آموزش برنامه نویسی اندروید
در محیط اندروید استودیو

استفاده از فونت دلخواه در اندروید

مدرس : سید مهدی مطهری

www.android-studio.ir

به نام خدا

یکی از نیازهای ضروری یک اپلیکیشن، بخصوص اپلیکیشن‌های فارسی زبان، امکان استفاده از فونت‌های دلخواه در برنامه است. پیاده سازی فونت در اندروید به روش‌های مختلفی قابل انجام است که در این مبحث سه روش را بررسی می‌کنیم.

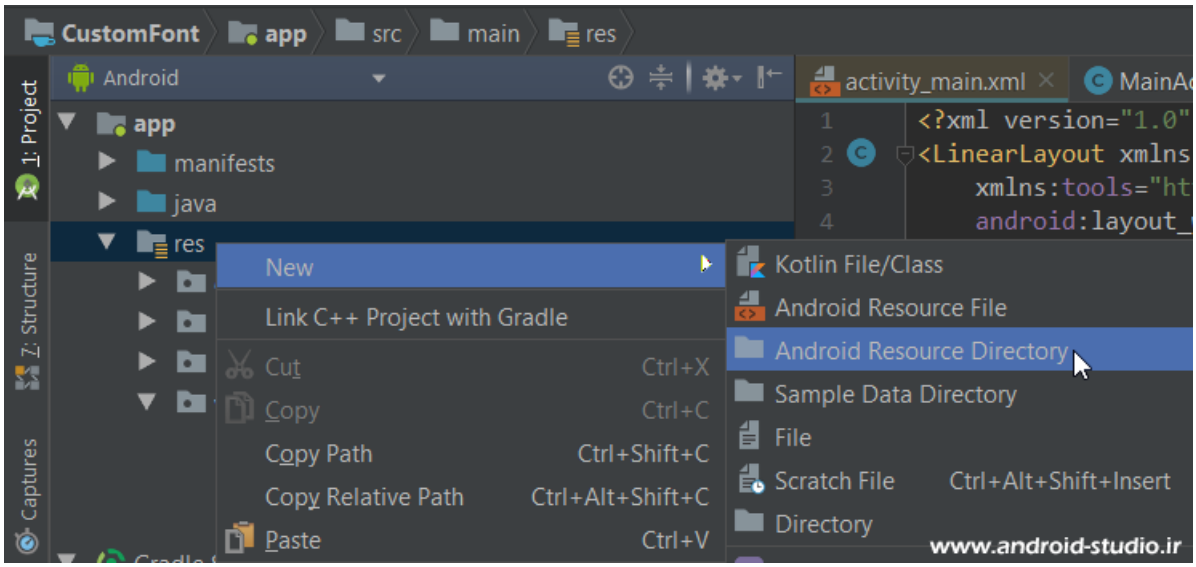
ابتدا یک پروژه جدید با نام CustomFont و یک Empty Activity ایجاد می‌کنم.

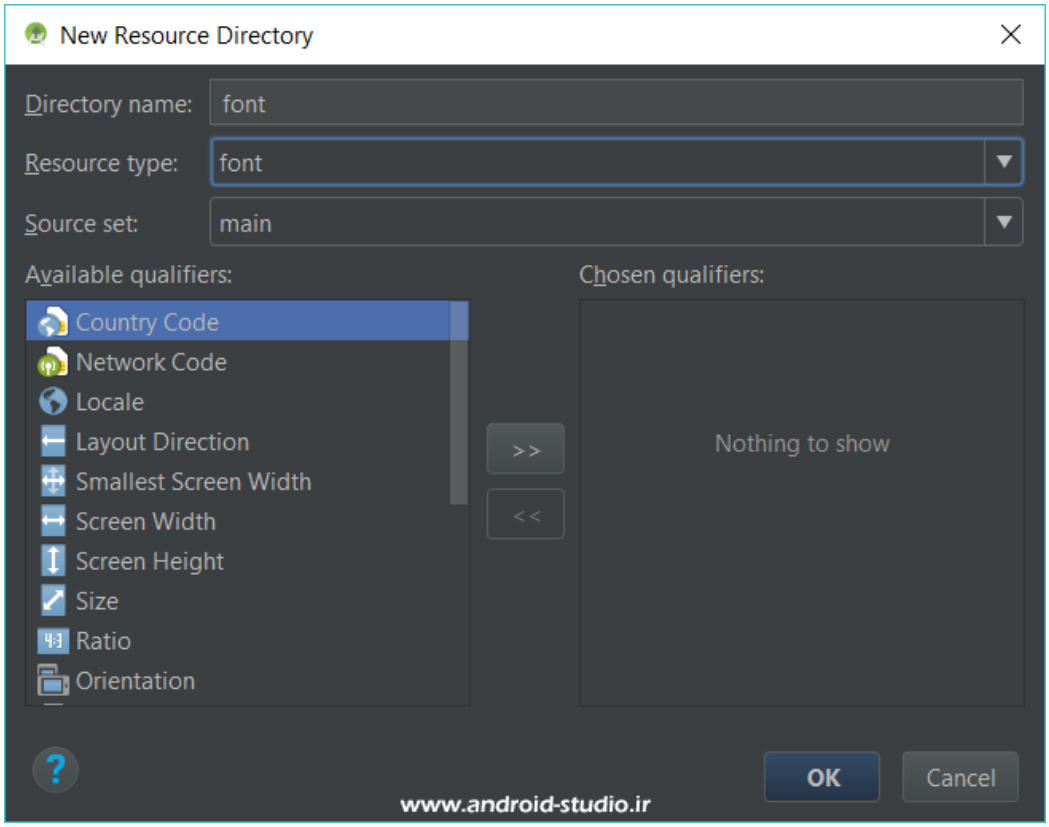
تغییر فونت Widget ها در XML:

