آموزش برنامه نویسی اندروید
در محیط اندروید استودیو

نمایش صفحات وب در اکتیویتی توسط WebView

مدرس : سیدمهدی مطهری

www.android-studio.ir

به نام خدا

در این مبحث ابتدا به معرفی WebView در اندروید پرداخته سپس با نحوه فراخوانی و نمایش صفحات وب (HTML) آنلاین و آفلاین آشنا می‌شویم.

WebView چیست؟

WebView یکی دیگر از View (Widget)های کاربردی پلتفرم اندروید است. توسط این View می‌توانیم یک صفحه‌ی وب آنلاین یا آفلاین (لوکال) را درون یک Activity به کاربر نمایش دهیم. درست مانند باز کردن یک وب‌سایت در مرورگری مانند Chrome با این تفاوت که کاربر، صفحه وب را درون برنامه مشاهده می‌کند و به یک مرورگر مستقل منتقل نمی‌شود.

در WebView امکان کنترل محتوای دریافتی از صفحه‌ی وب وجود دارد. به عنوان مثال می‌توانیم تعیین کنیم کدهای JavaScript (جاوا اسکریپت) که در طراحی صفحات وب با کدهای HTML ترکیب می‌شوند امکان اجرا داشته باشد یا خیر.

کاربردهای WebView در اندروید

کاربردهای متعددی را می‌توان برای این View ذکر کرد. یکی از مثال‌های بارز و پرکاربرد، نمایش نقشه‌های آنلاین مانند Google Maps است. اگر قصد دارید محل شرکت خود را روی یک نقشه نشان دهید، استفاده از WebView یکی از گزینه‌های ساده و در دسترس است.

یک کاربرد دیگر، نمایش محتوایی است که مرتب در حال تغییر و بروزرسانی است. تعدادی از اپلیکیشن‌های مطرح (ازجمله اینستاگرام) برای نمایش متن "شرایط و قوانین استفاده از اپلیکیشن" از یک وب ویو استفاده می‌کنند. درنتیجه کاربر با مراجعه به این صفحه (اکتیویتی) متن قوانین را به صورت آنلاین و بروز از سرورهای اینستاگرام دریافت می‌کند. مزیت این روش در این است که برای اصلاح و بروزرسانی متن نیازی به انجام تغییرات درون خود برنامه و انتشار نسخه جدید نیست و کاربر بدون نیاز به بروزرسانی اپلیکیشن، هربار که قصد مطالعه قوانین برنامه را داشته باشد، آخرین نسخه را مشاهده می‌کند.

یا فرض کنید یک وب‌سایت فروشگاهی راه اندازی کرده‌اید و به دلایلی (وقت کم، هزینه‌ی زیاد و...) امکان ساخت اپلیکیشن کامل آنرا ندارید. در اینجا به راحتی می‌توان همان وب‌سایت را در قالب یک اپلیکیشن موبایلی به کاربر عرضه کرد. درست مانند این است که کاربر وارد یک مرورگر شده و آدرس وب‌سایت شما را وارد کند. تنها تفاوت در این است که نیازی به وارد کردن آدرس نیست و به محض اجرای برنامه، وب‌سایت لود می‌شود.

نکته: استفاده از این قابلیت برای نمایش یک وب‌سایت کامل در قالب یک اپلیکیشن موبایلی، راهکار استانداردی نیست و تجربه کاربری (UX) قابل قبولی را رقم نمی‌زند. هنگامی از این راهکار استفاده کنید که هیچ گزینه‌ی دیگری روی میز نباشد! در این صورت وب‌سایت باید برای انواع صفحات نمایش بهینه شده باشد، یعنی پیاده سازی طراحی واکنشگرا (Responsive) الزامی است.

در ادامه و در قالب یک پروژه شما را با WebView و قابلیت‌های آن آشنا می‌کنم.

ساخت پروژه WebView در اندروید استودیو

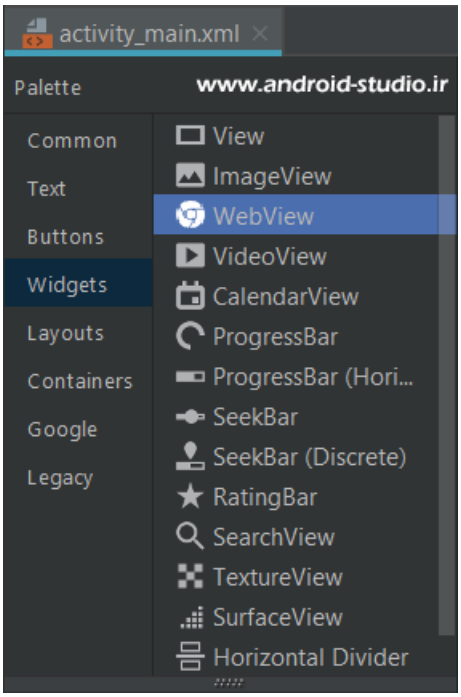
یک پروژه‌ی جدید در اندروید استودیو با نام WebView می‌سازم. اکتیویتی پیش فرض را از نوع Empty Activity انتخاب می‌کنم.

