آموزش برنامه نویسی اندروید
در محیط اندروید استودیو

آشنایی با Service ها در اندروید

مدرس : سیدمهدی مطهری

www.android-studio.ir

به نام خدا

در توسعه و برنامه نویسی اپلیکیشن‌های اندرویدی برای اجرای پردازش‌هایی که طولانی مدت بوده و یا لازم است در پس زمینه هم در حال اجرا باقی بمانند از کامپوننتی به نام Service (سرویس) استفاده می‌شود. در این مبحث ابتدا به معرفی کامپوننت Service در اندروید پرداخته سپس نحوه پیاده سازی آن را در قالب یک پروژه بررسی می‌کنیم.

معرفی Service در اندروید

Service یکی از کامپوننت‌های پرکاربرد در سیستم عامل اندروید محسوب می‌شود. از کامپوننت سرویس برای اجرای عملیات و پردازش‌های طولانی مدت و تکرار شونده در پس زمینه (Background) استفاده می‌شود. پردازش‌هایی که ارتباطی با رابط کاربری (UI) نداشته و باید بدور از چشم کاربر انجام شود. منظور از اجرا در پس زمینه این است که مهم نیست برنامه باز باشد یا بسته. یعنی بعد از اجرای سرویس، حتی اگر کاربر از برنامه ما خارج شد و برنامه دیگری را باز کرد هم سرویس در پشت صحنه به کار خود ادامه می‌دهد. به طور خلاصه می‌توان گفت سرویس‌ها هیچ وابستگی به چرخه حیات Activity ندارد.

احتمالا می‌پرسید این قابلیت در چه مواردی بکار می‌رود؟ با چند مثال توضیح می‌دهم:

یک برنامه پخش موزیک را درنظر بگیرید. در حالت عادی اگر پخش فایل‌های صوتی داخل یک اکتیویتی انجام شود، به محض خروج کاربر از برنامه و یا حتی در صورتی که یک اکتیویتی دیگر از برنامه را باز

کند، پخش صوت متوقف خواهد شد. در حالی که کاربر انتظار دارد با خروج از برنامه یا رفتن به سایر صفحات برنامه (مانند لیست آلبوم‌ها و...) بازهم پخش موسیقی ادامه پیدا کند.

برنامه مدیریت دانلود (Download Manager) یک مثال دیگر برای کاربرد Service در اندروید است. زمانی که کاربر در حال کار با برنامه دیگری است یا حتی هنگامی که دستگاه در حالت idle (بیکاری) قرار دارد هم این برنامه باید بتواند فایلی که دانلود آن قبلا آغاز شده را در پس زمینه ادامه داده و با خروج کاربر از برنامه و متوقف شدن اکتیویتی، عملیات دانلود متوقف نشود. به عبارت دیگر چنانچه برنامه نویس کامپوننت سرویس را در برنامه پیاده سازی نکند، کاربر برای دانلود فایل مجبور است تا اتمام فرایند دانلود در برنامه و صفحه‌ی دانلود باقی بماند!

امیدوارم توانسته باشم فلسفه‌ی Service را بدرستی شرح دهم.

Service در اندروید با اولویت بالاتری نسبت به Activity های غیرفعال و یا نامرئی اجرا می‌شوند، بنابراین احتمال اینکه توسط اندروید بسته شوند کمتر است.

انواع Service در اندروید

به طور کلی سرویس‌ها در اندروید دو نوع هستند:

۱: Started Services: سرویسی است که با شروع شدنش فقط کاری که تعریف شده را انجام می‌دهد و پس از آن هیچ ارتباطی با آن نداریم تا زمانی که عمل مدنظر انجام شده و متوقف می‌گردد. این سرویس با فراخوانی متد ()startService یا ()startForegroundService از طریق یک کامپوننت اندرویدی (مانند Activity) استارت شده و توسط ()stopService یا ()stopSelf متوقف می‌شود. به عبارت دیگر در این حالت با استفاده از intent فقط می‌توانیم سرویس را راه اندازی و متوقف کنیم و هیچ کنترل دیگری روی آن نداریم.

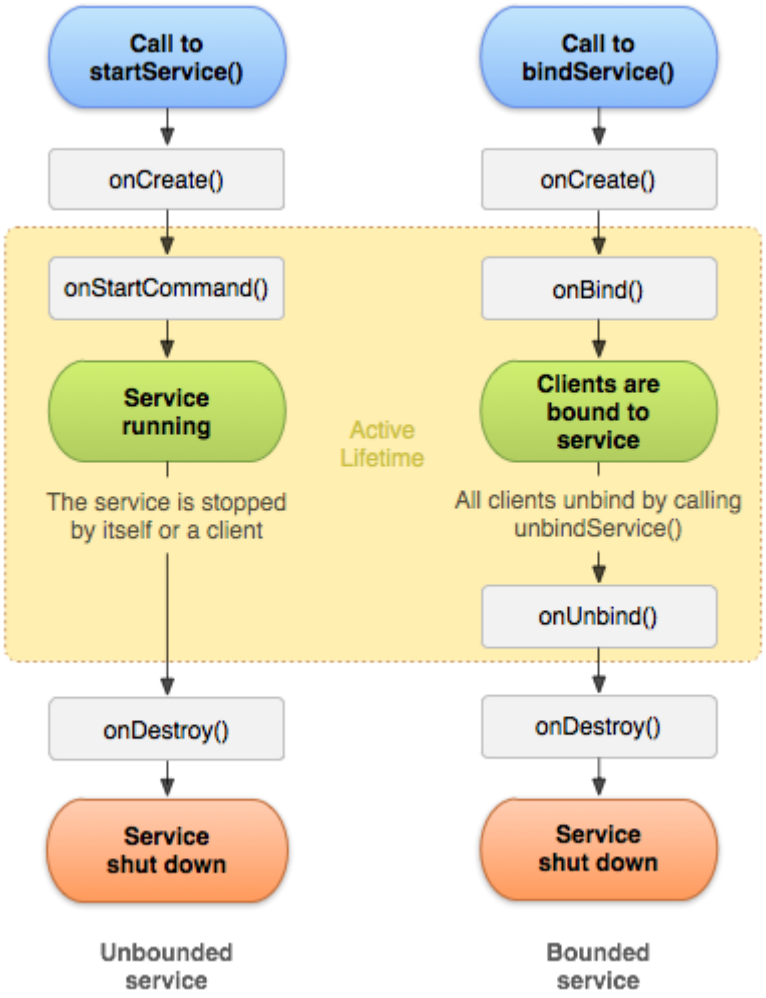
برای مثال با استفاده از این سرویس می‌توانیم یک فایل صوتی را play کنیم. تنها کاری که بعد از استارت سرویس می‌توانیم انجام دهیم، متوقف کردن آن است که باعث می‌شود پخش فایل صوتی نیز متوقف شود. اما در این بین هیچ دسترسی و ارتباطی با سرویس نداریم تا برای مثال بتوانیم چک کنیم چند ثانیه تا انتهای فایل صوتی باقی مانده است.

۲: Bound Services: برخلاف Started Service، در Bound Service کامپوننت اندرویدی مانند اکتیویتی یا کلاینت، امکان تعامل با سرویس را داشته و دائما پیغام‌های بین سرویس و کامپوننت مبادله می‌شود. در واقع یک Bound Service یک واسط کلاینت سرور (client-server) ارائه می‌دهد و به کامپوننت‌ها اجازه می‌دهد تا با Service ارتباط برقرار کنند، درخواست‌های خود را به آن ارسال و

همچنین نتایج را نیز دریافت نمایند. با استفاده از متد bindService() ارتباط بین کامپوننت و سرویس برقرار شده و توسط unbindService() این ارتباط قطع می‌گردد.

