MOBILE DEVELOPMENT

SERVICES

CONTENTS

ANDROID SERVICE

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

Services, like other application objects (activitties, broadcast listeners…), 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: startService() and bindService().
- Each startService call invokes the onStart() method of the service class, however the service is started only with the first call.
- Only one stopService() call is needed to stop the service, no matter how many times startService() was called.

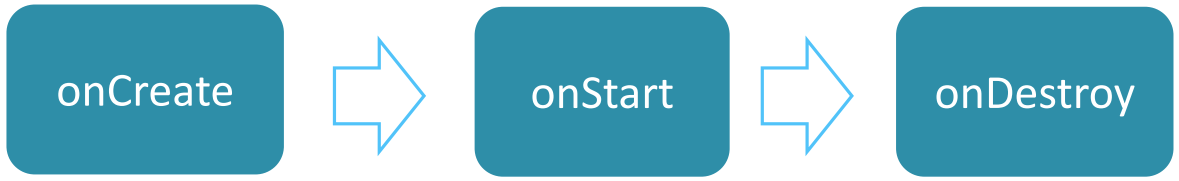
Taken from: http://developer.android.com/guide/components/services.html

ANDROID SERVICE

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()



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().

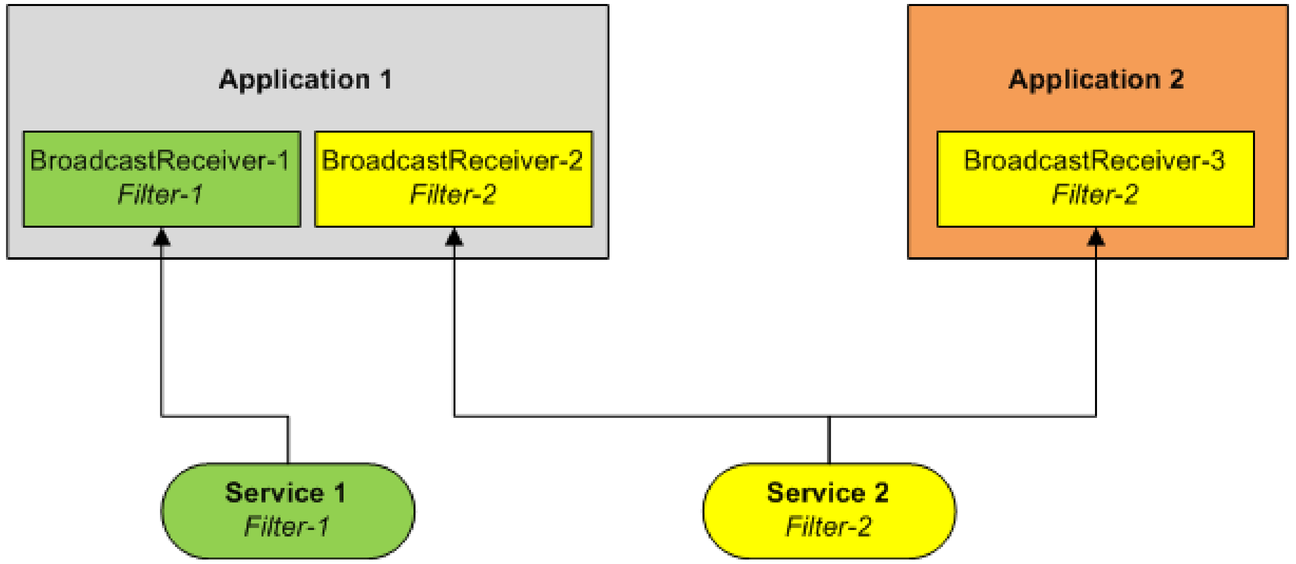
For example, a music playback service could create the thread where the music will be played in onCreate(), and then stop the thread in onDestroy().

BROADCAST RECEIVER

Broadcast Receiver Lifecycle

A Broadcast Receiver is an application class that listens for global Intents that are broadcasted to anyone who bothers to listen, 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.



BROADCAST RECEIVER

Registering a Broadcast Receiver

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

Broadcast Receiver Lifecycle

```
void onReceive (Context context, Intent broadcastMsg)
```

A broadcast receiver has a single callback method:

- 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 its onReceive() method.
- 3. When onReceive() returns, it is inactive.

BROADCAST RECEIVER

Services, BroadcastReceivers and the AdroidManifest

The manifest of applications using Android Services must include:

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

```
<?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="10"></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>
```

BROADCAST RECEIVER

Types of Broadcasts

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

- 1. Normal broadcasts (sent with 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 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 (abortBroadcast())so that it won't be passed to other receivers.
  - Ordering receivers for execution 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.

