>", "I am dead-1");
     }
}//MyService1
```

Services

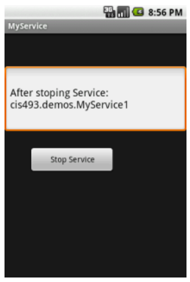
Example 1. *cont.*



```
Time     pid   tag                      Message
11...  W  644   KeyCharacterMap          Using default keymap: /system/usr/keychars/qwer
11...  I  589   ActivityManager          Starting activity: Intent { act=android.intent.
11...  I  773   <<MyService1-onStart>>   I am alive-1!
11...  I  773   <<MyService1-onStart>>   I did something very quickly
11...  I  589   ActivityManager          Displayed activity cis493.demos/.ServiceDriver1
11...  I  589   ActivityManager          Stopping service: cis493.demos/.MyService1
11...  I  773   <<MyService1-onDestroy>> I am dead-1
```

According to the Log

1. Main Activity is started (no displayed yet)
2. Service is started (onCreate, onStart)
3. Main Activity UI is displayed
4. User stops Service



After stoping Service:
cis493.demos.MyService1

18

Services

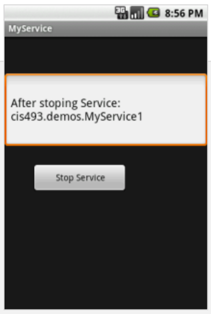
Example 1. *cont.* *Manifest*

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
     package="cis493.demos"
     android:versionCode="1"
     android:versionName="1.0">
   <application android:icon="@drawable/icon"
           android:label="@string/app_name">
      <activity android:name=".ServiceDriver1"
            android:label="@string/app_name">
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
   <service android:name=".MyService1"> </service>
   </application>
   <uses-sdk android:minSdkVersion="4" />

</manifest>
```

Services

Example 1. *cont.* *Layout*

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
android:id="@+id/widget32"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
xmlns:android="http://schemas.android.com/apk/res/android"
>
<EditText
android:id="@+id/txtMsg"
android:layout_width="fill_parent"
android:layout_height="120px"
android:textSize="18sp"
android:layout_x="0px"
android:layout_y="57px"
>
</EditText>
<Button
android:id="@+id/btnStopService"
android:layout_width="151px"
android:layout_height="wrap_content"
android:text=" Stop Service"
android:layout_x="43px"
android:layout_y="200px"
>
</Button>
</AbsoluteLayout>
```



After stoping Service:
cis493.demos.MyService1

Services

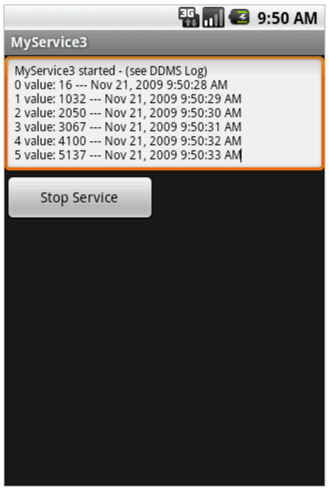
Example3. Realistic Activity-Service Interaction

1. The main activity starts the *service* and registers a *receiver*.
2. The service is slow, therefore it runs in a parallel thread its time consuming task.
3. When done with a computing cycle, the service adds a message to an intent.
4. The *intent* is broadcasted using the filter: matos.action.GOSERVICE3.
5. A *BroadcastReceiver* (defined inside the main Activity) uses the previous filter and catches the message (displays the contents on the main UI).
6. At some point the main activity stops the service and finishes executing.

Services

Example 2. Layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
android:id="@+id/widget32"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical"
xmlns:android="http://schemas.android.com/apk/res/android"
>
<EditText
android:id="@+id/txtMsg"
android:layout_width="fill_parent"
android:layout_height="120px"
android:textSize="12sp"
>
</EditText>
<Button
android:id="@+id/btnStopService"
android:layout_width="151px"
android:layout_height="wrap_content"
android:text="Stop Service"

>
</Button>
</LinearLayout>
```



```
MyService3 started - (see DDMS Log)
0 value: 16 --- Nov 21, 2009 9:50:28 AM
1 value: 1032 --- Nov 21, 2009 9:50:29 AM
2 value: 2050 --- Nov 21, 2009 9:50:30 AM
3 value: 3067 --- Nov 21, 2009 9:50:31 AM
4 value: 4100 --- Nov 21, 2009 9:50:32 AM
5 value: 5137 --- Nov 21, 2009 9:50:33 AM
```