برای مثال توسط این سرویس می‌توانیم بررسی کنیم موزیک در حال پخش در چه وضعیتی قرار دارد. یا فایلی که دانلود آن قبلا آغاز شده چند درصد پیشرفت داشته و چند درصد از اتمام عملیات دانلود باقی مانده است.

چرخه حیات سرویس در اندروید مطابق تصویر زیر است:



در این آموزش فقط Started Service را بررسی می‌کنیم. به امید خدا Bound Service را در آینده و در قالب آموزش‌های پروژه محور بررسی خواهیم کرد.

انواع Started Service ها

Started Service نیز به دو دسته تقسیم می‌شود که ابتدا به طور مختصر توضیحاتی را بیان می‌کنم. در ادامه و به طور مفصل هردو حالت را تمرین و بررسی می‌کنیم.

۱: Background Service: سرویسی است که پردازشی را بدون اطلاع کاربر در پس زمینه انجام می‌دهد.

۲: Forground Service: سرویسی است که پردازش را توسط یک نوتیفیکیشن (Notification) به کاربر اطلاع می‌دهد.

توضیحات تکمیلی هرکدام را به ادامه مبحث موکول می‌کنم.

مطابق مبحث آموزش ساخت پروژه جدید در اندروید استودیو یک پروژه با نام StartService با یک اکتیویتی از نوع Empty Activity و زبان Java ایجاد می‌کنم. همچنین Min SDK را روی API 19 قرار دادم.

در این تمرین قصد دارم توسط Started Service یک فایل صوتی را در پس زمینه اندروید اجرا کنم به طوری که با بسته شدن اکتیویتی یا برنامه، پخش صوت متوقف نشود.

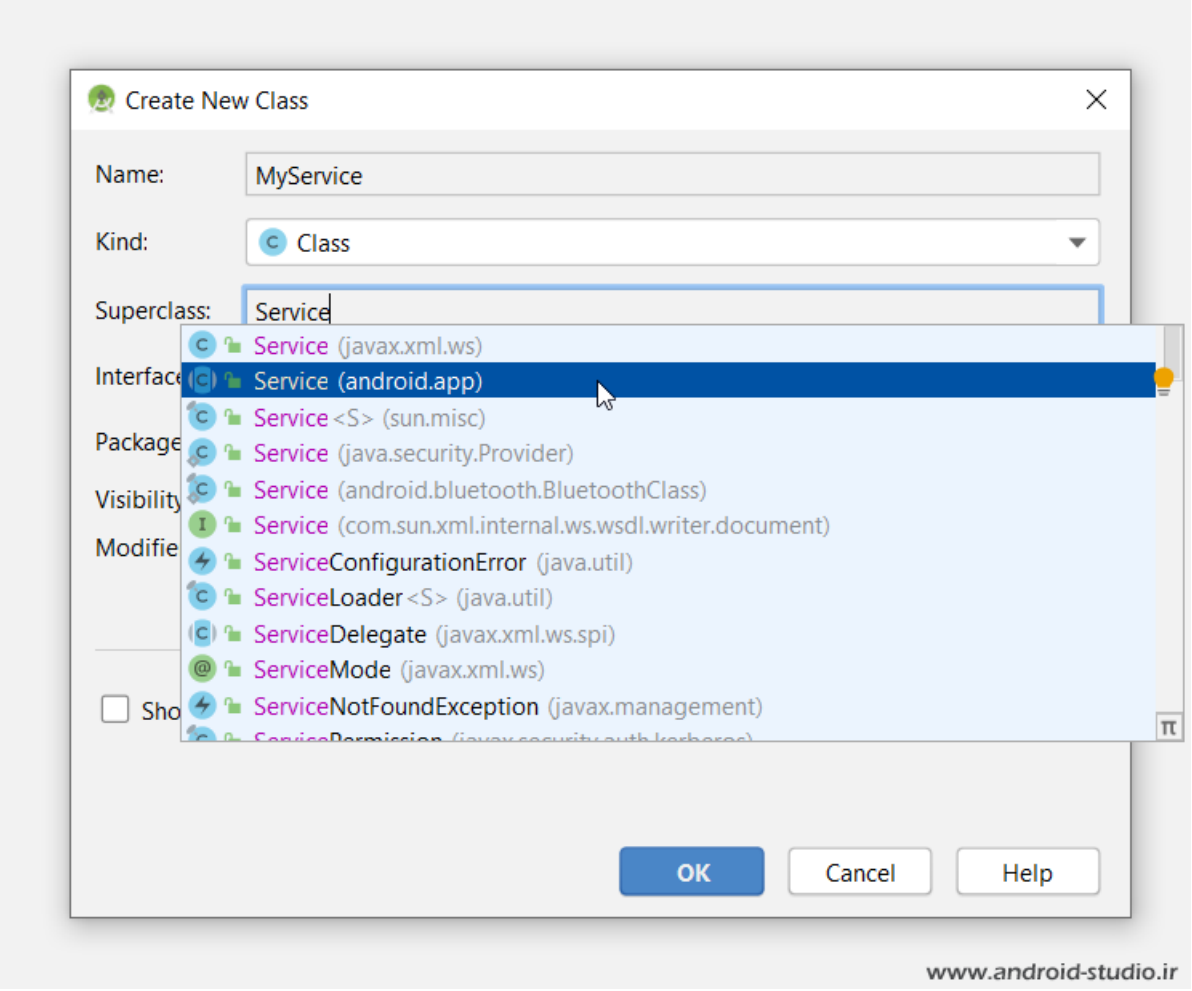
ابتدا سه Button در Layout اکتیویتی پروژه تعریف می‌کنم:

activity_main.xml

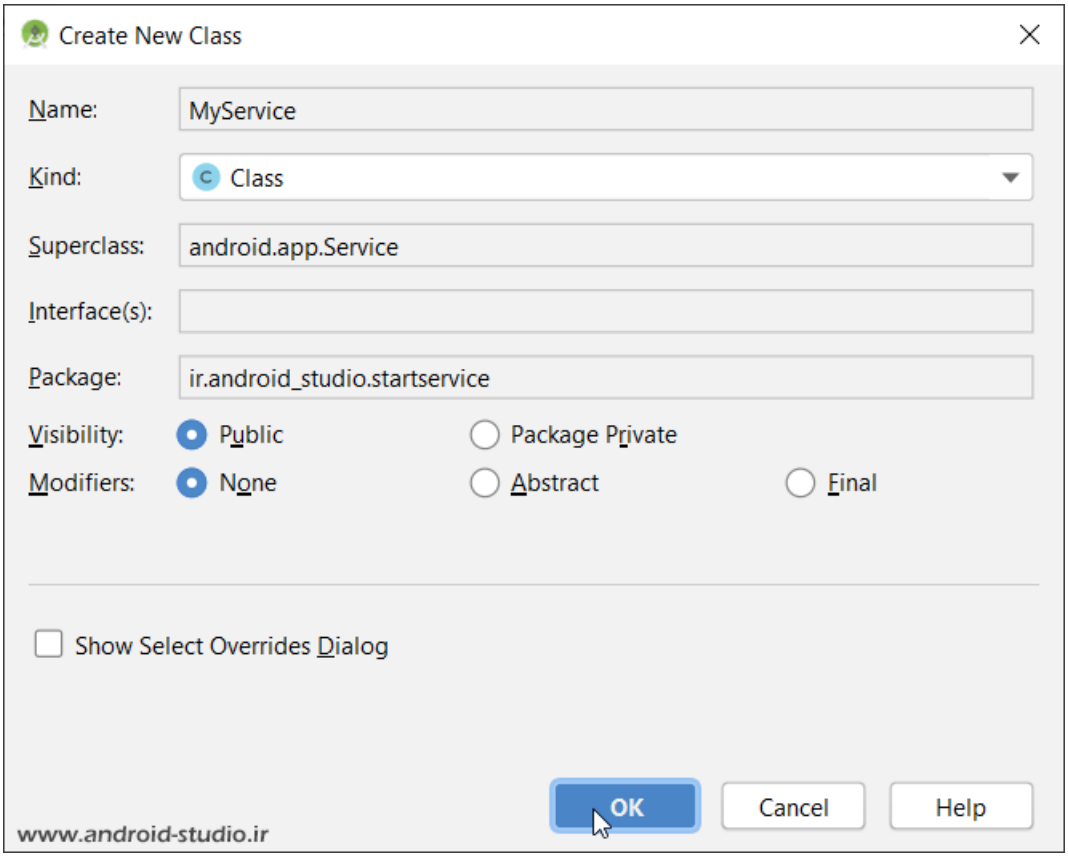
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Start"
        android:id="@+id/start_btn"
        android:layout_marginTop="30dp"
        android:layout_gravity="center" />