EXAMPLES (Main steps – main activity)

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);
ComponentName service = 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)
```

EXAMPLES (Main steps – the service)

The Service uses its onStart method to do the following:

1. Create an Intent with 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);
```

EXAMPLES (Main steps – main activity)

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);
```

EXAMPLES 1

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

```
public class TestMyService1 extends Activity implements OnClickListener {
 TextView txtMsg; ComponentName service;Intent intentMyService1;
 @Override
 public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
  txtMsg = (TextView) findViewById(R.id.txtMsg);
  findViewById(R.id.btnStopService).setOnClickListener(this);
  intentMyService1 = new Intent(this, MyService1.class);
  service = startService(intentMyService1);
  txtMsg.setText("MyService1 started\n (see LogCat)");
 }
 @Override
 public void onClick(View v) {
  // assume: v.getid == R.id.btnStopService
  try {
   stopService(intentMyService1);
   txtMsg.setText("After stopping Service: \n" + service.getClassName());
  }
  catch (Exception e) { Toast.makeText(this, e.getMessage(), 1).show(); }
 }//onClick
}
```

EXAMPLES 1

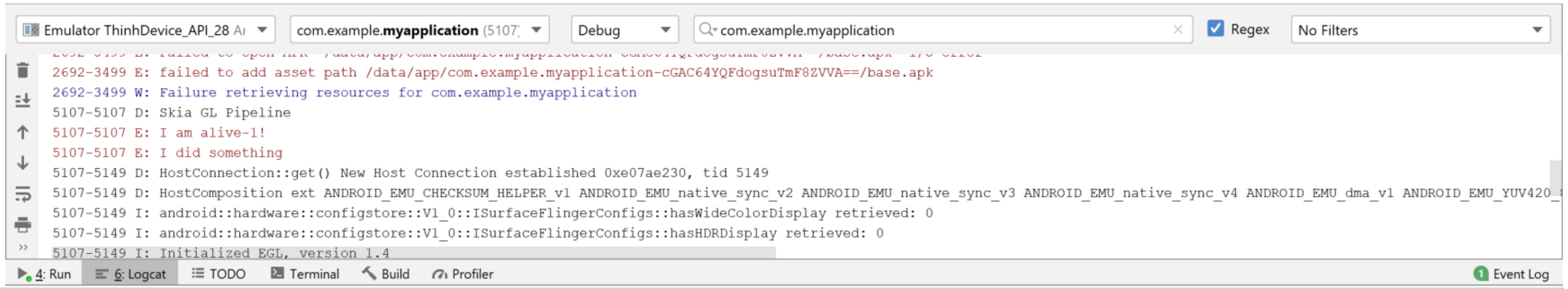
The MyService1

```
//non CPU intensive service running the main task in its main thread
package cis.matos;
import . . .
public class MyService1 extends Service {
  @Override
  public IBinder onBind(Intent arg0) { return null; }
  @Override
  public void onCreate() { super.onCreate(); }
  @Override
  public void onStart(Intent intent, int startId) {
    Log.e (“<<MyService1-onStart>>”, “I am alive-1!”);
    Log.e (“<<MyService1-onStart>>”, “I did something”);
  }
  @Override
  public void onDestroy() { Log.e (“<<MyService1-onDestroy>>”, “I am dead-1”); }
} //MyService1
```

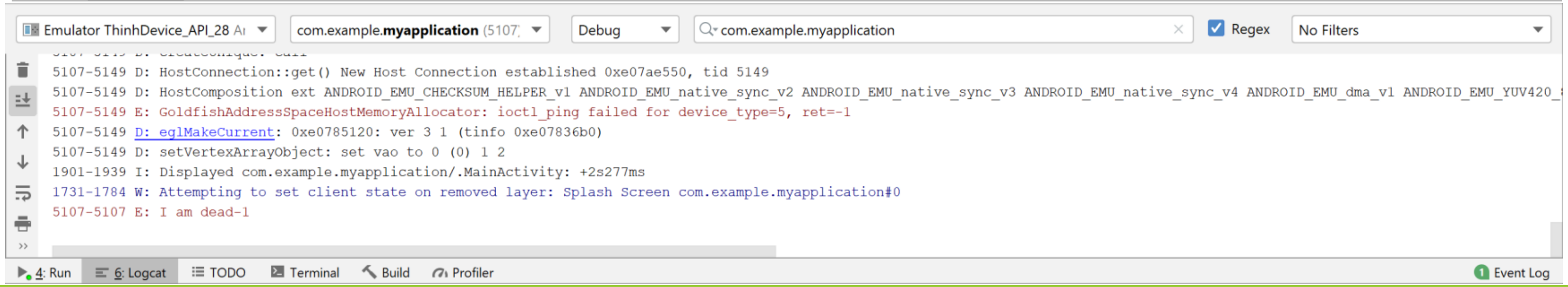
EXAMPLES 1

According to the Log

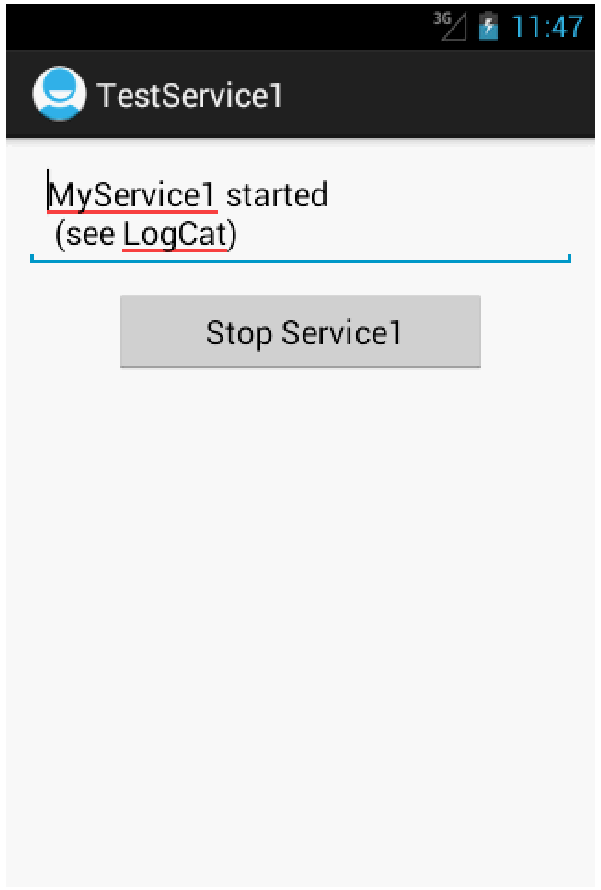
- 1. Main Activity is started
- 2. Service is started (onCreate, onStart)
- 3. Main Activity UI is displayed
- 4. User stops Service


Emulator ThinhDevice_API_28
com.example.myapplication (5107
Debug
com.example.myapplication
Regex
No Filters
4: Run
6: Logcat
TODO
Terminal
Build
Profiler
Event Log

