Mobility Programming

Lecture 4: Android Service

DR. RAMEZ ALKHATIB

Today

- Overview of Services in Android
- What they are?
- What would you use them for?
- And how to create them…

What are Services?

- An Application Component that
  - Has no UI
- Stay in the background
- Provide a long-running support for the app
- Services can do basically anything in the background:
  - Download/Upload Data
  - Listen for data changes (location, etc.) or Intents
  - Log information
  - MP3 Playback

why ?

- Runs in the background as normal even if the app is minimized
- Not on it’s own thread (unless explicitly programmed to do so)
- Exposes non-visual functionality to third parties
- Allows proper interprocess communication (if needed)

Services and Activities

- Activities can communicate with a service
- Or access the data collected by a service
- Think about the email service
- Checks for new mail
- Collects new mail and stores it somewhere
- Notifies user that there is new mail

Services and Activities

- User switches to the Inbox Activity
- Inbox Activity then fetches new mails and displays them
- Or MP3 playback
- Have activities that let you change the playing track or the volume

Creating a Service

- Mechanically similar to an Activity
  - Register the service in the manifest
  - Create a subclass of android.app.Service
- In practice, can get tricky…

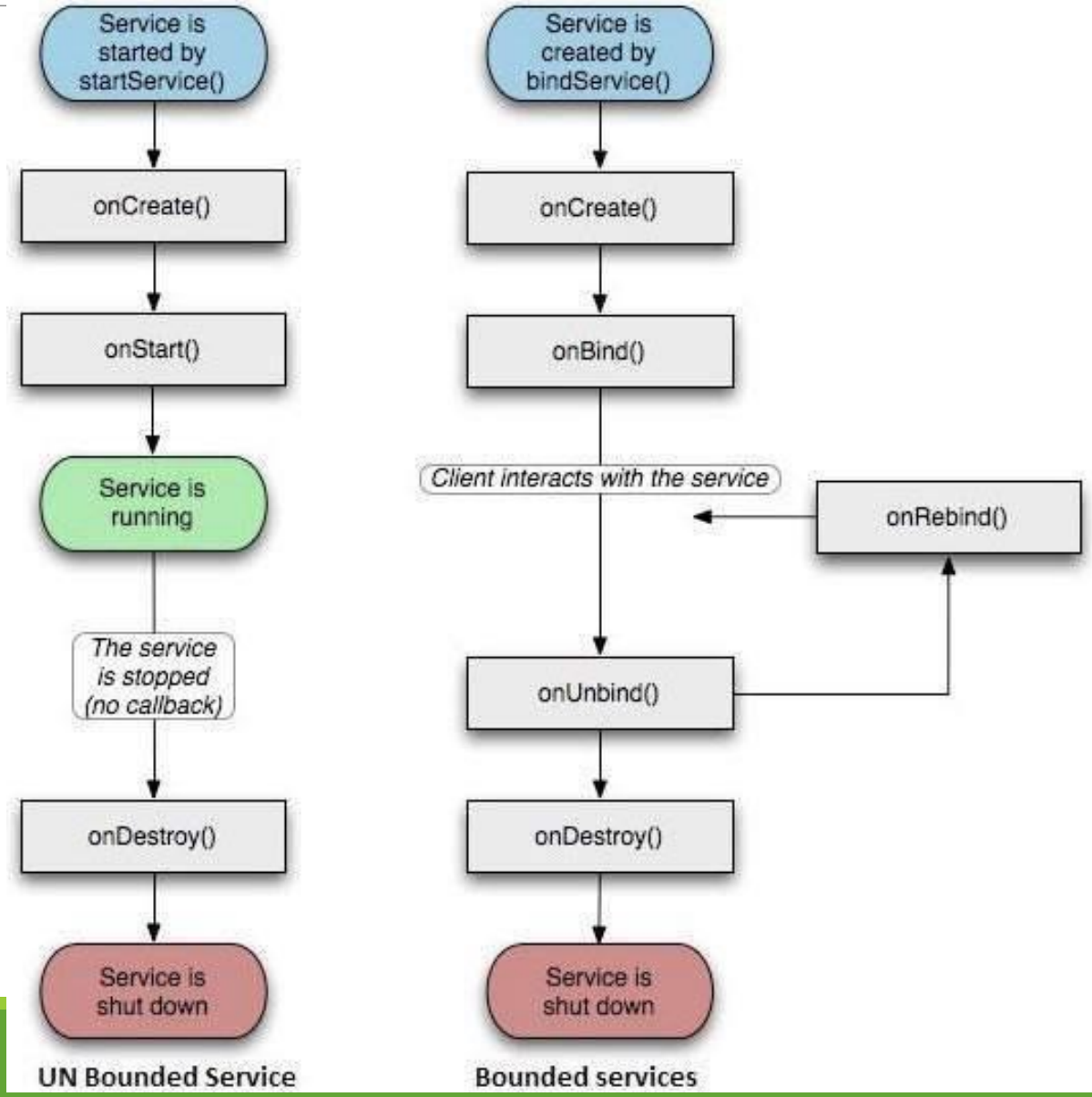
Services

- Services are designed to support communication with
  - Local Activities (in the same process)
  - Remote Activities
- Won’t consider remote access here but it has affected the design of Services