    <Button
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:text="Stop"
        android:id="@+id/stop_btn"
        android:layout_marginTop="30dp"
        android:layout_gravity="center" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Activity Two"
        android:id="@+id/two_btn"
        android:layout_marginTop="30dp"
        android:layout_gravity="center" />

</LinearLayout>
```

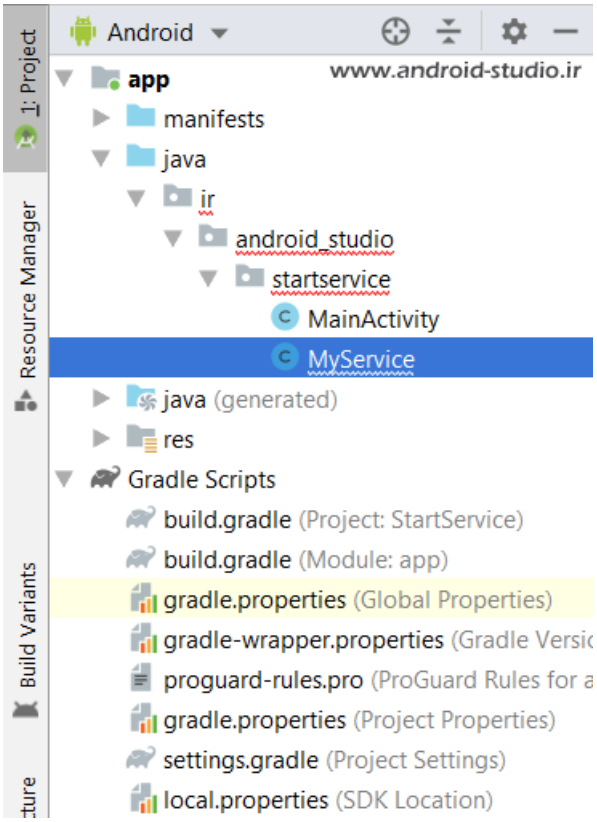
در قدم بعد یک کلاس از نوع Service به پروژه اضافه می‌کنم. برای اینکار مانند اضافه کردن یک کلاس معمولی اقدام می‌کنیم. روی فولدر پکیج پروژه راست کلیک کرده سپس از مسیر New > Java Class پنجره ساخت کلاس را باز می‌کنم:



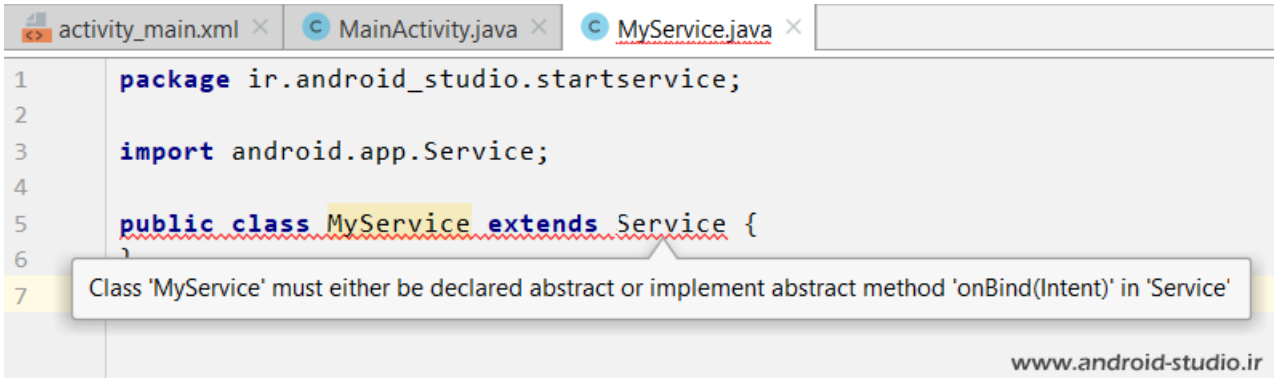