```
2692-3499 E: failed to add asset path /data/app/com.example.myapplication-cGAC64YQFdogsuTmF8ZVVA==/base.apk
2692-3499 W: Failure retrieving resources for com.example.myapplication
5107-5107 D: Skia GL Pipeline
5107-5107 E: I am alive-1!
5107-5107 E: I did something
5107-5149 D: HostConnection::get() New Host Connection established 0xe07ae230, tid 5149
5107-5149 D: HostComposition ext ANDROID_EMU_CHECKSUM_HELPER_v1 ANDROID_EMU_native_sync_v2 ANDROID_EMU_native_sync_v3 ANDROID_EMU_native_sync_v4 ANDROID_EMU_dma_v1 ANDROID_EMU_YUV420_
5107-5149 I: android::hardware::configstore::V1_0::ISurfaceFlingerConfigs::hasWideColorDisplay retrieved: 0
5107-5149 I: android::hardware::configstore::V1_0::ISurfaceFlingerConfigs::hasHDRDisplay retrieved: 0
5107-5149 I: Initialized EGL, version 1.4
```


Emulator ThinhDevice_API_28
com.example.myapplication (5107
Debug
com.example.myapplication
Regex
No Filters
4: Run
6: Logcat
TODO
Terminal
Build
Profiler
Event Log

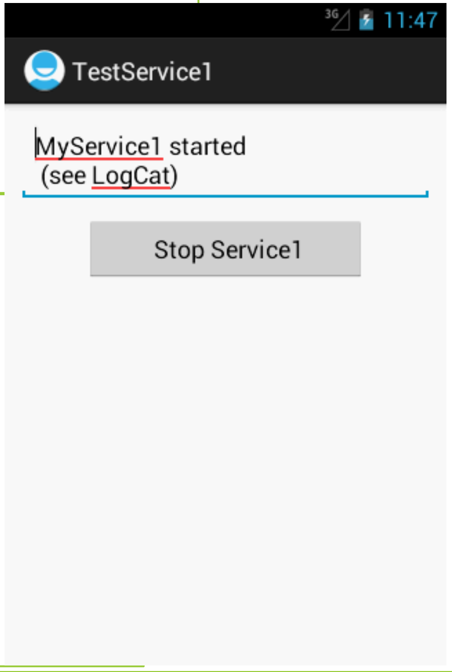
```
5107-5149 D: HostConnection::get() New Host Connection established 0xe07ae550, tid 5149
5107-5149 D: HostComposition ext ANDROID_EMU_CHECKSUM_HELPER_v1 ANDROID_EMU_native_sync_v2 ANDROID_EMU_native_sync_v3 ANDROID_EMU_native_sync_v4 ANDROID_EMU_dma_v1 ANDROID_EMU_YUV420_
5107-5149 E: GoldfishAddressSpaceHostMemoryAllocator: ioctl_ping failed for device_type=5, ret=-1
5107-5149 D: eglMakeCurrent: 0xe0785120: ver 3 1 (tinfo 0xe07836b0)
5107-5149 D: setVertexArrayObject: set vao to 0 (0) 1 2
1901-1939 I: Displayed com.example.myapplication/.MainActivity: +2s277ms
1731-1784 W: Attempting to set client state on removed layer: Splash Screen com.example.myapplication#0
5107-5107 E: I am dead-1
```



EXAMPLES 1 (MANIFEST & LAYOUT)

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="csu.matos" android:versionCode="1" android:versionName="1.0">
 <uses-sdk android:minSdkVersion="8" android:targetSdkVersion="15" />
 <application android:icon="@drawable/ic_launcher" android:label="@string/app_name" android:theme="@style/AppTheme">
  <activity android:name=".TestMyService1" android:label="@string/title_activity_test_service1">
   <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
   </intent-filter>
  </activity>
  <service android:name="MyService1" />
 </application>
</manifest>
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="match_parent" android:layout_height="match_parent"
              android:orientation="vertical" >
 <EditText    android:id="@+id/txtMsg" android:layout_width="match_parent"
              android:layout_height="wrap_content" android:inputType="none" android:layout_margin="10dp" />
 <Button      android:id="@+id/btnStopService" android:layout_width="204dp"
              android:layout_height="wrap_content" android:layout_gravity="center"
              android:text="Stop Service1"/>
</LinearLayout>
```



MyService1 started
(see LogCat)

EXAMPLES 2 (A more interesting 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.

