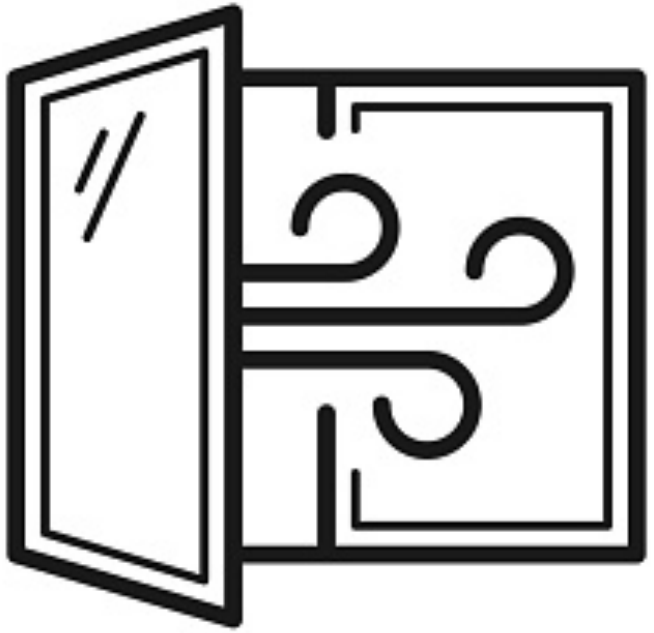
Android App Programming

Lecture 4: Services
November 2, 2021

Anastasia Bezerianos
Université Paris-Saclay

course based in part on that of Thomas Nowak

in class



Last lecture recap

Web services

- web service = set of responses from a web server that aren't HTML
- return formats: typically JSON or XML
- query formats: often HTTP GET request w/ query strings
  - ex: https://www.google.com/search?q=hi
- find web APIs: www.programmableweb.com

Services

Services

- for background tasks, repetitive or long operations
- can run even if application is not visible (but they are in the same thread as the application that started them)
- ex: downloads (that should continue after the app quits)
- ex: music player
- ex: polling an SMTP for new emails