برای کلاس Service نام دلخواه MyService را انتخاب کردم. این کلاس باید از کلاس Service در اندروید ارث بری (extend) شود بنابراین در فیلد Superclass عبارت Service را وارد کرده و گزینه مربوط به android.app را انتخاب می‌کنم:



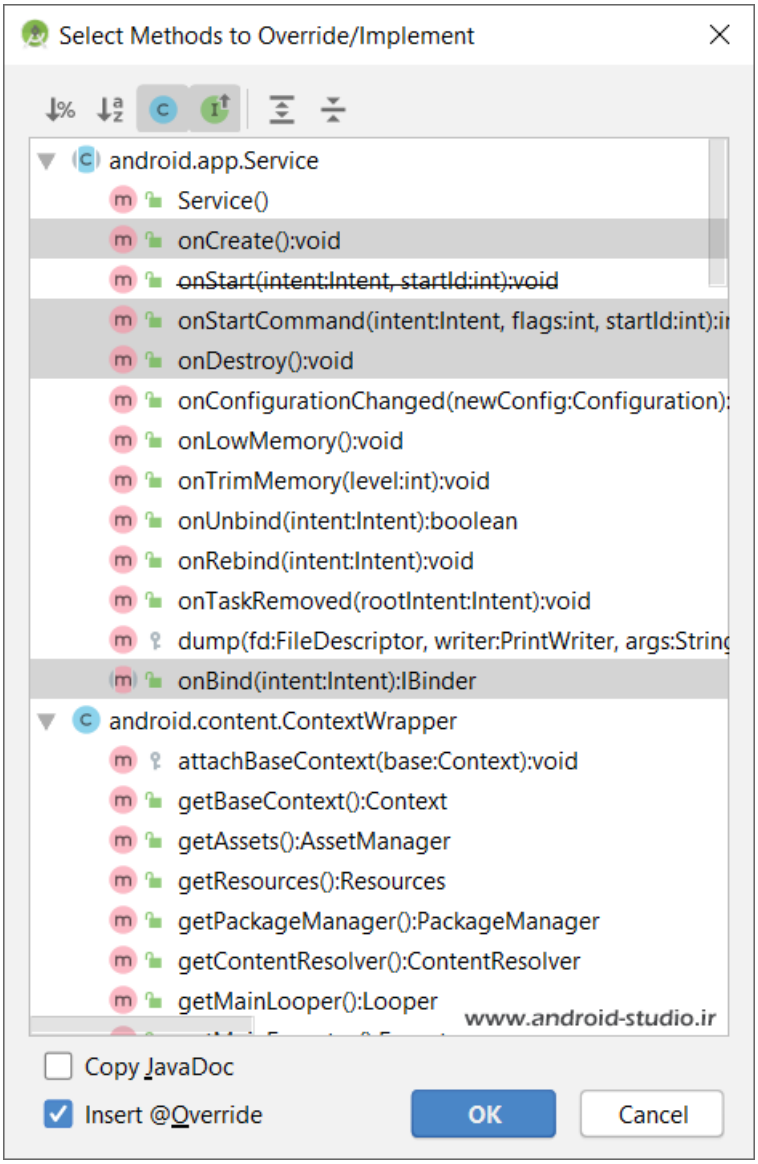
بدون تغییر سایر گزینه‌ها، کلاس را می‌سازم. کلاس در کنار MainActivity به پروژه اضافه شد:



در حال حاضر یک اخطار در کلاس سرویس داریم:



کلاس سرویس چندین متد callback دارد که به تناسب نیاز هر پروژه تعدادی از آنها را استفاده می‌کنیم. اما متد onBind حتما باید تعریف شود. در جلسات قبل بجای Override کردن دستی متدها، با استفاده از کلیدهای ترکیبی alt + insert و انتخاب گزینه Override Methods به صورت خودکار متدها را به کلاس اضافه می‌کردیم:

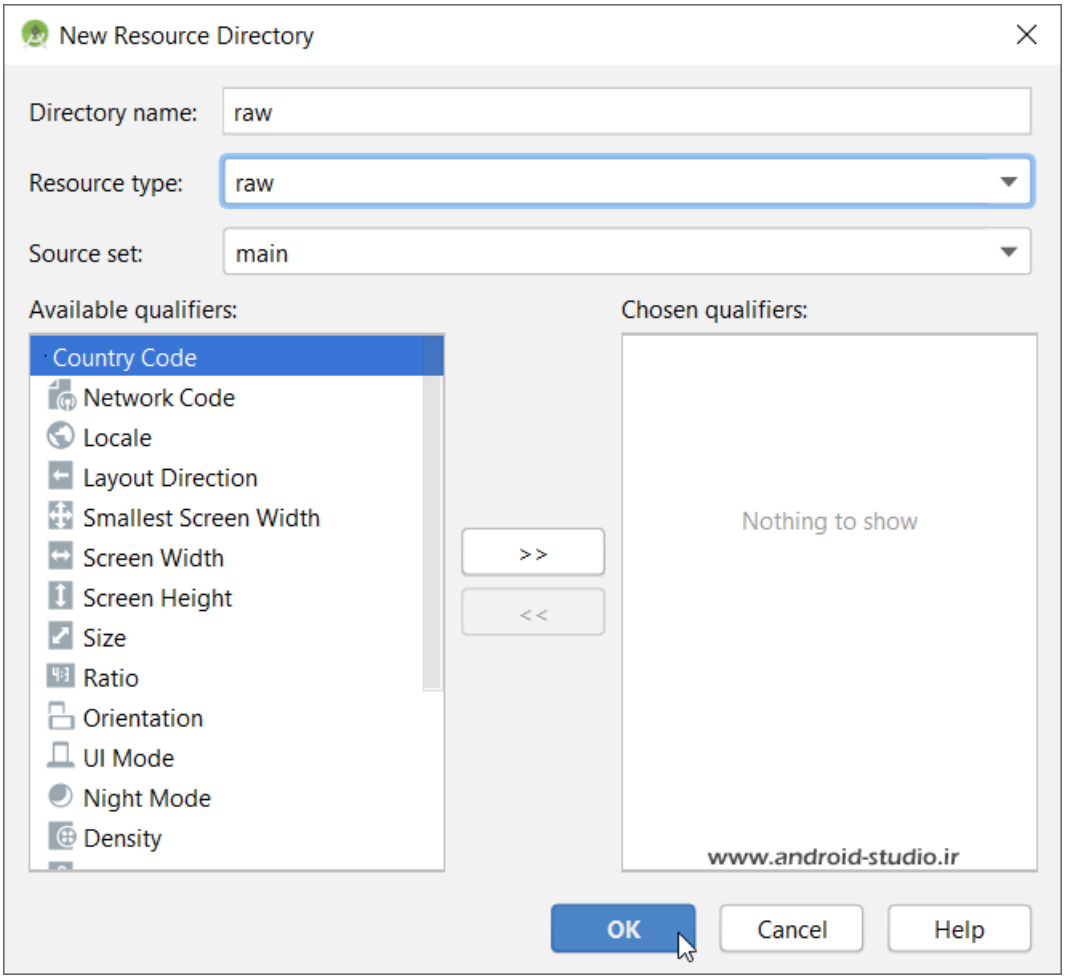


مطابق تصویر فوق من چهار متد onCreate، onStartCommand، onDestroy و onBind را به کلاس سرویس اضافه کردم:

MyService.java