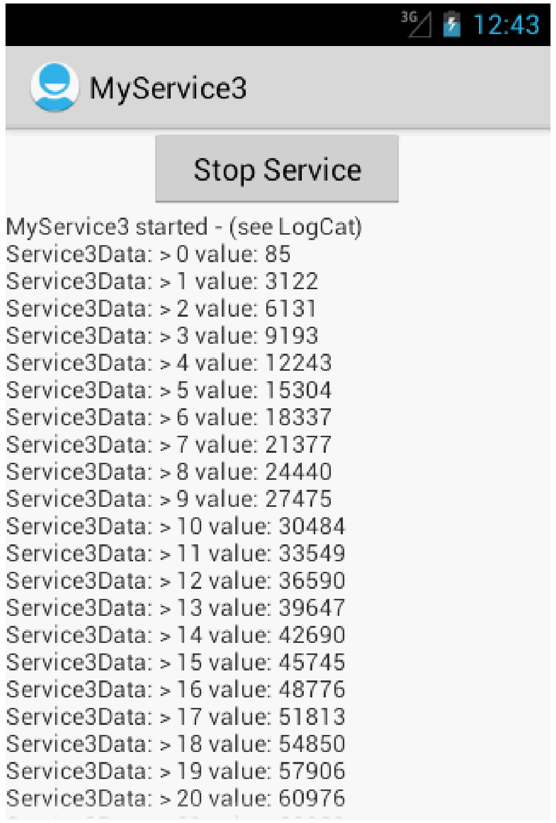
4. The intent is broadcasted using the filter: matos.action.GOSERVICE3.

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.

EXAMPLES 2
(A more interesting activity-service interaction)

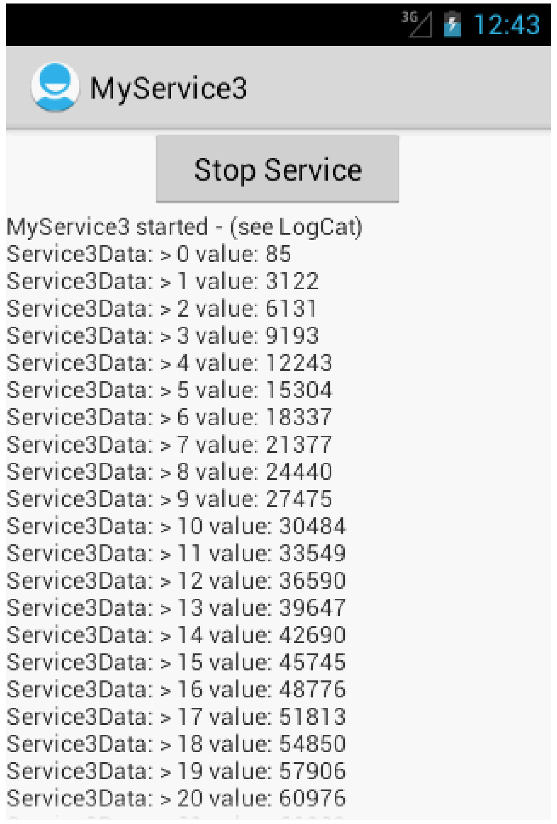
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical">
 <Button      android:id="@+id/btnStopService"
              android:layout_width="151dip"
              android:layout_height="wrap_content"
              android:layout_gravity="center"
              android:text="Stop Service" />
 <ScrollView  android:layout_width="match_parent" android:layout_height="wrap_content" >
  <TextView  android:id="@+id/txtMsg"
             android:layout_width="match_parent"
             android:layout_height="wrap_content"
             android:inputType="none" />
 </ScrollView>
</LinearLayout>
```



```
MyService3 started - (see LogCat)
Service3Data: > 0 value: 85
Service3Data: > 1 value: 3122
Service3Data: > 2 value: 6131
Service3Data: > 3 value: 9193
Service3Data: > 4 value: 12243
Service3Data: > 5 value: 15304
Service3Data: > 6 value: 18337
Service3Data: > 7 value: 21377
Service3Data: > 8 value: 24440
Service3Data: > 9 value: 27475
Service3Data: > 10 value: 30484
Service3Data: > 11 value: 33549
Service3Data: > 12 value: 36590
Service3Data: > 13 value: 39647
Service3Data: > 14 value: 42690
Service3Data: > 15 value: 45745
Service3Data: > 16 value: 48776
Service3Data: > 17 value: 51813
Service3Data: > 18 value: 54850
Service3Data: > 19 value: 57906
Service3Data: > 20 value: 60976
```

EXAMPLES 2
(A more interesting activity-service interaction)

```
<?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="10" />
 <application    android:icon="@drawable/ic_launcher"
                 android:label="@string/app_name"
                 android:theme="@android:style/Theme.Holo.Light">
  <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"/>
 </application>
</manifest>
```