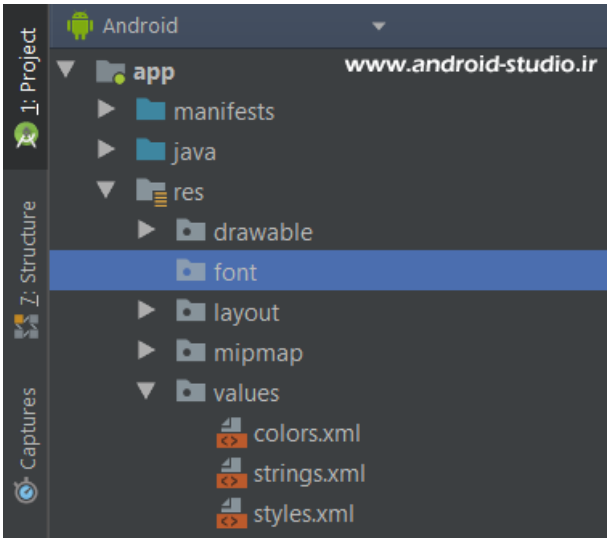
همزمان با معرفی API 26 قابلیت جدیدی در اندروید استودیو نسخه ۳ اضافه شد که امکان تعیین فونت برای widget ها بدون نیاز به استفاده از متدهای جاوا و تنها با افزودن خاصیت fontFamily به هر ویجت را فراهم می‌کند. این قابلیت از API 16 به بالا پشتیبانی می‌کند که لازم است کتابخانه appcompat-v7 حتما در پروژه وجود داشته باشد.

ابتدا می‌بایست یک دایرکتوری با نام font به res اضافه کنیم:

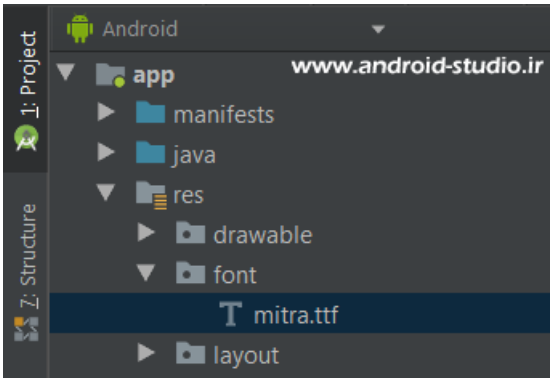
Res > New > Android Resource Directory







حالا فایل فونت یا فونت‌های مدنظر (با پسوند ttf. یا otf.) را درون این دایرکتوری قرار می‌دهم:

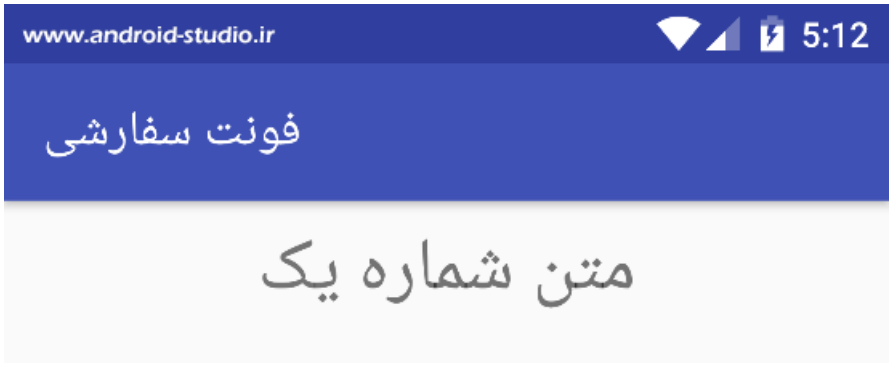


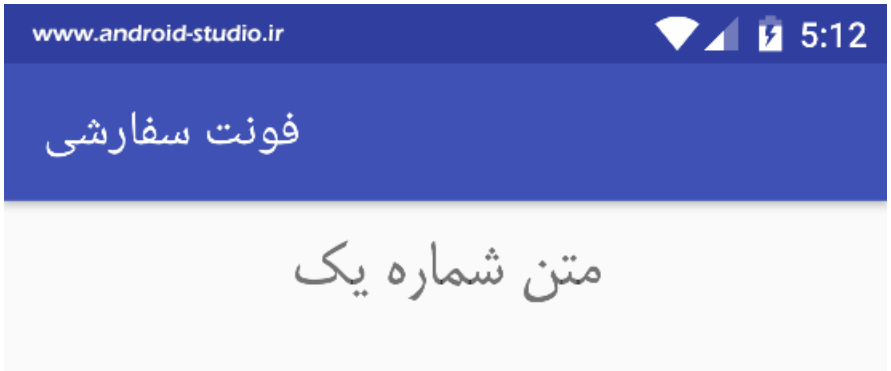
تذکر: در نامگذاری فایل فونت فقط از حروف کوچک، اعداد و زیرخط (underline) استفاده کنید.

در نهایت توسط خاصیت fontFamily فونت را به یک TextView اضافه می‌کنم. خروجی را قبل و بعد از تعریف این خاصیت مشاهده می‌کنید:

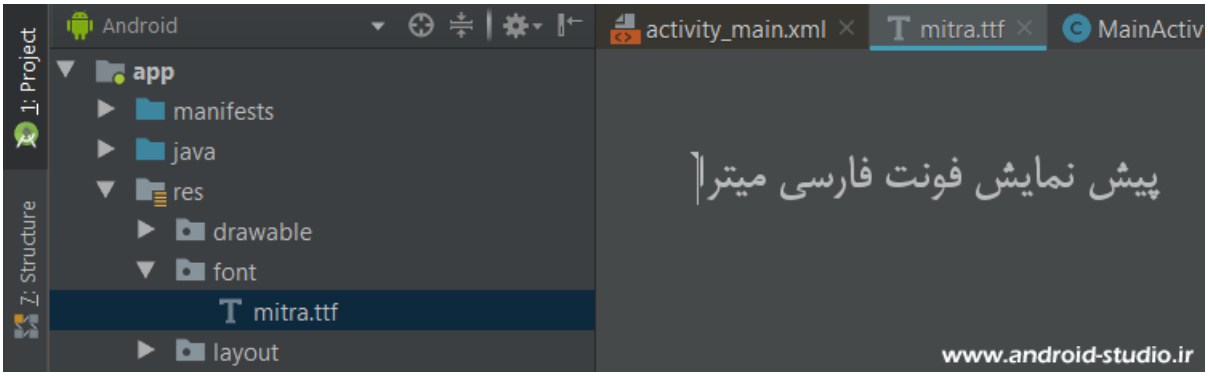
:activity_main.xml

```
<?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"
    android:orientation="vertical"
    tools:context=".MainActivity"
    android:padding="8dp">


    <TextView
        android:id="@+id/txt_one"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="متن شماره یک"
        android:textAlignment="center"
        android:fontFamily="@font/mitra"
        android:textSize="30sp"/>


</LinearLayout>
```





امکانی جهت پیش نمایش فونت در محیط اندروید استودیو فراهم گردیده که با دابل کلیک روی فونت، ادیتور باز شده و می‌توانید متن دلخواه را تایپ کنید:



ست کردن فونت توسط Typeface:

در این قسمت می‌خواهم در اکتیویتی و توسط کلاس Typeface فونت مدنظرم را به id ویجت مربوطه ارسال کنم. یک TextView با شناسه txt_two به صفحه اضافه می‌کنم:

```
<TextView
    android:id="@+id/txt_two"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="متن شماره دو"
    android:textAlignment="center"
    android:textSize="30sp"/>
```

در ادامه اکتیویتی را به اینصورت تکمیل می‌کنم:

:MainActivity.java

```
package ir.android_studio.customfont;

import android.graphics.Typeface;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    TextView txtTwo;

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

        txtTwo = findViewById(R.id.txt_two);
        Typeface tf = getResources().getFont(R.font.afaaq);
        txtTwo.setTypeface(tf);

    }
}
```