```
package ir.android_studio.startservice;

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

import androidx.annotation.Nullable;

public class MyService extends Service {
    @Override
    public void onCreate() {
        super.onCreate();
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        return super.onStartCommand(intent, flags, startId);
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
    }

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

قبل از تکمیل متدهای سرویس ابتدا یک فایل صوتی به پروژه اضافه می‌کنم. روی فولدر res پروژه راست کلیک کرده، New و سپس یکی از گزینه‌های Android Resource Directory یا Directory را انتخاب می‌کنم. در مرحله بعد با انتخاب گزینه raw یا نوشتن دستی آن، فولدر raw به فولدر res اضافه می‌شود:



حالا یک فایل mp3 را به raw منتقل می‌کنم:

در مرحله بعد، کلاس Service را به صورت زیر تکمیل می‌کنم:

MyService.java

```
package ir.android_studio.startservice;

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

import androidx.annotation.Nullable;

public class MyService extends Service {

    private MediaPlayer soundPlayer;

    @Override
    public void onCreate() {

        Toast.makeText(this, "سرویس ساخته شد", Toast.LENGTH_SHORT).show();
        soundPlayer = MediaPlayer.create(this, R.raw.song);
        soundPlayer.setLooping(false);

    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {

        Toast.makeText(this, "سرویس استارت شد", Toast.LENGTH_SHORT).show();
        soundPlayer.start();
        return START_STICKY;

    }

    @Override
    public void onDestroy() {

        Toast.makeText(this, "سرویس متوقف شد", Toast.LENGTH_SHORT).show();
        soundPlayer.stop();

    }

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

ابتدا درون کلاس یک نمونه از متد MediaPlayer با نام دلخواه soundPlayer تعریف کردم. از این متد برای پخش فایل‌های صوتی در اندروید استفاده می‌شود که قبلا در آموزش ساخت Splash Screen با آن آشنا شدیم.

هنگامی که یک سرویس برای اولین بار توسط startService() یا bindService() ساخته می‌شود متد onCreate فراخوانی خواهد شد. بنابراین در این متد فقط کدهایی که باید یکبار اجرا شوند را تعریف می‌کنیم. همانطور که ملاحظه می‌کنید من در این متد با استفاده از MediaPlayer.create محل قرارگیری فایل صوتی را تعیین کرده‌ام. همچنین در خط بعد توسط setLooper و مقدار false تعیین کردم که فایل صوتی پس از اتمام مجدد تکرار نشود. قطعا این دو دستور فقط یکبار نیاز به اجرا دارد و لازم نیست در هر بار استارت سرویس کدها فراخوانی شوند.

همانطور که اشاره شد می‌خواهیم سرویس توسط startService() فراخوانی شود بنابراین دستورات مربوط به زمان اجرای سرویس باید درون متد onStartCommand تعریف شود. بعد از ساخته شدن سرویس، این متد فراخوانی می‌شود. توسط دستور soundPlayer.start() فرمان play کردن فایل صوتی صادر می‌شود.

متد onStartCommand باید یک مقدار integer را return کند. این مقدار به سیستم اعلام می‌کند که سرویس پس از متوقف شدن باید در چه حالتی قرار گیرد. دو مقدار اصلی را در ادامه توضیح می‌دهم:

۱: START_NOT_STICKY: در این حالت بعد از توقف سرویس توسط سیستم، سرویس را در حالت ساخته شده نگه نمی‌دارد و صرفا هنگامی سرویس دوباره ساخته خواهد شد که startService() فراخوانی شود. این گزینه مناسب سرویس‌هایی است که یک کار موقت را انجام داده و نیاز به تکرار ندارند.

۲: START_STICKY: بعد از توقف سرویس توسط سیستم، سرویس را در حالت ساخته شده نگه می‌دارد. یعنی متد onCreate و onStartCommand را بلافاصله مجدد اجرا می‌کند ولی مقدار intent ای که برای آن درنظر می‌گیرد برابر با null است و نه intent ای که در مرتبه قبل برای آن ارسال شده. در این صورت شاهد سرعت بیشتری در اجرای درخواست‌های بعدی خواهیم بود که از کامپوننت دریافت می‌شوند.

۳: START_REDELIVER_INTENT: در این حالت علاوه بر اینکه مانند حالت دوم، سرویس را در حالت ساخته شده نگه می‌دارد، intent را هم برابر آخرین intent ای که به متد پاس داده شده قرار می‌دهد. این گزینه مناسبترین انتخاب برای مواردی است که سرویس باید عملیات را از جایی که قبلا متوقف شده ادامه دهد. مانند ادامه پخش یک فایل صوتی یا دانلود یک فایل.

با توجه به توضیحات فوق، گزینه دوم برای این تمرین مناسب تر است. هرچند با توجه به سرعت بالای پردازش و حجم کم کدهای این پروژه، عملا تفاوتی را در خروجی کار مشاهده نخواهیم کرد اما همواره باید بهینه ترین گزینه را انتخاب کنیم.

در نهایت و در متد onDestroy سرویس، پخش فایل صوتی stop شده است. متد onDestroy هنگامی اجرا می‌شود که ()stopService از سمت کامپونتی مانند اکتیویتی یا کلاینت فراخوانی شود.

متد onBind در این تمرین بدون تغییر باقی مانده و مقدار null را برمی‌گرداند چون سرویس ما از نوع Started هست نه Bound.

خب! فعلا کاری با کلاس سرویس نداریم. همانطور که Activity ها برای آنکه توسط اندروید شناسایی شوند باید داخل مانیفست پروژه تعریف شده باشند، Service هم از این قاعده مستثنی نیست. داخل تگ application فایل AndroidManifest.xml تگ زیر را اضافه می‌کنم:

```
<service android:name=".MyService" />
```

نام کلاس سرویس به عنوان مقدار name قرار می‌گیرد.

AndroidManifest.xml

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

    <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=".ActivityTwo"></activity>
        <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>
```

در صورت عدم تعریف سرویس در مانیفست، امکان استفاده از آن میسر نخواهد بود.

حالا نوبت به تکمیل کلاس اکتیویتی می‌رسد:

MainActivity.java

```
package ir.android_studio.startservice;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {

    private Button startButton, stopButton, activityBtn;
    private Intent servIntent;

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

        startButton = findViewById(R.id.start_btn);
        stopButton = findViewById(R.id.stop_btn);
        activityBtn = findViewById(R.id.two_btn);

        servIntent = new Intent(this, MyService.class);

        startButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                startService(servIntent);

            }
        });

        stopButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                stopService(servIntent);

            }
        });

        activityBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                startActivity(new Intent(MainActivity.this, ActivityTwo.class));

            }
```

```
        });
    }
}
```

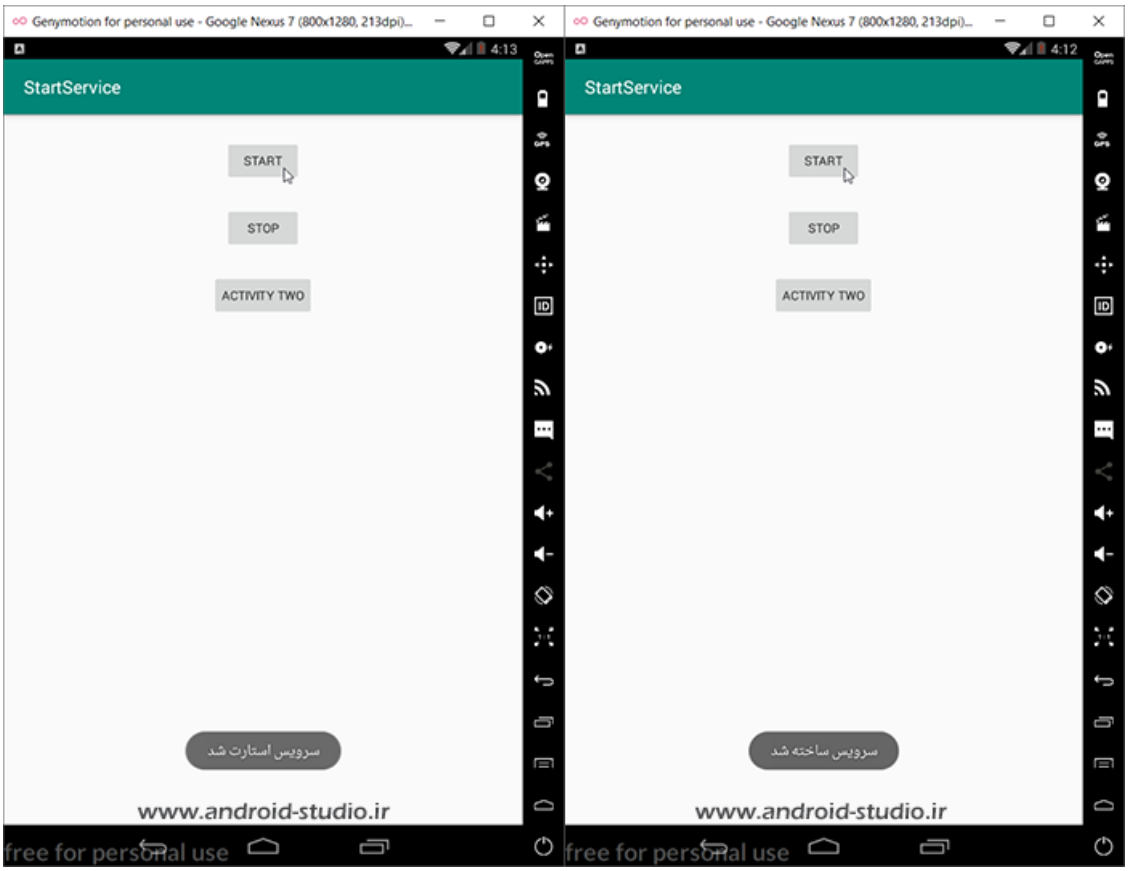
ابتدا سه دکمه شروع، توقف و انتقال به اکتیویتی دوم را داخل اکتیویتی تعریف کردم. برای فراخوانی Service توسط ()startService و ()stopService از intent استفاده می‌کنیم که قبلا در مبحث آموزش کار با intent در اندروید با این قابلیت آشنا شدیم.

یک نمونه از Intent با نام servIntent تعریف کردم. ورودی اول، کانتکست باید وارد شود که this تعریف شده؛ یعنی همین کلاس فعلی. پارامتر دوم هم کلاس سرویس تعریف شده است.