```
MyService3 started - (see LogCat)
Service3Data: > 0 value: 85
Service3Data: > 1 value: 3122
Service3Data: > 2 value: 6131
Service3Data: > 3 value: 9193
Service3Data: > 4 value: 12243
Service3Data: > 5 value: 15304
Service3Data: > 6 value: 18337
Service3Data: > 7 value: 21377
Service3Data: > 8 value: 24440
Service3Data: > 9 value: 27475
Service3Data: > 10 value: 30484
Service3Data: > 11 value: 33549
Service3Data: > 12 value: 36590
Service3Data: > 13 value: 39647
Service3Data: > 14 value: 42690
Service3Data: > 15 value: 45745
Service3Data: > 16 value: 48776
Service3Data: > 17 value: 51813
Service3Data: > 18 value: 54850
Service3Data: > 19 value: 57906
Service3Data: > 20 value: 60976
```

EXAMPLES 2
(A more interesting activity-service interaction)

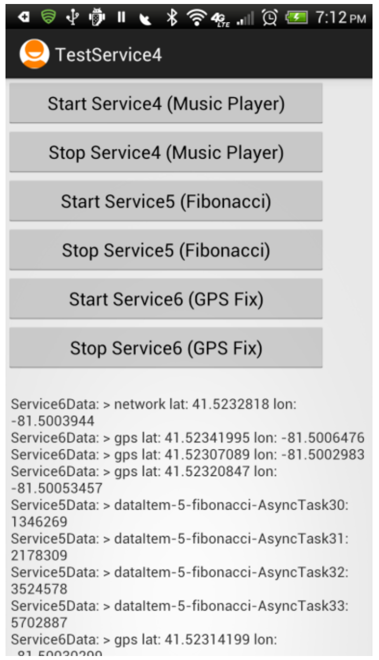
```
public class MyServiceDriver3 extends Activity implements OnClickListener {
  TextView txtMsg; ComponentName service;
  Intent intentMyService3;
  BroadcastReceiver receiver;
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState); setContentView(R.layout.main);
    txtMsg = (TextView) findViewById(R.id.txtMsg);
    intentMyService3 = new Intent(this, MyService3.class);
    service = startService(intentMyService3);
    txtMsg.setText("MyService3 started - (see LogCat)");
    findViewById(R.id.btnStopService).setOnClickListener(this);
    // register & define filter for local listener
    IntentFilter mainFilter = new IntentFilter("matos.action.GOSERVICE3");
    receiver = new MyMainLocalReceiver();
    registerReceiver(receiver, mainFilter);
  }//onCreate
  public void onClick(View v) { // assume: v.getId() == R.id.btnStopService
    try {
      stopService(intentMyService3);
      txtMsg.setText("After stoping Service: \n" + service.getClassName() );
    }
    catch (Exception e) { e.printStackTrace(); }
  }
```

```
  @Override
  protected void onDestroy() {
    super.onDestroy();
    try {
      stopService(intentMyService3);
      unregisterReceiver(receiver);
    }
    catch (Exception e) { Log.e ("MAIN3-DESTROY>>>", e.getMessage() ); }
    Log.e ("MAIN3-DESTROY>>>" , "Adios" );
  }
  public class MyMainLocalReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context localContext, Intent callerIntent) {
      String serviceData = callerIntent.getStringExtra("service3Data");
      Log.e ("MAIN>>>", "Data received from Service3: " + serviceData);
      String now = "\nService3Data: > " + serviceData;
      txtMsg.append(now);
    }
  }//MyMainLocalReceiver
}//MyServiceDriver3
```

EXAMPLES 2
(A more interesting activity-service interaction)

```
public class MyService3 extends Service {
 boolean isRunning = true;
 @Override public IBinder onBind(Intent arg0) { return null; }
 @Override public void onCreate() { super.onCreate(); }
 @Override
 public void onStart(Intent intent, int startId) {
  Log.e (“<<MyService3-onStart>>”, “I am alive-3!”);
  Thread serviceThread = new Thread ( new Runnable(){
   public void run() {
    for(int i=0; (i< 120) & isRunning; i++) {
     try { //fake that you are very busy here
      Thread.sleep(1000);
      Intent intentDataForMyClient = new Intent(“matos.action.GOSERVICE3”);
      String msg = “data-item-” + i;
      intentDataForMyClient.putExtra(“service3Data”, msg);
      sendBroadcast(intentDataForMyClient);
     }
     catch (Exception e) { e.printStackTrace(); }
    }//for
   }//run
  });
  serviceThread.start();
 }//onStart
```

```
 @Override
 public void onDestroy() {
  super.onDestroy();
  Log.e (“<<MyService3-onDestroy>>”, “I am Dead-3”);
  isRunning = false;
 }//onDestroy
}//MyService3
```


TestService4

Start Service4 (Music Player)

Stop Service4 (Music Player)

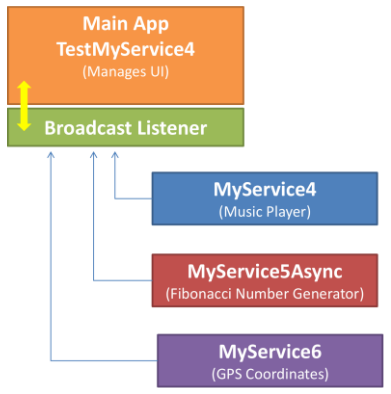
Start Service5 (Fibonacci)

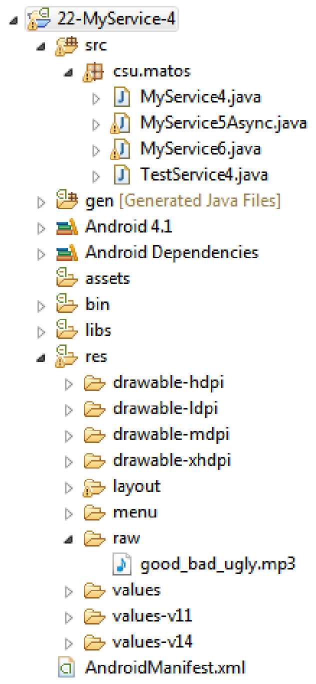
Stop Service5 (Fibonacci)

Start Service6 (GPS Fix)

Stop Service6 (GPS Fix)