Service Lifecycle

- Two ways of starting a service
  - Either send an Intent with Context.startService()
  - Or, bind to a service using Context.bindService()
- In both cases, if the service is not running it will be created

Service Lifecycle



Service

- By nature, services are singleton objects
- The Service sub-class object is created if necessary
- Then onCreate() is called
  - Need to call the superclass method
- Then either onStartCommand or onBind will be called
- The **Service** base class defines various callback methods:
  - onStartCommand()- onBind()- onRebind()- onUnbind()- onCreate()- onDestroy()
  - You don't need to implement all the callbacks methods.
  - It's important that you understand each one and implement those that ensure your app behaves the way users expect

```
import android.app.Service;
import android.os.IBinder;
import android.content.Intent;
import android.os.Bundle;

public class HelloService extends Service {

   /** indicates how to behave if the service is killed */
   int mStartMode;

   /** interface for clients that bind */
   IBinder mBinder;

   /** indicates whether onRebind should be used */
   boolean mAllowRebind;

   /** Called when the service is being created. */
   @Override
   public void onCreate() {

   }

   /** The service is starting, due to a call to startService() */
   @Override
   public int onStartCommand(Intent intent, int flags, int startId) {
      return mStartMode;
   }

   /** A client is binding to the service with bindService() */
   @Override
   public IBinder onBind(Intent intent) {
      return mBinder;
   }
```

onCreate()

- The system calls this method when the service is first created using ***onStartCommand()*** or ***onBind()***.
- This call is required to perform one-time set-up.

```
/** Called when the service is being created. */
@Override
public void onCreate() {

}
```

onStartCommand()

- The system calls this method when another component, such as an activity, requests that the service be started, by calling ***startService().***

```
/** The service is starting, due to a call to startService() */
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
   return mStartMode;
}
```

- If you implement this method, it is your responsibility to stop the service when its work is done, by calling ***stopSelf()*** or ***stopService()*** methods.

onBind()

- The system calls this method when another component wants to bind with the service by calling ***bindService()***.
- If you implement this method, you must provide an interface that clients use to communicate with the service, by returning an *IBinder* object.
- You must always implement this method, but if you don't want to allow binding, then you should return *null*.

```
/** A client is binding to the service with bindService() */
@Override
public IBinder onBind(Intent intent) {
   return mBinder;
}
```

onUnbind()

- The system calls this method when all clients have disconnected from a particular interface published by the service.

```
/** Called when all clients have unbound with unbindService() */
@Override
public boolean onUnbind(Intent intent) {
   return mAllowRebind;
}
```

onRebind()

- The system calls this method when new clients have connected to the service, after it had previously been notified that all had disconnected in its ***onUnbind(Intent)***.

```
/** Called when a client is binding to the service with bindService()*/
@Override
public void onRebind(Intent intent) {

}
```

onDestroy()

- The system calls this method when the service is no longer used and is being destroyed.
- Your service should implement this to clean up any resources such as threads, registered listeners, receivers, etc.

```
/** Called when The service is no longer used and is being destroyed */
@Override
public void onDestroy() {

}
```

EXAMPLE

```
import android.os.Bundle;
import android.app.Activity;
import android.util.Log;
import android.view.View;

public class MainActivity extends Activity {
   String msg = "Android : ";

   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      Log.d(msg, "The onCreate() event");
   }

   public void startService(View view) {
      startService(new Intent(getBaseContext(), MyService.class));
   }

   // Method to stop the service
   public void stopService(View view) {
      stopService(new Intent(getBaseContext(), MyService.class));
   }
}
```

EXAMPLE

```
public class MyService extends Service {
    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        // Let it continue running until it is stopped.
        Toast.makeText(this, "Service Started", Toast.LENGTH_LONG).show();
        return START_STICKY;
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        Toast.makeText(this, "Service Destroyed", Toast.LENGTH_LONG).show();
    }
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.tutorialspoint7.myapplication">

   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      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" />
   </application>

</manifest>
```

res/layout/activity_main.xml file

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_par
   android:layout_height="match_parent" android:paddingLeft="@dimen/activity_hori
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".MainAc

   <TextView
      android:id="@+id/textView1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Example of services"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"
      android:textSize="30dp" />

   <TextView
      android:id="@+id/textView2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Tutorials point "
      android:textColor="#ff87ff09"
      android:textSize="30dp"
      android:layout_above="@+id/imageButton"
      android:layout_centerHorizontal="true"
      android:layout_marginBottom="40dp" />

   <ImageButton
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/imageButton"
      android:src="@drawable/abc"
      android:layout_centerVertical="true"
      android:layout_centerHorizontal="true" />

   <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/button2"
      android:text="Start Services"
      android:onClick="startService"
      android:layout_below="@+id/imageButton"
      android:layout_centerHorizontal="true" />