در ادامه برای هریک از دکمه‌ها یک setOnClickListener تعریف شده. داخل onClick دکمه استارت، ()startService با ورودی servIntent تعریف شده که با کلیک روی این دکمه ابتدا متد onCreate و سپس onStartCommand سرویس فراخوانی خواهد شد. در onClick دکمه توقف هم ()stopService با ورودی servIntent تعریف شده که با کلیک روی آن، متد onDestroy کلاس سرویس فراخوانی خواهد شد. برای دکمه سوم هم یک intent دیگر تعریف کردم که از MainActivity به یک اکتیویتی دیگر منتقل شود. یک اکتیویتی با نام ActivityTwo به پروژه اضافه کرده‌ام. توجه داشته باشید ()onCreate فقط مرتبه اول اجرا می‌شود و در دفعات بعد که دستور استارت سرویس صادر می‌شود فقط onStartCommand اجرا خواهد شد، تا زمانی که سرویس متوقف نشود.

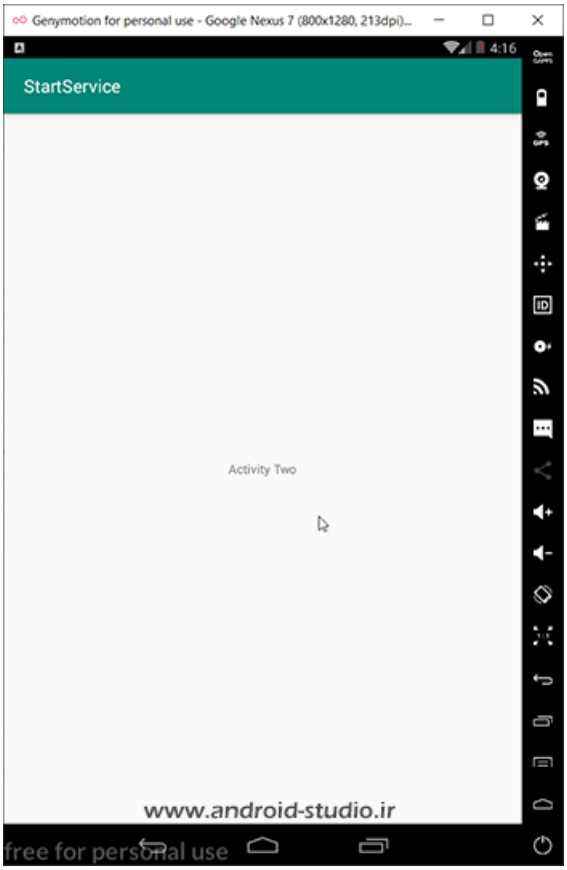
خب! حالا پروژه را روی یک دیوایس پایینتر از اندروید Oero (API 26) اجرا می‌کنم:

روی دکمه Start کلیک می‌کنم:



به ترتیب ابتدا Toast "سرویس ساخته شد" و سپس "سرویس استارت شد" اجرا شدند که نشانگر اجرای متدهای onCreate و onStartCommand سرویس است. موزیک نیز در حال پخش است. اگر مجدد روی دکمه استارت کلیک کنید فقط پیغام "سرویس استارت شد" را مشاهده خواهید کرد که نشان می‌دهد در مراتب بعد فقط onStartCommand اجرا می‌شود.

روی دکمه سوم کلیک می‌کنم:



با خروج از اکتیویتی اصلی همچنان فایل صوتی در حال اجراست و متوقف نشد. در نهایت از برنامه خارج می‌شوم:

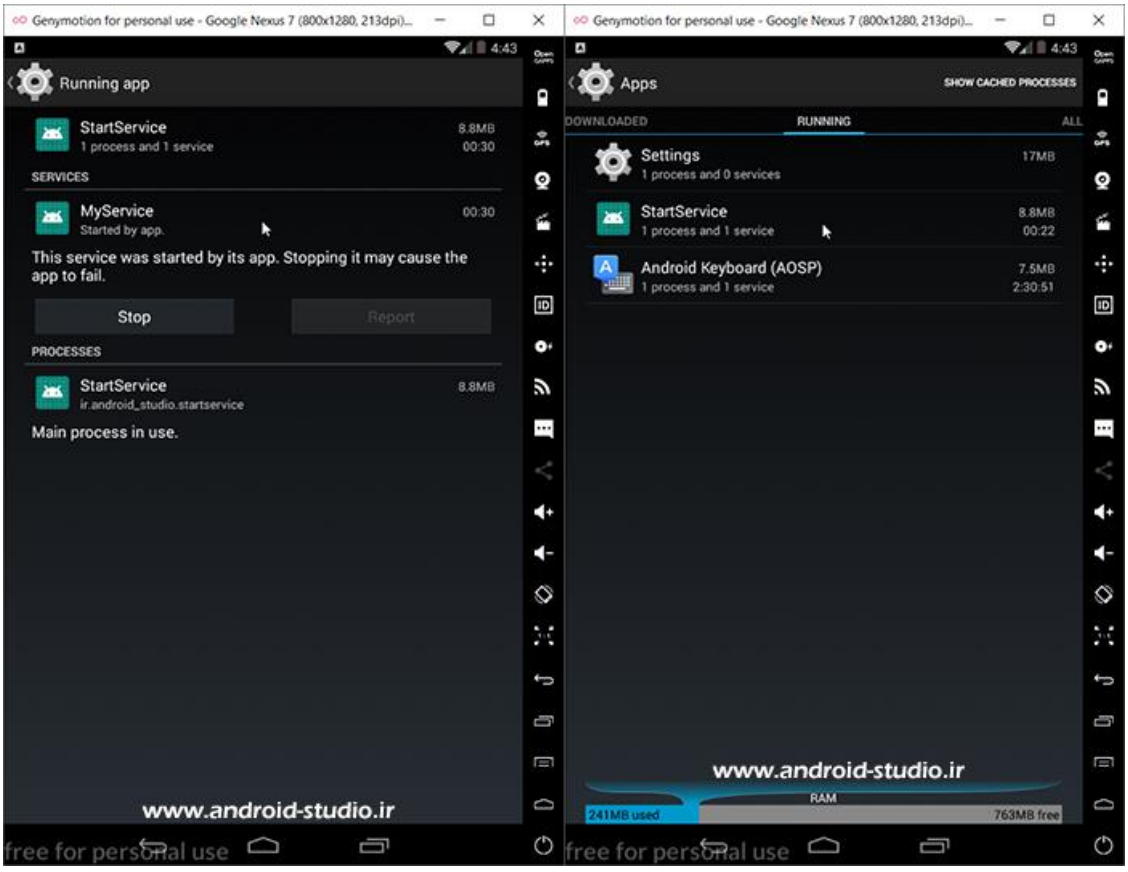
باز هم اجرای فایل صوتی متوقف نشد.

با کلیک روی دکمه Stop متد onDestroy اجرا شده و پخش موزیک متوقف می‌شود:

در قسمت تنظیمات اندروید لیست سرویس‌های در حال اجرای برنامه‌ها را می‌توانیم ببینیم. در نسخه‌های قدیمی اندروید (اندروید ۵ و قبل از آن) با رفتن به مسیر:

Settings > Apps > RUNNING

به این لیست دسترسی داریم:



در لیست RUNNING نام برنامه ما مشاهده می‌شود. به جزئیات دقت کنید. یک سرویس به مدت ۲۲ ثانیه در حال اجراست. روی آن کلیک می‌کنم. در صفحه جدید نام سرویس یعنی MyService نیز مشاهده می‌شود. سرویس را از اینجا هم می‌توانیم متوقف کنیم. در صورت توقف سرویس چه از اینجا چه کلیک روی دکمه Stop درون اکتیویتی اصلی، سرویس متوقف شده و از این لیست حذف می‌شود.