```
Service6Data: > network lat: 41.5232818 lon:
-81.5003944
Service6Data: > gps lat: 41.52341995 lon: -81.5006476
Service6Data: > gps lat: 41.52307089 lon: -81.5002983
Service6Data: > gps lat: 41.52320847 lon:
-81.50053457
Service5Data: > dataItem-5-fibonacci-AsyncTask30:
1346269
Service5Data: > dataItem-5-fibonacci-AsyncTask31:
2178309
Service5Data: > dataItem-5-fibonacci-AsyncTask32:
3524578
Service5Data: > dataItem-5-fibonacci-AsyncTask33:
5702887
Service6Data: > gps lat: 41.52314199 lon:
```




```
22-MyService-4
  src
    csu.matos
      MyService4.java
      MyService5Async.java
      MyService6.java
      TestService4.java
  gen [Generated Java Files]
  Android 4.1
  Android Dependencies
  assets
  bin
  libs
  res
    drawable-hdpi
    drawable-ldpi
    drawable-mdpi
    drawable-xhdpi
    layout
    menu
    raw
      good_bad_ugly.mp3
    values
    values-v11
    values-v14
  AndroidManifest.xml
```

EXAMPLES 3 (An app connected to multiple services)

In this application the Main Activity starts three services:

1. MyService4: A music player whose input is an mp3 resource file stored in res/raw.

2. MyService5Async: A service producing Fibonacci numbers in the 20-50 range. The task of number generation is implemented inside an AsyncTask. The efficiency of this Fibonacci implementation is O(2^n) [intentionally slow!]

3. MyService6: The service returns GPS coordinates. Two methods are used to obtain the current location (a) a quick Network-provider based reading (coarse location) , and (b) a more precise but slower Satellite reading (fine location).

The Main Application defines and registers a BroadcastReceiver capable of attending messages matching any of the three filters used by the broadcasting services above. Received results are displayed on the user's screen.

EXAMPLES 3 (MainActivity: TestMyService4.java)

```
public class TestService4 extends Activity implements OnClickListener {
  TextView txtMsg;
  Intent intentCallService4, intentCallService5, intentCallService6;
  BroadcastReceiver receiver;
  @Override
  public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.main);
   txtMsg = (TextView) findViewById(R.id.txtMsg);
   findViewById(R.id.btnStart4).setOnClickListener(this);
   findViewById(R.id.btnStop4).setOnClickListener(this);
   findViewById(R.id.btnStart5).setOnClickListener(this);
   findViewById(R.id.btnStop5).setOnClickListener(this);
   findViewById(R.id.btnStart6).setOnClickListener(this);
   findViewById(R.id.btnStop6).setOnClickListener(this);
   Log.e("MAIN", "Main started");
   // get ready to invoke execution of background services
   intentCallService4 = new Intent(this, MyService4.class);
   intentCallService5 = new Intent(this, MyService5Async.class);
   intentCallService6 = new Intent(this, MyService6.class);
   // register local listener & define triggering filter
   IntentFilter filter5 = new IntentFilter("matos.action.GOSERVICE5");
   IntentFilter filter6 = new IntentFilter("matos.action.GPSFIX");
   receiver = new MyEmbeddedBroadcastReceiver();
   registerReceiver(receiver, filter5);
   registerReceiver(receiver, filter6);
  }// onCreate
```

```
@Override
 public void onClick(View v) {
  if (v.getId() == R.id.btnStart4) {
    Log.e("MAIN", "onClick: starting service4");
    startService(intentCallService4);
  }
  else if (v.getId() == R.id.btnStop4) {
    Log.e("MAIN", "onClick: stopping service4");
    stopService(intentCallService4);
  }
  else if (v.getId() == R.id.btnStart5) {
    Log.e("MAIN", "onClick: starting service5");
    startService(intentCallService5);
  }
  else if (v.getId() == R.id.btnStop5) {
    Log.e("MAIN", "onClick: stopping service5");
    stopService(intentCallService5);
  }
  else if (v.getId() == R.id.btnStart6) {
    Log.e("MAIN", "onClick: starting service6");
    startService(intentCallService6);
  }
  else if (v.getId() == R.id.btnStop6) {
    Log.e("MAIN", "onClick: stopping service6");
    stopService(intentCallService6);
  }
 }// onClick
```