   <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Stop Services"
      android:id="@+id/button"
      android:onClick="stopService"
      android:layout_below="@+id/button2"
      android:layout_alignLeft="@+id/button2"
      android:layout_alignStart="@+id/button2"
      android:layout_alignRight="@+id/button2"
      android:layout_alignEnd="@+id/button2" />

</RelativeLayout>
```

```
<ImageButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/imageButton"
    android:src="@drawable/abc"
    android:layout_centerVertical="true"
    android:layout_centerHorizontal="true" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/button2"
    android:text="Start Services"
    android:onClick="startService"
    android:layout_below="@+id/imageButton"
    android:layout_centerHorizontal="true" />
```

Two types of services

- Default Service : Does not handle threads, must be done manually
- Intent Service : Handles requests one by one

```
public class HelloIntentService extends IntentService {

  /**
   * A constructor is required, and must call the super IntentService(String)
   * constructor with a name for the worker thread.
   */
  public HelloIntentService() {
      super("HelloIntentService");
  }

  /**
   * The IntentService calls this method from the default worker thread with
   * the intent that started the service. When this method returns, IntentService
   * stops the service, as appropriate.
   */
  @Override
  protected void onHandleIntent(Intent intent) {
      // Normally we would do some work here, like download a file.
      // For our sample, we just sleep for 5 seconds.

  }
}
```

More on services

- How about remote calls - or long running service ?
  - To be used if you require that the service is accessed by third party apps
  - Provide a messaging interface

```
Intent intent = new Intent(this, HelloService.class);
startService(intent);
```

Example 1 : Default Service (1/2)

How to communicate with a remote service

```
public class MessengerService extends Service {


    /**
     * Handler of incoming messages from clients.
     */
    class IncomingHandler extends Handler {
        @Override
        public void handleMessage(Message msg) {
            switch (msg.what) {
                case MSG_REGISTER_CLIENT:
                    mClients.add(msg.replyTo);
                    break;
                case MSG_UNREGISTER_CLIENT:
                    mClients.remove(msg.replyTo);
                    break;
                case MSG_SET_VALUE:
                    // dome something
                    break;
                default:
                    super.handleMessage(msg);
            }
        }
    }

    final Messenger mMessenger = new Messenger(new IncomingHandler());

    ....
    @Override
    public IBinder onBind(Intent intent) {
        return mMessenger.getBinder();
    }
```

Example 1 : Default Service (2/2)
How to communicate with a remote service

```
<service android:name=".app.MessengerService"
        android:process=":remote" />
```

```
// within an Activity
private ServiceConnection mConnection = new ServiceConnection() {
    public void onServiceConnected(ComponentName className,
            IBinder service) {

        mService = new Messenger(service);

        try {
            Message msg = Message.obtain(null,
                    MessengerService.MSG_REGISTER_CLIENT);
            msg.replyTo = mMessenger;
            mService.send(msg);

            // Give it some value as an example.
            msg = Message.obtain(null,
                    MessengerService.MSG_SET_VALUE, this.hashCode(), 0);
            mService.send(msg);
        } catch (RemoteException e) {
            // In this case the service has crashed before we could even
            // do anything with it; we can count on soon being
            // disconnected (and then reconnected if it can be restarted)
            // so there is no need to do anything here.
        }

        // As part of the sample, tell the user what happened.
        Toast.makeText(Binding.this, R.string.remote_service_connected,
                Toast.LENGTH_SHORT).show();
    }
```

Example 2 : Intent Service (1/2)
How to set up a notification

```
public class NotificationIntentService extends IntentService {
    private static final int NOTIFICATION_ID = 1;
    private static final String ACTION_ADD = "ACTION_ADD";
    private static final String INCIDENT_DESP = "INCIDENT_DESP";
    private static final String ROUTE_INCIDENT = "ACTION_ROUTE";
    private static final String ACTION_DELETE = "ACTION_DELETE";

    public NotificationIntentService() { super(NotificationIntentService.class.getSimpleName()); }

    /**...*/
    public static Intent createIntentAddNotificationService(Context context, String param) {...}

//...

    public static Intent createIntentDeleteNotification(Context context) {...}

    @Override
    protected void onHandleIntent(Intent intent) {...}

    private void processAddNotification(String param) {...}

    private void processRouteIncidentNotificationService(String param) {...}

    private void processDeleteNotification() {
        // TODO: Handle action Baz
        throw new UnsupportedOperationException("Not yet implemented");
    }
}
```

Example 2 : Intent Service (2/2)
How to set up a notification

```
@Override
protected void onHandleIntent(Intent intent) {
    Log.d(getClass().getSimpleName(), "onHandleIntent, started handling a notification event");
    if (intent != null) {
        try {
            String action = intent.getAction();
            if (ACTION_ADD.equals(action)) {
                processAddNotification(intent.getStringExtra(INCIDENT_DESP));
            }
            if (ROUTE_INCIDENT.equals(action)) {
                processRouteIncidentNotificationService(intent.getStringExtra(INCIDENT_DESP));
            }
            if (ACTION_DELETE.equals(action)) {
                processDeleteNotification();
            }
        } finally {
            WakefulBroadcastReceiver.completeWakefulIntent(intent);
        }
    }
}
```

Questions?