Services

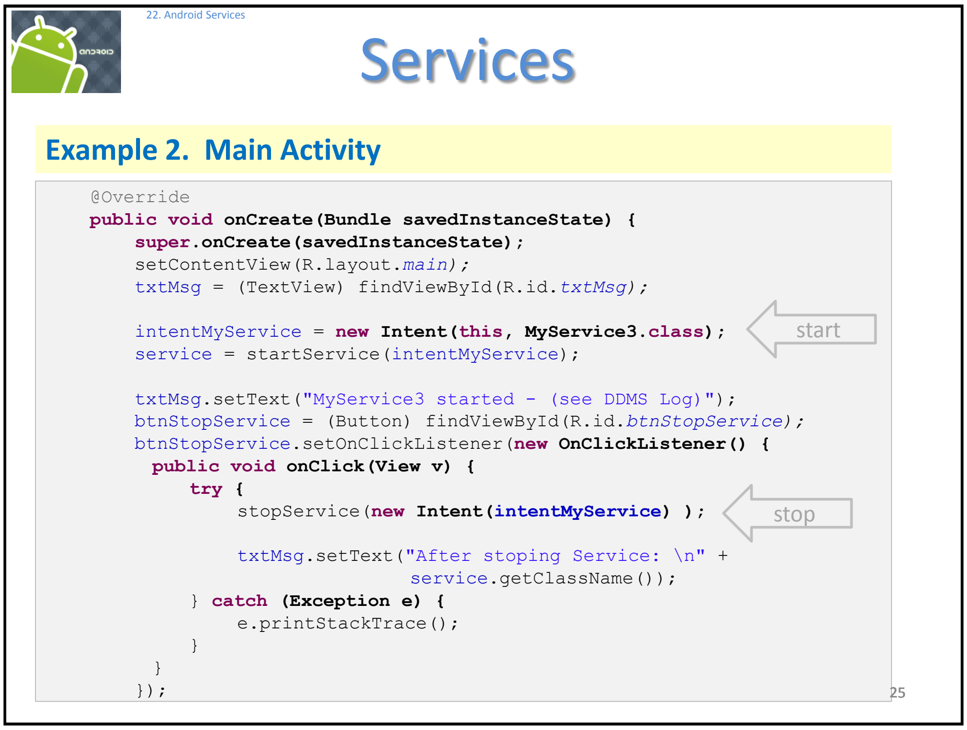
Example 2. Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="cis493.demos"
    android:versionCode="1"
    android:versionName="1.0.0">

  <uses-sdk android:minSdkVersion="4"></uses-sdk>

  <application android:icon="@drawable/icon" android:label="@string/app_name">

    <activity android:name=".MyServiceDriver3"
          android:label="@string/app_name">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>

    <service android:name="MyService3">
    </service>

  </application>

</manifest>
```

Services

Example 2. Main Activity

```
// Application logic and its BroadcastReceiver in the same class
package cis493.demos;
import java.util.Date;
import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.os.SystemClock;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.*;