```
 public class MyEmbeddedBroadcastReceiver extends BroadcastReceiver {
  @Override
  public void onReceive(Context context, Intent intent) {
   Log.e("MAIN>>", "ACTION: " + intent.getAction());
   if (intent.getAction().equals("matos.action.GOSERVICE5")) {
    String service5Data = intent.getStringExtra("MyService5DataItem");
    Log.e("MAIN>>", "Data received from Service5: " + service5Data);
    txtMsg.append("\nService5Data: > " + service5Data);
   }
   else if (intent.getAction().equals("matos.action.GPSFIX")) {
    double latitude = intent.getDoubleExtra("latitude", -1);
    double longitude = intent.getDoubleExtra("longitude", -1);
    String provider = intent.getStringExtra("provider");
    String service6Data = provider + " lat: " + Double.toString(latitude)
                                    + " lon: " + Double.toString(longitude);
    Log.e("MAIN>>", "Data received from Service6:" + service6Data);
    txtMsg.append("\nService6Data: > "+ service6Data);
   }
  }//onReceive
 }// MyEmbeddedBroadcastReceiver
}// TestService4 class
```

EXAMPLES 3 (MyService4 – a music player)

```
public class MyService4 extends Service {
  public static boolean boolIsServiceCreated = false; MediaPlayer player;
  @Override public IBinder onBind(Intent intent) { return null; }
  @Override
  public void onCreate() {
    Toast.makeText(this, “MyService4 Created”, Toast.LENGTH_LONG).show();
    Log.e(“MyService4”, “onCreate”);
    boolIsServiceCreated = true;
    player = MediaPlayer.create(getApplicationContext(), R.raw.good_bad_ugly);
  }
  @Override
  public void onDestroy() {
    Toast.makeText(this, “MyService4 Stopped”, Toast.LENGTH_LONG).show();
    Log.e(“MyService4”, “onDestroy”);
    player.stop(); player.release(); player = null;
  }
  @Override
  public void onStart(Intent intent, int startid) {
    if (player.isPlaying()) Toast.makeText(this, “MyService4 Already Started ” + startid, Toast.LENGTH_LONG).show();
    else Toast.makeText(this, “MyService4 Started ” + startid, Toast.LENGTH_LONG).show();
    Log.e(“MyService4”, “onStart”);
    player.start();
  }
}
```

EXAMPLES 3 (MyService5Async – a slow Fibonacci number gen)

```
public class MyService5Async extends Service {
 boolean isRunning = true;
 private Handler handler = new Handler() {
  @Override
  public void handleMessage(Message msg) {
   super.handleMessage(msg);
   Log.e("MyService5Async-Handler", "Handler got from MyService5Async: " + (String)msg.obj);
  }
 };
 @Override public IBinder onBind(Intent arg0) { return null; }
 @Override public void onCreate() { super.onCreate(); }
 @Override
 public void onStart(Intent intent, int startId) {
  Log.e ("<<MyService5Async-onStart>>", "I am alive-5Async!");
  // we place slow work of service in an AsynTask so the response we send our caller who run
  // a "startService(...)" method gets a quick OK from us.
  new ComputeFibonacciRecursivelyTask().execute(20, 50);
 }//onStart
 // this recursive evaluation of Fibonacci numbers is exponential O(2^n)
 // for large n values it should be very time-consuming!
 public Integer fibonacci(Integer n){
  if ( n==0 || n==1 ) return 1;
  else return fibonacci(n-1) + fibonacci(n-2);
 }
```

```
 @Override
 public void onDestroy() { //super.onDestroy();
  Log.e ("<<MyService5Async-onDestroy>>", "I am dead-5-Async");
  isRunning = false;
 }//onDestroy
 public class ComputeFibonacciRecursivelyTask extends AsyncTask <Integer, Integer, Integer>{
  @Override
  protected Integer doInBackground(Integer... params) {
   for (int i=params[0]; i<params[1]; i++){ Integer fibn = fibonacci(i); publishProgress(i, fibn); }
   return null;
  }
  @Override
  protected void onProgressUpdate(Integer... values) {
   super.onProgressUpdate(values);
   Intent intentFilter5 = new Intent("matos.action.GOSERVICE5");
   String data = "dataItem-5-fibonacci-AsyncTask" + values[0] + ": " + values[1];
   intentFilter5.putExtra("MyService5DataItem", data);
   sendBroadcast(intentFilter5);
   // (next id not really needed!!! - we did the broadcasting already)
   Message msg = handler.obtainMessage(5, data);
   handler.sendMessage(msg);
  }
 }// ComputeFibonacciRecursivelyTask
}//MyService5
```

EXAMPLES 3
(MyService6 – a GPS service broadcasting locations)

```
public class MyService6 extends Service {
  String GPS_FILTER = “matos.action.GPSFIX”;
  Thread serviceThread;
  LocationManager lm;
  GPSListener myLocationListener;
  @Override
  public IBinder onBind(Intent arg0) { return null; }
  @Override
  public void onCreate() { super.onCreate(); }
  @Override
  public void onStart(Intent intent, int startId) {
    Log.e(“<<MyGpsService-onStart>>”, “I am alive-GPS!”);
    serviceThread = new Thread(new Runnable() {
      public void run() {
        getGPSFix_Version1(); // uses NETWORK provider
        getGPSFix_Version2(); // uses GPS chip provider
      }// run
    });
    serviceThread.start();
  }// onStart
```