تا اینجا ما یک Background Service را تست کردیم. مشکل این نوع اجرای سرویس در این است که در مواقع کمبود منابع (مانند Memory) اندروید سرویس در حال اجرا را متوقف می‌کند. بنابراین ممکن است قبل از آنکه عملیات مدنظر به پایان برسد، توسط سیستم ناتمام بماند. یعنی کدی که تا اینجا روی شبیه ساز اجرا کردیم ممکن است بعد از گذشت مثلا ۴۰ ثانیه یا دو دقیقه سرویس متوقف شود (بخصوص در اندرویدهای جدیدتر). برای حل این مسئله باید سرویس را به حالت Foreground (پیش زمینه) ببریم. سیستم عامل اندروید به Foreground Service ها اولویت بالاتری نسبت به Background Service اختصاص داده و باعث می‌شود تا در مواقع کمبود منابع هم سرویس به کار خود ادامه داده و توسط سیستم متوقف نشود. درست همان اولویتی که به کامپونتی مانند Activity می‌دهد. اندروید یک اکتیویتی را به دلیل کمبود منابع متوقف نمی‌کند.

برای اجرای Service در اندروید در حالت Foreground باید یک نوتیفیکیشن نیز در Status Bar اندروید نمایش داده شود تا کاربر در جریان اجرای سرویس قرار گیرد. این Notification تا زمانی که سرویس

متوقف نشده قابل حذف نیست. برای تبدیل یک Background Service به Foreground Service لازم است متد ()startForeground درون خود سرویس فراخوانی شود.

متد ()startForeground دو ورودی می‌گیرد. اولی یک id (شناسه) از جنس int و دومی یک Notification. مطابق مبحث آموزش کار با Notification در اندروید یک نوتیفیکیشن در سرویس تعریف می‌کنم:

MyService.java

```
package ir.android_studio.startservice;

import android.app.Notification;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.app.Service;
import android.content.Intent;
import android.media.MediaPlayer;
import android.os.Build;
import android.os.IBinder;
import android.widget.Toast;

import androidx.annotation.Nullable;
import androidx.core.app.NotificationCompat;

public class MyService extends Service {

    private MediaPlayer soundPlayer;
    private String CHANNEL_ID = "channelId";
    private NotificationManager notifManager;

    @Override
    public void onCreate() {

        Toast.makeText(this, "سرویس ساخته شد", Toast.LENGTH_SHORT).show();
        soundPlayer = MediaPlayer.create(this, R.raw.song);
        soundPlayer.setLooping(false);

    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {

        Toast.makeText(this, "سرویس استارت شد", Toast.LENGTH_SHORT).show();

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {

            String offerChannelName = "Service Channel";
            String offerChannelDescription= "Music Channel";
            int offerChannelImportance = NotificationManager.IMPORTANCE_DEFAULT;

            NotificationChannel notifChannel = new NotificationChannel(CHANNEL_ID,
```

```
offerChannelName, offerChannelImportance);
            notifChannel.setDescription(offerChannelDescription);
            notifManager = getSystemService(NotificationManager.class);
            notifManager.createNotificationChannel(notifChannel);

        }

        NotificationCompat.Builder sNotifBuilder = new NotificationCompat.Builder(this,
CHANNEL_ID)
                .setSmallIcon(R.drawable.music)
                .setContentTitle("موسیقی")
                .setContentText("یک موسیقی در حال اجراست");

        Notification servNotification = sNotifBuilder.build();

        startForeground(1, servNotification);

        soundPlayer.start();
        return START_STICKY;

    }

    @Override
    public void onDestroy() {

        Toast.makeText(this, "سرویس متوقف شد", Toast.LENGTH_SHORT).show();
        soundPlayer.stop();

    }

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

نکته: متد startForeground() نوتیفیکیشن از نوع Notification را می‌پذیرد نه NotificationCompat.Builder.

در کد فوق ملاحظه می‌کنید ابتدا یک Notification Channel ایجاد کردم. همانطور که قبلا در مبحث نوتیفیکیشن‌ها آشنا شدید، Channel برای اندروید ۸ به بالا ضروری است بنابراین شرطی تعریف شده که کد فقط در صورتی اجرا شود که برنامه روی اندروید O و به بالا در حال اجراست. سپس یک Notification تعریف شده. بعد از تعریف نوتیفیکیشن متد startForeground را نوشتم. برای ورودی اول یک عدد دلخواه از جنس int و ورودی دوم نوتیفیکیشنی که قبلا ساختیم را معرفی می‌کنیم.

نکته: شناسه (id) تعریف شده برای ورودی نخست باید عددی غیر از صفر باشد.

مجدد پروژه را روی دیوایس اجرا می‌کنم:

با کلیک روی دکمه Start سرویس اجرا شده و نوتیفیکیشن نیز در استاتوس بار اندروید ظاهر می‌گردد. این سرویس تا زمانی که stopService از سمت کامپوننت (یعنی اکتیویتی ما و به واسطه دکمه Stop) به سرویس ارسال نشود متوقف نخواهد شد.

تغییرات Service در Android O (اندروید ۸)

در اندروید ۸ شاهد تغییراتی در فراخوانی و اجرای Service ها هستیم. مورد نخست اینکه بجای ()startService باید توسط ()startForegroundService سرویس را فراخوانی کنیم. مورد دوم اینکه پس از فراخوانی سرویس تنها ۵ ثانیه مهلت داریم تا متد ()startForeground را اجرا کنیم و اگر در طول این مدت متد اجرا نشود، سرویس توسط سیستم متوقف خواهد شد.

پس تغییری که لازم است انجام دهیم این است که بررسی کنیم اگر برنامه روی اندروید ۸ و به بالا درحال اجراست سرویس توسط ()startForegroundService فراخوانی شود:

```
startButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {

            startForegroundService(servIntent);

        } else {

            startService(servIntent);
        }

    }
});
```

اما نیاز نیست مدیریت این کار را خومان انجام دهیم! کلاس ContextCompat اندروید این مدیریت را برای ما انجام می‌دهد و نیازی به نوشتن کدهای اضافی نیست:

```
startButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        ContextCompat.startForegroundService(MainActivity.this, servIntent);

    }
});
```

با اضافه کردن ContextCompat به ابتدای startForegroundService() ورژن اندروید بررسی شده و متد مناسب آن فراخوانی می‌شود. برای اینکه ببینید این کد چه کاری انجام می‌دهد درحالی که نشانگر موس را روی startForegroundService قرار داده‌اید، کلیدهای ترکیبی ctrl + B را بزنید:

activity_main.xml | MainActivity.java | ContextCompat.java | MyService.java | AndroidManifest.xml