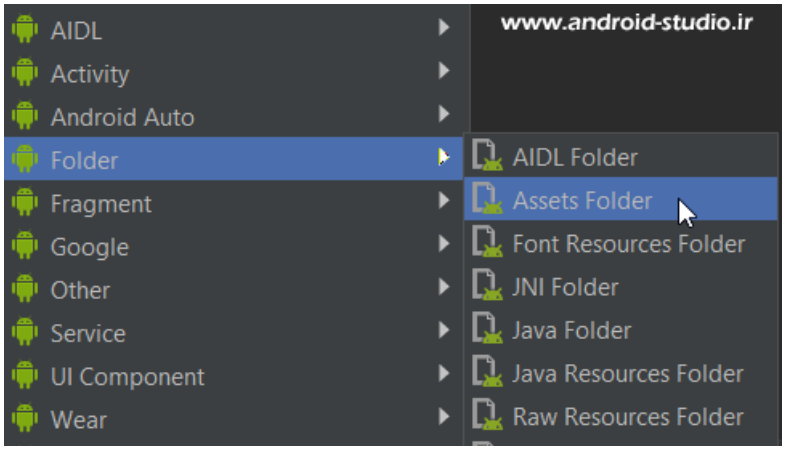
در خط اول TextView در اکتیویتی تعریف شده. در خط دوم یک نمونه از Typeface ساخته شده که توسط متد ()getFont فونت afaaq از دایرکتوری font (یعنی R.font) فراخوانی شده است. (پوشه font در دایرکتوری res قرار گرفته بنابراین قبل از ()getFont متد ()getResources نوشته شده. در نهایت توسط متد ()setTypeface فونتی که در tf تعریف شده را روی txtTwo اعمال می‌کنم:

در قسمت متد getFont یک ارور داریم:

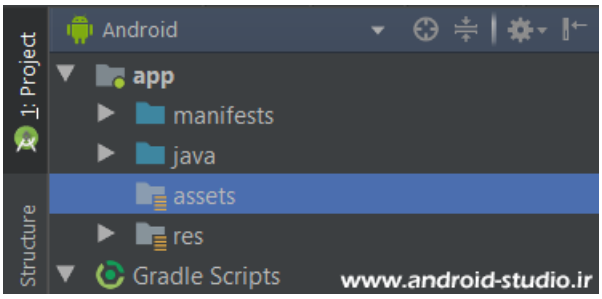
این متد در API 26 و به بالا قابل استفاده است. یعنی اگر پروژه را روی یک دیوایس پایینتر از API 26 اجرا کنم، تغییر فونت اعمال نمی‌شود و با توجه به اینکه MinSDK پروژه را روی ۱۷ قرار داده‌ام، این مسئله برای تعداد زیادی از کاربران ایجاد مشکل کرده و بجای فونت مدنظر من، فونت پیش فرض دیوایس خود را مشاهده خواهند کرد. استفاده از این متد در زمانی مناسب است که MinSDK پروژه حداقل ۲۶ باشد.

پس ناچارا لازم است Typeface را به صورت دیگری پیاده سازی کنیم. از متدی با نام createFromAsset استفاده می‌کنم که البته فایل فونت را از فولدر assets می‌گیرد. من این فولدر را در پروژه ندارم بنابراین ابتدا آنرا ایجاد می‌کنم:

app > New > Folder > Assets Folder

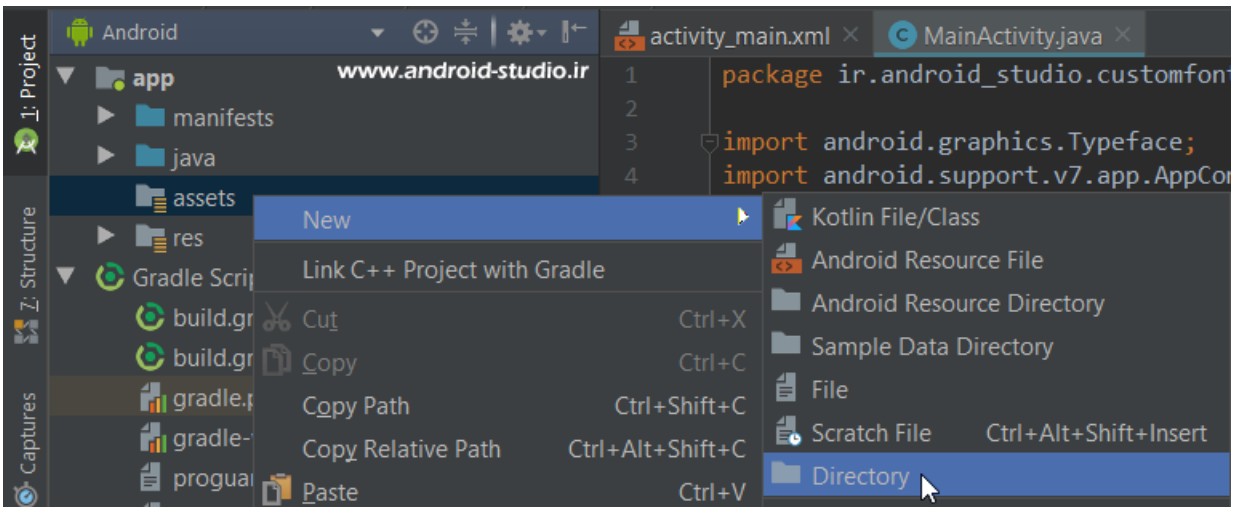


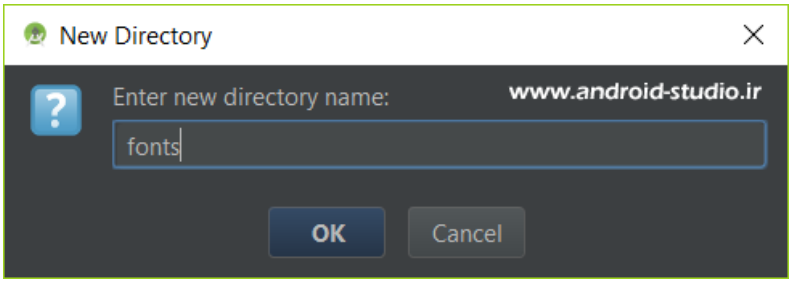
در پنجره ی باز شده با انتخاب گزینه Finish فولدر اضافه می‌شود:

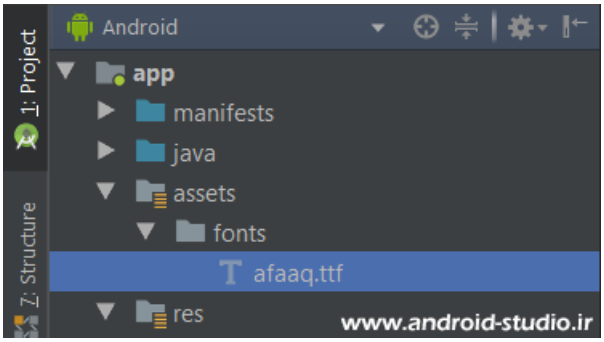


حالا باید فونت را مجدد به این فولدر هم اضافه کنم. اینکه فایل فونت مستقیما درون assets قرار گیرد یا داخل یک فولدر ویژه فونت، تفاوتی نمی‌کند اما من برای مرتب بودن پروژه، یک فولدر با نام fonts به assets اضافه کرده و همان فونت قبل یعنی afaaq.ttf را درون آن قرار می‌دهم:

assets > New > Directory







(فونت را از فولدر font قبلی copy و اینجا paste کردم)

```
package ir.android_studio.customfont;

import android.graphics.Typeface;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    TextView txtTwo, txtThree;

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

        // getFont Method
        txtTwo = findViewById(R.id.txt_two);
        Typeface tf = getResources().getFont(R.font.afaaq);
        txtTwo.setTypeface(tf);

        // createFromAsset Method
        txtThree = findViewById(R.id.txt_three);
        Typeface atf = Typeface.createFromAsset(getAssets(), "fonts/afaaq.ttf");
        txtThree.setTypeface(atf);

    }
}
```

```
<TextView
    android:id="@+id/txt_three"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="متن شماره سه"
    android:textAlignment="center"
    android:textSize="30sp"/>
```

ملاحظه می‌کنید TextView سوم با شناسه txt_three تعریف و توسط متد createFromAsset() فونت موجود در مسیر fonts/afaaq.ttf دایرکتوری assets روی آن اعمال شد:

تغییر فونت در کل برنامه توسط کتابخانه Calligraphy:

در روش‌هایی که تاکنون بررسی شد، برای هر ویجت باید فونت را جداگانه تعریف می‌کردیم. برای اپلیکیشنی که تعداد کمی ویجت دارد یا قصد اعمال فونت فقط در تعداد محدودی از ویجت‌ها را داریم، این روش‌ها ساده و مناسب خواهد بود. اما در مواردی که در حال توسعه یک اپلیکیشن با اکتیویتی‌ها و ویجت‌های متعدد هستیم و لازم است فونت روی همه آنها اعمال شود، استفاده از روش‌های فوق منطقی نیست و علاوه بر صرف زمان زیاد، احتمال خطا را نیز افزایش می‌دهد. در اینصورت لازم است راهی را انتخاب کنیم که بتوان یک فونت را در سراسر پروژه تعریف کرد بدون آنکه نیاز به اعمال تغییرات روی تک تک ویجت‌ها (مانند TextView، Toolbar، Menu و...) باشیم.

در حال حاضر متداول‌ترین راه برای این کار، استفاده از کتابخانه‌ی Calligraphy است که توسط شخصی با نام Christopher Jenkins تهیه و در گیت هاب و به صورت رایگان در دسترس توسعه دهندگان قرار گرفته است:

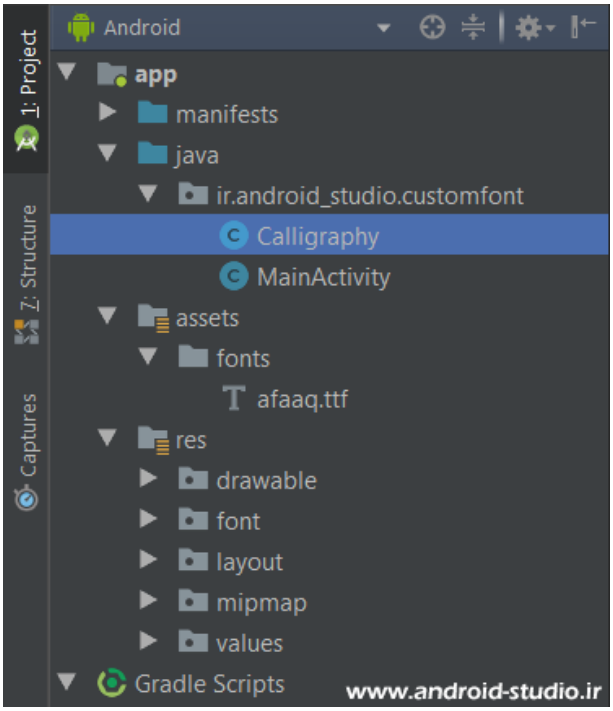
https://github.com/chrisjenx/Calligraphy

تذکر: در حدود دو ماه قبل از نگارش این آموزش، توسعه دهنده این کتابخانه نسخه جدید کتابخانه را در صفحه جدیدی در گیت هاب با عنوان Calligraphy 3 معرفی و اعلام کرده نسخه فعلی بروزرسانی نخواهد شد. با اینحال من از همین نسخه رایج فعلی یعنی 2.3.0 استفاده می‌کنم. نسخه جدید تفاوت محسوسی در پیاده سازی و استفاده نداشته و با مطالعه این مبحث، از نسخه ۳ نیز می‌توانید استفاده کنید.

مطابق توضیحات موجود در صفحه گیت هاب، کتابخانه را به بلاک dependencies در build.gradle(app) اضافه و پروژه را سینک می‌کنم تا کتابخانه دانلود شود:

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:27.1.1'
    compile 'uk.co.chrisjenx:calligraphy:2.3.0'
}
```

در قدم بعد لازم است یک کلاس به پکیج اصلی پروژه اضافه کرده و سپس آنرا از کلاس Application ارث بری کنیم. یک کلاس با نام دلخواه Calligraphy ایجاد کردم:

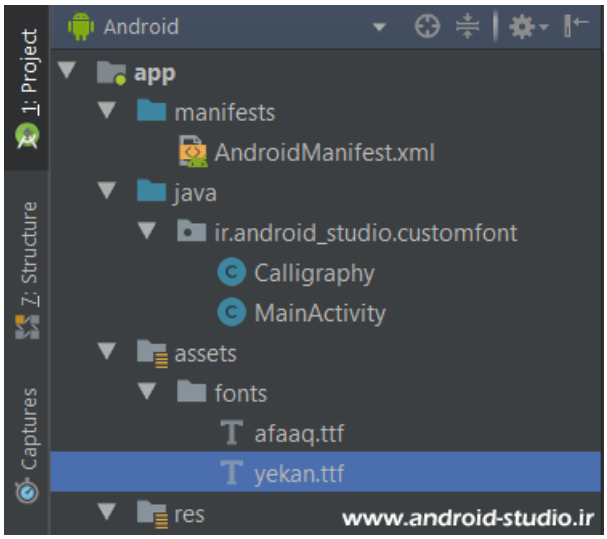



```
package ir.android_studio.customfont;

import android.app.Application;

public class Calligraphy extends Application {

}
```

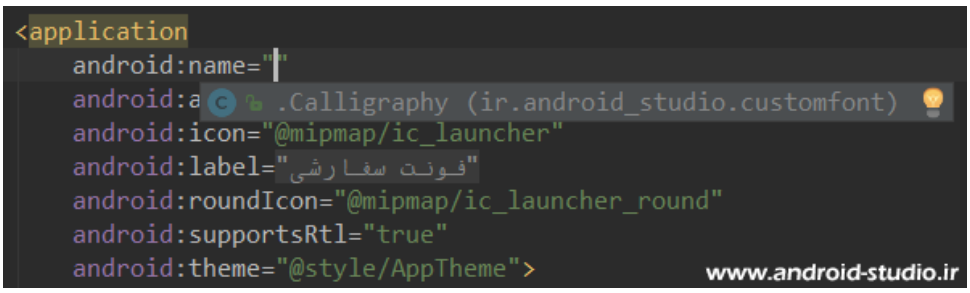
سپس یک متد ()onCreate به صورت زیر به کلاس اضافه می‌کنم:

```
package ir.android_studio.customfont;

import android.app.Application;

import uk.co.chrisjenx.calligraphy.CalligraphyConfig;

public class Calligraphy extends Application {

    @Override
    public void onCreate() {
        super.onCreate();
        CalligraphyConfig.initDefault(new CalligraphyConfig.Builder()
                .setDefaultFontPath("fonts/yekan.ttf")
                .setFontAttrId(R.attr.fontPath)
                .build()
        );