public class MyServiceDriver3 extends Activity {
    TextView txtMsg;
    Button btnStopService;
    ComponentName service;
    Intent intentMyService;
    BroadcastReceiver receiver;
```



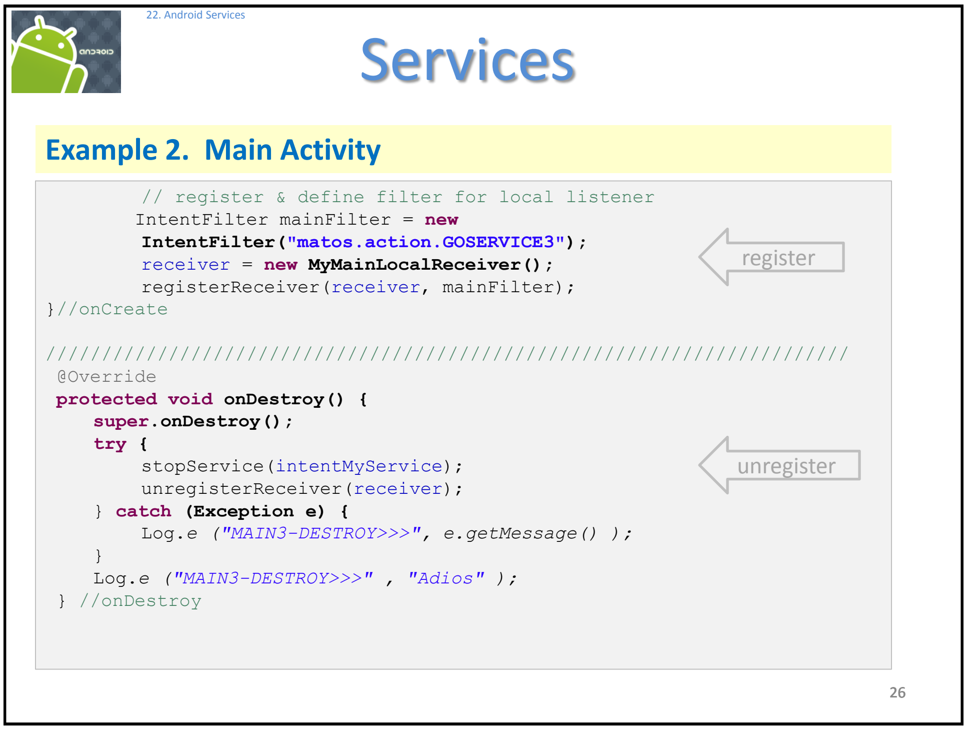
Services

Example 2. Main Activity

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        txtMsg = (TextView) findViewById(R.id.txtMsg);

        intentMyService = new Intent(this, MyService3.class);
        service = startService(intentMyService);

        txtMsg.setText("MyService3 started - (see DDMS Log)");
        btnStopService = (Button) findViewById(R.id.btnStopService);
        btnStopService.setOnClickListener(new OnClickListener() {
          public void onClick(View v) {
              try {
                  stopService(new Intent(intentMyService) );

                  txtMsg.setText("After stoping Service: \n" +
                                 service.getClassName());
              } catch (Exception e) {
                  e.printStackTrace();
              }
          }
        });
```

start

stop



Services

Example 2. Main Activity

```
        // register & define filter for local listener
        IntentFilter mainFilter = new
         IntentFilter("matos.action.GOSERVICE3");
         receiver = new MyMainLocalReceiver();
         registerReceiver(receiver, mainFilter);
}//onCreate

//////////////////////////////////////////////////////////////////////////
 @Override
 protected void onDestroy() {
     super.onDestroy();
     try {
         stopService(intentMyService);
         unregisterReceiver(receiver);
     } catch (Exception e) {
         Log.e ("MAIN3-DESTROY>>>", e.getMessage() );
     }
     Log.e ("MAIN3-DESTROY>>>" , "Adios" );
 } //onDestroy
```

register

unregister

Services

Example 2. Main Activity

```
/////////////////////////////////////////////////////////////////////////
// local (embedded) RECEIVER

public class MyMainLocalReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context localContext, Intent callerIntent) {

        String serviceData = callerIntent.getStringExtra("serviceData");

        Log.e ("MAIN>>>", serviceData + " -receiving data "
               + SystemClock.elapsedRealtime() );

        String now = "\n" + serviceData + " --- "
                   + new Date().toLocaleString();
        txtMsg.append(now);
        }
    }//MyMainLocalReceiver

}//MyServiceDriver4
```