```
661     }
662
663     /**
664      * startForegroundService() was introduced in O, just call startService
665      * for before O.
666      *
667      * @param context Context to start Service from.
668      * @param intent The description of the Service to start.
669      *
670      * @see Context#startForegroundService(Intent)
671      * @see Context#startService(Intent)
672      */
673 @   public static void startForegroundService(@NonNull Context context, @NonNull Intent intent) {
674         if (Build.VERSION.SDK_INT >= 26) {
675             context.startForegroundService(intent);
676         } else {
677             // Pre-O behavior.
678             context.startService(intent);
679         }
680     }
```

در کلاس ContextCompat یک تابع با نام startForegroundService تعریف شده و کار آن بررسی شرطی است که ما قبلا تعریف کردیم. بنابراین بررسی این شرط را به عهده این کلاس گذاشته و از نوشتن کدهای اضافی صرف نظر می‌کنیم.

تغییرات Service در Android P (اندروید ۹)

هرچه از عمر سیستم عامل محبوب اندروید می‌گذرد تدابیر امنیتی بیشتری برای حفظ امنیت و حریم شخصی کاربران لحاظ می‌شود. برای سازگاری Foreground Service در اندروید ۹ باید یک Permission یا مجوز دسترسی به سرویس را در مانیفست پروژه تعریف کنیم. در مباحث قبل مانند آموزش کار با کتابخانه Retrofit و همچنین آموزش کار با دوربین توسط Camera2 API با مجوزها آشنا شدیم. برای اعطای مجوز Foreground Service به برنامه، تگ uses-permission با مقدار android.permission.FOREGROUND_SERVICE را به مانیفست پروژه اضافه کردم:

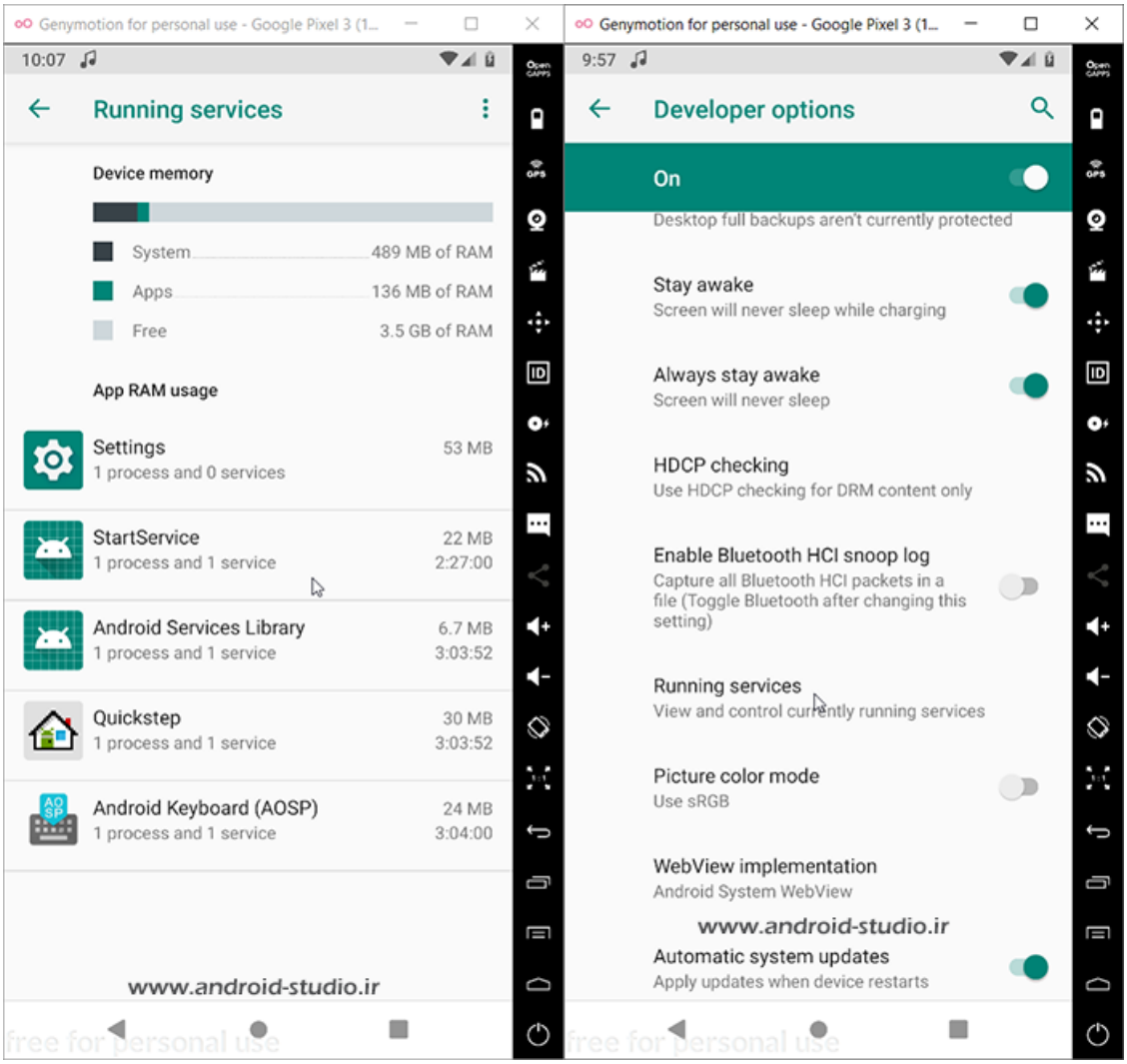
AndroidManifest.xml

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

    <uses-permission android:name="android.permission.FOREGROUND_SERVICE" />

    <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=".ActivityTwo"></activity>
        <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>
```

حالا پروژه را روی یک دیوایس با اندروید ۹ هم اجرا و تست می‌کنم:

با کلیک روی دکمه Start نوتیفیکیشن ظاهر شده و موزیک play می‌شود. بنابراین Foreground Service ما در همه نسخه‌های اندروید فعلی سازگار است.

برای مشاهده سرویس‌های در حال اجرا در اندروید ۶ و به بالا این گزینه در قسمت Developer Options قرار گرفته:



در تصویر فوق مشاهده می‌کنید سرویس به مدت بیشتر از دو ساعت در حال اجرا بوده و توسط سیستم متوقف نشده تا زمانی که دستور توقف توسط ()stopService به سرویس ارسال شود.

نکته: اگر بخواهیم سرویس پس از اتمام عملیات و بصورت خودکار متوقف شود باید متد ()stopSelf درون سرویس فراخوانی شود. اگر بخواهیم Foreground Service به Background Service تبدیل شود باید متد ()stopForeground درون سرویس فراخوانی شود. این متد فقط نوتیفیکیشن را حذف می‌کند و برای توقف سرویس باید بعد از آن ()stopSelf نیز فراخوانی شود.

نکته: Service را با Multi Threading (پردازش چند نخی) اشتباه نگیرید. در آموزش کار با Camera2 API تا حدودی با مفهوم Thread و کاربرد آن آشنا شدیم. از Thread برای پردازش عملیاتی استفاده می‌شود که صرفا هنگام تعامل کاربر با برنامه در حال اجرا هستند و با خروج از برنامه، آن عملیات نیز متوقف می‌شود (مانند ارتباط با دوربین). همچنین خروجی پردازش موجود در Thread های غیر از

Main Thread (UI Thread) باید به Main Thread منتقل شوند. ولی Service برای پردازش عملیاتی استفاده می‌شود که ارتباطی با رابط کاربری (UI) نداشته و هنگامی که برنامه باز نیست هم باید در حال اجرا باشد. ضمن اینکه خود Service در Main Thread اجرا می‌شود و چنانچه پردازش سنگینی در سرویس انجام می‌شود باید آنرا به Thread ای غیر از Main Thread انتقال داد تا این عملیات باعث وقفه در تعامل کاربر با رابط کاربری نشود. یکی از روش‌های پردازش سرویس در Thread جداگانه که ویژه سرویس نیز تهیه شده استفاده از IntentService است. به امید خدا در آینده مفصل به مبحث Multi Threading و Handler ها خواهم پرداخت.

موفق و پیروز باشید.

مطالعه بیشتر:

https://android-developers.googleblog.com/2010/02/service-api-changes-starting-with.html

https://developer.android.com/guide/components/services

https://developer.android.com/reference/android/app/Service.html

https://developer.android.com/guide/components/bound-services

توجه: سورس پروژه درون پوشه Exercises قرار دارد

با ارائه انتقادات و پیشنهادات خود، ما را در ارائه آموزش‌های بهتر یاری فرمائید.

www.android-studio.ir