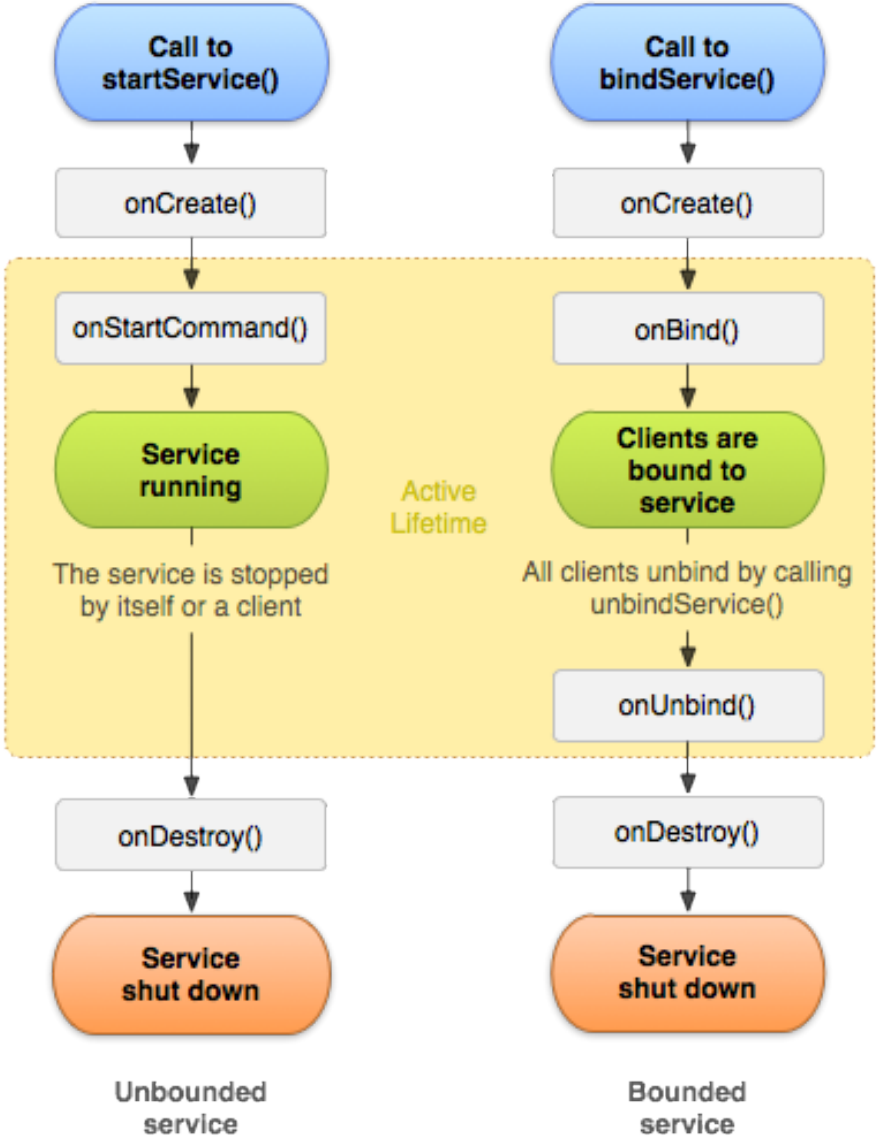
Services

- run at higher priority than inactive or invisible activities
- less likely to be terminated
- can be configured to restart when terminated
- Android already runs several System Services that apps can use (given the right permissions). Examples that may be running are Android Play Service , Bluetooth Service, Location Manager Service, Battery Service, etc.

Services

- Multiple types of services:
  - Started / Unbound Service (started by another service or activity, runs indefinitely unless stopped by a client or itself). Started services cannot be bound.
  - Bound Services (runs as long as there's one activity active that bound it)
  - Foreground / Background services

Service Lifecycle



Started Services

- For continuous, arbitrary time tasks …
- extend `Service` class
- override `onStartCommand` method, normally returning `START_STICKY`
- `START_STICKY` service explicitly started and stopped to run for arbitrary periods of time, such as a service performing background music playback. Restarts if terminated by the system
- override `onBind` method returning `null`
- start service by creating an `Intent` and calling `startService` (or `startForegroundService`)

Coding demo

Started service running continuously

Lab 4: Ex 1

MyService

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

public class MyService extends Service {

    MediaPlayer myPlayer; // using Android media player

    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
    public void onCreate() {
        Toast.makeText(this, "Service Created", Toast.LENGTH_LONG).show();
        // Toast is a Notification widget
        myPlayer = MediaPlayer.create(this, R.raw.file_example_mp3_700kb);
        // load music file with player
        myPlayer.setLooping(false); // Set looping for the player
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        Toast.makeText(this, "Service Started", Toast.LENGTH_LONG).show();
        // Toast is a Notification widget
        myPlayer.start(); // start the music
        return Service.START_STICKY; // start the music
    }

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

onBind() should return nothing

setup the player

start the music

The service will try to restart if stopped

stop the music

MainActivity

```
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
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 startMusic(View view) {
        Intent music_i = new Intent(this, MyService.class);
        startService(music_i);
    }

    public void stopMusic(View view) {
        Intent music_i = new Intent(this, MyService.class);
        stopService(music_i);
    }

    public void secondActivity(View view) {
        Intent second_i = new Intent(this, secondActivity.class);
        startActivity(second_i);
    }
}
```

Use intents to communicate with the started Service.

Use methods `startService (Intent)` and `stopService(Intent)` to start a Started service.

Aside

- More on Toast notifications
- https://developer.android.com/guide/topics/ui/notifiers/toasts

- More on MediaPlayer
- https://developer.android.com/guide/topics/media/mediaplayer

Bound Services

- are the server in a client-server interface
- activities can bind to the service, send requests, receive responses, and perform interprocess communication (IPC)
- lives while it serves another application component and does **not** run in the background indefinitely
- better if you have a frequent communication with the service

Bound Services

- Bound services are started when a component binds to them (just binding also starts them)
- the stop when they are no longer bound by any component
- (aside: more rarely, they can be started if another components starts them as a started service, e.g. through `startService`, and **then** binds to them)

Bound Services

- create a `Service` class
- add an **inner class** that extends `Binder` and that returns a reference to the service instance, so that clients can call methods of the service
- keep an instance of the inner class binder
- override the `onBind` method to return the instance of the inner class binder
- add any processing methods
- good policy to `stopSelf()` when `onTaskRemoved` is called (otherwise the service may run if it is never ubound)

Bound Services

- from your Activities you can bind to the service by …
- create a reference to the service
- create a `ServiceConnection` object to communicate with service
- start the service with an Intent and `bindService`
- unbind the service when you no longer need it (e.g., when your activity stops)

Coding demo

Started service for one task

Lab 4: Ex 2

MyBoundedService

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

import java.util.Random;

public class MyBoundedService extends Service {

    private IBinder mBinder = new MyBinder();

    private final Random mGenerator = new Random();

    public class MyBinder extends Binder {
        MyBoundedService getService(){
            return MyBoundedService.this;
        }
    }

    @Override
    public IBinder onBind(Intent intent) {
        return mBinder;
    }

    @Override
    public void onTaskRemoved(Intent rootIntent) {
        super.onTaskRemoved(rootIntent);
        stopSelf();
    }