Get data

Services

Example 2. The Service

```
// Service3 uses a thread to run slow operation
package cis493.demos;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.util.Log;

public class MyService3 extends Service {
    boolean isRunning = true;
@Override
public IBinder onBind(Intent arg0) {
return null;
}

@Override
public void onCreate() {
super.onCreate();
}
```

Services

Example 2. The Service

```
@Override
public void onStart(Intent intent, int startId) {
     super.onStart(intent, startId);
     Log.e ("<<MyService3-onStart>>", "I am alive-3!");
     // we place the slow work of the service in its own thread
     // so the caller is not hung up waiting for us
     Thread triggerService = new Thread ( new Runnable(){
               long startingTime = System.currentTimeMillis();
               long tics = 0;
               public void run() {
               for(int i=0; (i< 120) & isRunning; i++) { //at most 10 minutes
               try {
                    //fake that you are very busy here
                    tics = System.currentTimeMillis() - startingTime;
                    Intent myFilteredResponse = new Intent("matos.action.GOSERVICE3");
                    String msg = i + " value: " + tics;
                    myFilteredResponse.putExtra("serviceData", msg);
                    sendBroadcast(myFilteredResponse);
                    Thread.sleep(1000); //five seconds
               } catch (Exception e) { e.printStackTrace(); }
               }//for
              }//run
     });
     triggerService.start();
}//onStart
```

Set filter

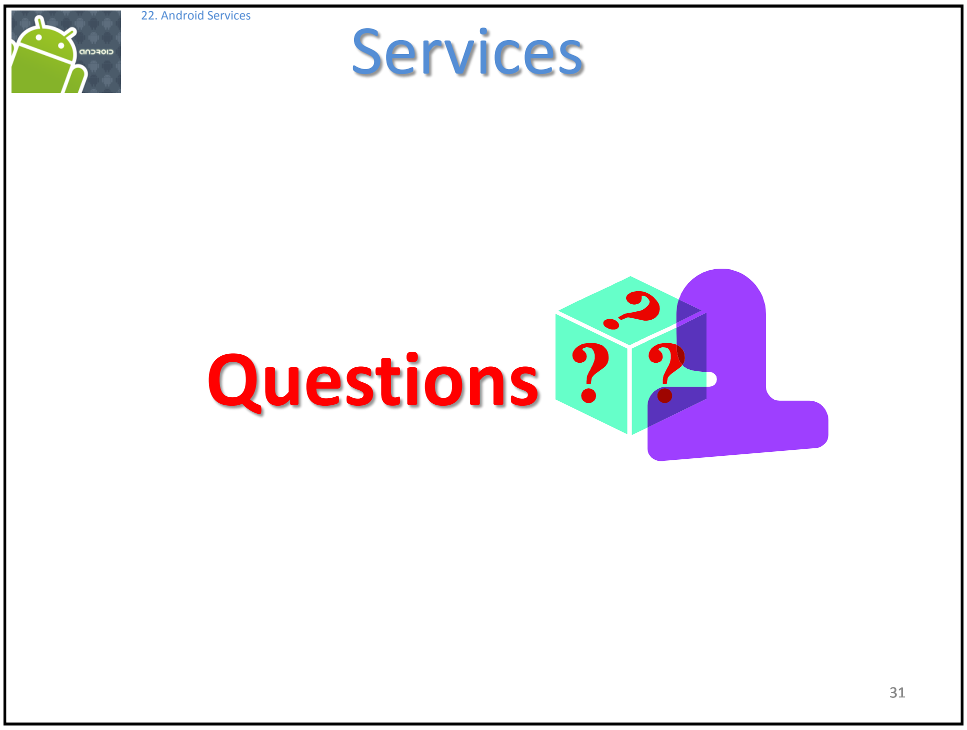
broadcasting

Services

Example 2. The Service

```
  @Override
  public void onDestroy() {
     super.onDestroy();
     Log.e ("<<MyService3-onDestroy>>", "I am dead-3");

     isRunning = false;

  }//onDestroy

}//MyService3
```

Stop thread



Services

Questions