    }

}
```

کد متد را می‌توانید از سورس پروژه یا صفحه گیت هاب کتابخانه کپی کرده و فقط مسیر فونت مدنظر خود را در ورودی setDefaultFontPath() اصلاح کنید. به جهت تشخیص بهتر، از یک فونت متفاوت با دو فونت قبل استفاده می‌کنم. این کتابخانه نیز فونت را از assets می‌خواند بنابراین فونت yekan.ttf را مانند مرحله قبل به assets/fonts اضافه کردم:



لازم است این کلاس را در مانیفست پروژه (AndroidManifest.xml) توسط خاصیت android:name معرفی کنیم:



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

    <application
        android:name=".Calligraphy"
        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=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

حالا کافیست متد زیر را در اکتیویتی یا اکتیویتی‌هایی که قصد داریم از این فونت تاثیر بپذیرند اضافه کنیم:

```
@Override
protected void attachBaseContext(Context newBase) {
    super.attachBaseContext(CalligraphyContextWrapper.wrap(newBase));
}
```

:MainActivity.java

```
package ir.android_studio.customfont;

import android.content.Context;
import android.graphics.Typeface;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.Toolbar;
import android.widget.TextView;

import uk.co.chrisjenx.calligraphy.CalligraphyContextWrapper;

public class MainActivity extends AppCompatActivity {

    TextView txtTwo, txtThree;
    Toolbar mToolbar;

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


        //Toolbar
        mToolbar = findViewById(R.id.m_toolbar);
        setSupportActionBar(mToolbar);

        //getFont Method
        txtTwo = findViewById(R.id.txt_two);
        Typeface tf = getResources().getFont(R.font.afaaq);
        txtTwo.setTypeface(tf);

        //createFromAsset Method
        txtThree = findViewById(R.id.txt_three);
        Typeface atf = Typeface.createFromAsset(getAssets(), "fonts/afaaq.ttf");
        txtThree.setTypeface(atf);

    }

    @Override
    protected void attachBaseContext(Context newBase) {
        super.attachBaseContext(CalligraphyContextWrapper.wrap(newBase));
    }

}
```

من یک Toolbar جایگزین اکشن‌بار پیش فرض کردم تا نتیجه را بهتر درک کنید.

:activity_main.xml

```
<?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"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:orientation="vertical"
    tools:context=".MainActivity">


    <android.support.v7.widget.Toolbar
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:id="@+id/m_toolbar"
        android:background="?attr/colorPrimary"
        android:elevation="3dp"
        app:titleTextColor="#fff"
        app:popupTheme="@style/ThemeOverlay.AppCompat.Dark" />

    <TextView
        android:id="@+id/txt_one"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="متن شماره یک"
        android:textAlignment="center"
        android:fontFamily="@font/mitra"
        android:textSize="30sp"/>

    <TextView
        android:id="@+id/txt_two"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="متن شماره دو"
        android:textAlignment="center"
        android:textSize="30sp"/>

    <TextView
        android:id="@+id/txt_three"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="متن شماره سه"
        android:textAlignment="center"
        android:textSize="30sp"/>

    <TextView
        android:id="@+id/txt_four"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="متن شماره چهار"
        android:textAlignment="center"
        android:textSize="30sp"/>

    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:text="متن دکمه" />


</LinearLayout>
```

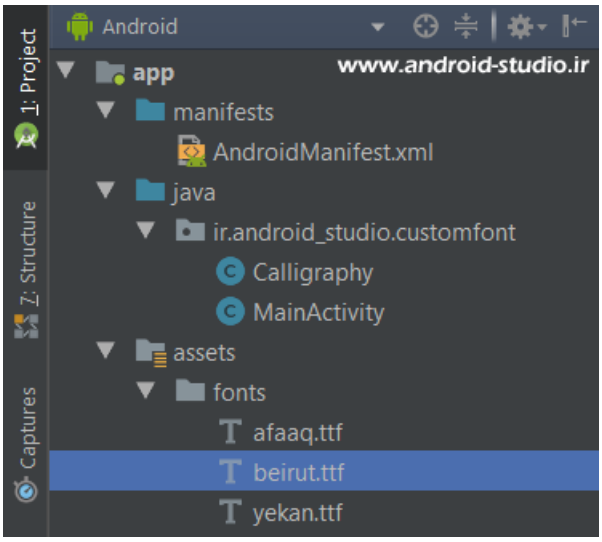
یک TextView دیگر با شناسه txt_four و یک Button در صفحه قرار دادم که نه از طریق xml و نه java فونتی روی آن اعمال نشده.

پروژه را اجرا می‌کنم:

ملاحظه می‌کنید title تولبار، TextView چهارم و Button، فونت yekan که در کالیگرافی تعریف شده را نشان می‌دهند. حتی TextView اول که فونت mitra در xml اعمال شده بود نیز فونت yekan را نمایش می‌دهد. یعنی فونت Calligraphy روی همه آیتم‌های درون اکتیویتی اعمال می‌شود مگر اینکه توسط Typeface فونتی متفاوت از آن تعریف شده باشد (مانند TextView شماره دو و سه).

با استفاده از Calligraphy هم می‌توان مستقیم و بدون نیاز به کدهای جاوا، روی ویجت‌ها و از طریق xml فونت اعمال کرد. اینکار عینا مانند روش xml ابتدای مبحث انجام می‌شود با این تفاوت که به‌جای

خاصیت fontFamily از fontPath استفاده می‌کنیم و فایل فونت نیز باید در assets قرار گرفته باشد. یک فونت دیگر با نام beirut.ttf به فولدر fonts در assets اضافه می‌کنم:



سپس یک دکمه دیگر به صفحه اضافه کرده و فونت را توسط fontPath تعریف می‌کنم:

```
<Button
    android:id="@+id/button_two"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="متن دکمه دوم"
    fontPath="fonts/beirut.ttf"/>
```

دکمه دوم فونت beirut را نشان می‌دهد.

شاید این سوال برایتان پیش آمده که تفاوت این روش و روش اول مبحث که هردو توسط یک خاصیت در xml انجام می‌شوند چیست و ما باید کدام گزینه را انتخاب کنیم؟

یک اصل کلی در توسعه برنامه‌ها (چه در بستر موبایل، وب و یا دسکتاپ) این است که تا حد امکان استفاده از کتابخانه‌ها و پلاگین‌ها به حداقل برسد ترجیحا از امکانات داخلی ای که در اختیار داریم بهره ببریم تا هم حجم برنامه و هم ایرادات، باگ‌ها و تداخلات احتمالی کتابخانه‌ها با سایر اجزای پروژه به حداقل برسد. یعنی در اینجا اگر نیازی به تغییر فونت سراسر پروژه نداشته و تنها چند ویجت محدود باید تغییر کند، بهتر است از کتابخانه Calligraphy صرف نظر نموده و از راهکارهای داخلی اندروید مانند xml و Typeface استفاده کنیم. اما در صورتی که نیاز به تغییر فونت کل پروژه بود، ناگزیر به استفاده از این کتابخانه هستیم و همانطور که در صفحات قبل مشاهده کردید، با اعمال فونت توسط کتابخانه، ویجتی که توسط fontFamily تعریف شده بود در عمل این خاصیت را از دست داده و فونت جدید در واقع روی آن Override می‌شود. بنابراین در این شرایط، برای تغییر یک یا چند ویجت محدود، می‌بایست از خاصیت xml مربوط به خود کتابخانه یعنی fontPath و یا Typeface بهره گرفت.

طبیعتا خاصیت مربوط به فونت را می‌توان در یک style نیز تعریف کرد تا لازم نباشد برای تک تک ویجت‌های مدنظر جداگانه تعریف گردد:

```
<style name="CustomFontStyle">
    <item name="fontPath">fonts/beirut.ttf</item>
</style>
```

با تعریف استایل فوق، هر ویجت و کامپوننتی که استایل CustomFontStyle را بپذیرد فونت نیز اعمال خواهد شد:

```
<Button
    android:id="@+id/button_two"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="متن دکمه دوم"
    style="@style/CustomFontStyle"/>
```

نکته: خاصیت fontPath بدون پیشوند android: یا app: بوده و به همین دلیل اندروید استودیو اخطاری با همین مضمون اعلام می‌کند که جای نگرانی نیست و خللی در کار ایجاد نمی‌کند. با اینحال اگر تمایل دارید این اخطار از محیط IDE شما حذف شود، راهکار آن در صفحه گیت‌هاب کتابخانه ذکر شده است.

تذکر: در انتخاب فونت دقت کنید. هر فونتی مناسب استفاده در اپلیکیشن نیست. برای مثال فونت‌های آفاق و بیروت که من در این مبحث استفاده کردم، بجز در موارد خاص (استفاده‌های فانتزی)، نتیجه مطلوبی نخواهد داشت. در حال حاضر فونت ایران سنس بیشترین استفاده را در بین اپ های فارسی زبان دارد که به دلیل رعایت حق مولف در این پروژه استفاده نکردم. در صورت تمایل به استفاده از این فونت لازم است لایسنس آن را از fontiran.com خریداری کنید.

تذکر: پروژه فوق، با نام CustomFont در پوشه Exercises قرار دارد.

منابع تکمیلی:

https://github.com/chrisjenx/Calligraphy

https://developer.android.com/guide/topics/ui/look-and-feel/fonts-in-xml

https://developer.android.com/reference/android/graphics/Typeface

با ارائه انتقادات و پیشنهادات خود، ما را در ارائه آموزش های بهتر یاری فرمائید.
www.android-studio.ir