```
public void getGPSFix_Version1() {
  // Get the location manager
  LocationManager locationManager = (LocationManager)getSystemService(Context.LOCATION_SERVICE);
  // work with best provider
  Criteria criteria = new Criteria();
  String provider = locationManager.getBestProvider(criteria, false);
  Location location = locationManager.getLastKnownLocation(provider);
  if ( location != null ){
    // capture location data sent by current provider
    double latitude = location.getLatitude(), longitude = location.getLongitude();
    // assemble data bundle to be broadcasted
    Intent myFilteredResponse = new Intent(GPS_FILTER);
    myFilteredResponse.putExtra(“latitude”, latitude);
    myFilteredResponse.putExtra(“longitude”, longitude);
    myFilteredResponse.putExtra(“provider”, provider);
    Log.e(“>>GPS_Service<<”, provider + “ =>Lat:” + latitude + “ lon:” + longitude);
    // send the location data out
    sendBroadcast(myFilteredResponse);
  }
}
```

EXAMPLES 3
(MyService6 – a GPS service broadcasting locations)

```
public void getGPSFix_Version2() {
  try {
    Looper.prepare();
    // try to get your GPS location using the LOCATION.SERVIVE provider
    lm = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
    // This listener will catch and disseminate location updates
    myLocationListener = new GPSListener();
    // define update frequency for GPS readings
    long minTime = 2000; // 2 seconds
    float minDistance = 5; // 5 meter
    // request GPS updates
    lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, minTime, minDistance, myLocationListener);
    Looper.loop();
  }
  catch (Exception e) { e.printStackTrace(); }
}
@Override
public void onDestroy() {
  super.onDestroy();
  Log.e("<<MyGpsService-onDestroy>>", "I am dead-GPS");
  try {
    lm.removeUpdates(myLocationListener); isRunning = false;
  }
  catch (Exception e) { Toast.makeText(getApplicationContext(), e.getMessage(), 1).show(); }
}// onDestroy
```

```
private class GPSListener implements LocationListener {
  public void onLocationChanged(Location location) {
  // capture location data sent by current provider
  double latitude = location.getLatitude(), longitude = location.getLongitude();
  // assemble data bundle to be broadcasted
  Intent myFilteredResponse = new Intent(GPS_FILTER);
  myFilteredResponse.putExtra("latitude", latitude);
  myFilteredResponse.putExtra("longitude", longitude);
  myFilteredResponse.putExtra("provider", location.getProvider());
  Log.e(">>GPS_Service<<", "Lat:" + latitude + " lon:" + longitude);
  // send the location data out
  sendBroadcast(myFilteredResponse);
 }
 public void onProviderDisabled(String provider) { }
 public void onProviderEnabled(String provider) { }
 public void onStatusChanged(String provider, int status, Bundle extras) { }
 };// GPSListener class
}// MyService3
```

EXAMPLES 3 (MANIFEST)

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="csu.matos" android:versionCode="1" android:versionName="1.0">
  <uses-sdk android:minSdkVersion="8" android:targetSdkVersion="15" />
  <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
  <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
  <application android:icon="@drawable/ic_launcher" android:label="@string/app_name" android:theme="@style/AppTheme" >
    <activity android:name=".TestService4" android:label="@string/title_activity_test_service4" android:screenOrientation="portrait">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
    <service android:name=".MyService4"/>
    <service android:name=".MyService5Async" />
    <service android:name=".MyService6" />
  </application>
</manifest>
```

EXAMPLES 3 (LAYOUT)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android" xmlns:tools="http://schemas.android.com/tools"
              android:layout_width="match_parent" android:layout_height="match_parent" >
 <LinearLayout    xmlns:android="http://schemas.android.com/apk/res/android"
                  android:layout_width="match_parent" android:layout_height="match_parent" android:orientation="vertical" >
  <Button         android:id="@+id/btnStart4" android:layout_width="wrap_content"
                  android:layout_height="wrap_content" android:ems="15" android:text="Start Service4 (Music Player)" />
  <Button         android:id="@+id/btnStop4" android:layout_width="wrap_content"
                  android:layout_height="wrap_content" android:ems="15" android:text="Stop Service4 (Music Player)" />
  <Button         android:id="@+id/btnStart5" android:layout_width="wrap_content"
                  android:layout_height="wrap_content" android:ems="15" android:text="Start Service5 (Fibonacci)" />
  <Button         android:id="@+id/btnStop5" android:layout_width="wrap_content"
                  android:layout_height="wrap_content" android:ems="15" android:text="Stop Service5 (Fibonacci)" />
  <Button         android:id="@+id/btnStart6" android:layout_width="wrap_content"
                  android:layout_height="wrap_content" android:ems="15" android:text="Start Service6 (GPS Fix)" />
  <Button         android:id="@+id/btnStop6" android:layout_width="wrap_content"
                  android:layout_height="wrap_content" android:ems="15" android:text="Stop Service6 (GPS Fix)" />
  <ScrollView     android:layout_width="match_parent" android:layout_height="wrap_content" >
   <TextView      android:id="@+id/txtMsg" android:layout_width="match_parent" android:layout_height="wrap_content" android:layout_margin="5dp" />
  </ScrollView>
 </LinearLayout>
</LinearLayout>
```