    /** method for clients */
    public int getRandomNumber() {
        return mGenerator.nextInt(100);
    }
}
```

Return an instance of this Service so clients can call public methods through a class Binder

Return the Binder

In case does not get unbound, to prevent it from running forever

public methods offered by this service

MainActivity

```
import androidx.appcompat.app.AppCompatActivity;

import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.os.Bundle;
import android.os.IBinder;
import android.view.View;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    MyBoundedService mService;
    boolean mBound = false;

    private ServiceConnection connection = new ServiceConnection() {
        @Override
        public void onServiceConnected(ComponentName className, IBinder service) {
            // We've bound to Service, cast the IBinder and get MyBoundedService instance
            MyBoundedService.MyBinder binder = (MyBoundedService.MyBinder) service;
            mService = binder.getService();
            mBound = true;
        }
        @Override
        public void onServiceDisconnected(ComponentName arg0) {
            mBound = false;
        }
    };

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

    @Override
    protected void onStart() {
        super.onStart();
        Intent intent = new Intent(this, MyBoundedService.class);
        bindService(intent, connection, Context.BIND_AUTO_CREATE);
    }

    @Override
    protected void onStop() {
        super.onStop();
        unbindService(connection);
        mBound = false;
    }

    public void readRandomNumberFromService(View view) {
        if (mBound) {
            int num = mService.getRandomNumber();
            Toast.makeText(this, "number: " + num, Toast.LENGTH_SHORT).show();
        }
    }
}
```

anonymous inner ServiceConnection class. We keep an instance in the variable connection.

It uses the Binder we have defined in our service to connect to it

binding (and starting) the service in a way that creates it (BIND_AUTO_CREATE) through the SeviceConnection connection

unbind

ask data from the service

Async work

- for one time or periodic requests
- running on separate threads
  - work manager
  - broadcasting Intents (deprecated since Android 23)

Work Managers

- Library that enqueues requests. A class extends `Worker` and you need to implement `doWork()`
- From an activity or other component you need to make a work request `OneTimeWorkRequest` or `PeriodicWorkRequest`

```
r = OneTimeWorkRequest.Builder(myWorker.class).build();
```

- And then enqueued `WorkManager.getInstance().enqueue(r);`
- Library needs to be declared in build.gradle Module dependencies

```
def work_version = "2.5.0" implementation "androidx.work:work-runtime:$work_version"
```

Coding demo

Started service for one task

Lab 4: Ex 3

Worker

```
public class MyWorker extends Worker {

    private final Random mGenerator = new Random();

    public MyWorker(@NonNull Context context, @NonNull WorkerParameters workerParams) {
        super(context, workerParams);
    }

    @NonNull
    @Override
    public Result doWork() {
        int int_result = getRandomNumber();
        String output = "result is " + int_result;
        Log.d("MyRandomService", output);
        return Result.success();
    }

    int getRandomNumber(){
        return mGenerator.nextInt(100);
    }
}
```

MainActivity

```
import androidx.appcompat.app.AppCompatActivity;
import androidx.work.PeriodicWorkRequest;
import androidx.work.WorkManager;

import android.os.Bundle;

import java.util.concurrent.TimeUnit;

public class MainActivity extends AppCompatActivity {

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

    // submit your request to the system, for example onCreate or when a user clicks a button
    // could also do a PeriodicWorkRequest but only every 15min
        OneTimeWorkRequest random_number_request = new OnTimeWorkRequest.Builder(MyWorker.class).build();

    // submit your request to the system, for example onCreate or when a user clicks a button
        WorkManager.getInstance().enqueue(random_number_request);
    }
}
```

Work Managers

- If you want to read & practice more, a detailed tutorial (90min)
- https://developer.android.com/codelabs/android-workmanager-java#0

Broadcasting Intents deprecated

- ~~used to communicate from service to activity.~~
- ~~true background task (has its own thread)~~
- ~~often for one time tasks in the background, such as downloading a file~~

Intent Services

- ~~extend `IntentService` class~~
- ~~override `onHandleIntent` method~~
- ~~override `onBind` method returning `null`~~
- ~~define Actions in your service~~
- ~~put data into the extra bundle, then call `sendBroadcast`~~

Intent Services

- ~~start service (e.g., in an Activity) by creating an `Intent` for the IntentService class and `startService`~~
- ~~register a BroadcastReceiver with the method `registerReceiver` for the action you want~~
- ~~you can also unregister with `unregisterReceiver`~~

Lab 4

1. Complete Exercise 1 & 2 following the class video
2. Complete exercise 3 using the code demo in slides
3. Complete exercise 4

Advance reading

- Next week's topic:
- Databases
- Read Ch. 55

Questions