همانطور که در مبحث وب سرویس در اندروید اشاره شد، جهت امکان برقراری ارتباط بین اپلیکیشن و شبکه (اینترنت) نیاز به تعریف مجوز دسترسی (Permission) داریم. بنابراین مجوز مربوطه را به مانیفست پروژه اضافه می‌کنم:

:AndroidManifest.xml

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

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

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

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

</manifest>
```

سپس در لایه‌ی رابط کاربری اکتیویتی (UI) در حالت Design ویجت WebView را روی صفحه‌ی پیش نمایش کشیده (Drag & drop) یا به صورت دستی و در حالت Text، تگ آنرا به Layout اضافه می‌کنم:



:activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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"
    tools:context=".MainActivity">

    <WebView
        android:id="@+id/web_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</RelativeLayout>
```

حالا فایل جاوای اکتیویتی یعنی MainActivity.java را به صورت زیر تکمیل می‌کنم:

```
package ir.android_studio.webview;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.webkit.WebView;

public class MainActivity extends AppCompatActivity {

    private WebView mWebView;

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

        mWebView = findViewById(R.id.web_view);

        String siteUrl = "https://android-studio.ir";
        mWebView.loadUrl(siteUrl);

    }
}
```

ابتدا یک شیء از کلاس WebView با نام دلخواه mWebView ساختم. سپس درون متد onCreate این شیء را به View مربوطه متصل کردم. در اینجا قصد دارم آدرس https://android-studio.ir را در این اکتیویتی نمایش دهم که این امر توسط متد loadUrl صورت می‌پذیرد. همانطور که از نام این متد پیداست، Url مدنظر ما بارگزاری (load) می‌شود. در نهایت محتوای این آدرس به mWebView که به ویجت web_view متصل شده ارسال می‌گردد.

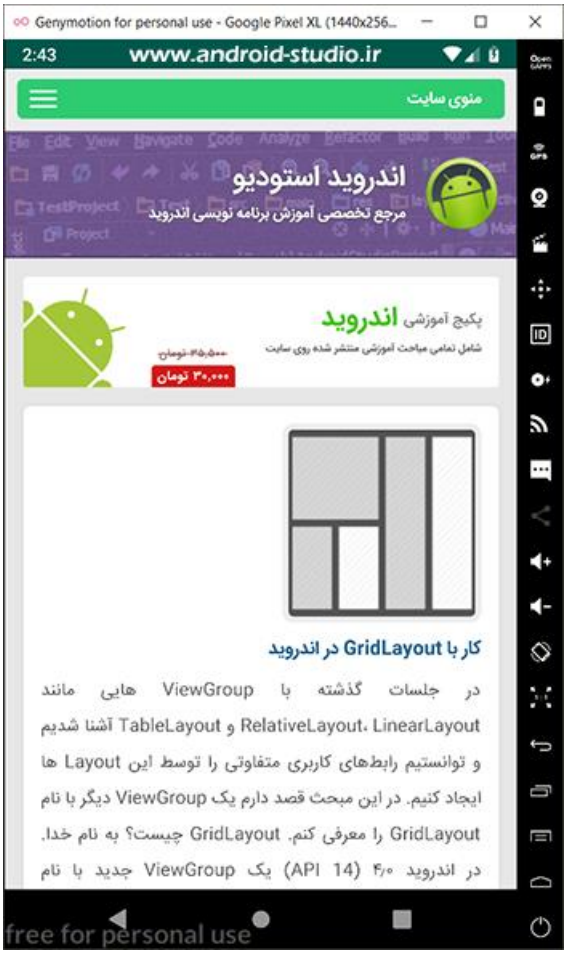
پروژه را اجرا می‌کنم. اگر شبیه ساز یا دیوایس دسترسی به اینترنت داشته و وب‌سایت نیز در دسترس باشد، باید محتوای آن را نمایش دهد.

صفحه‌ی وب با موفقیت درون اکتیویتی لود شده و می‌توانم به بالا و پایین اسکرول کنم.

اگر هدف من از ساخت این اپلیکیشن، تبدیل وب‌سایت به یک برنامه‌ی موبایلی باشد شاید تنها کار لازم حذف ActionBar است که در مبحث آموزش ساخت Toolbar در اندروید با نحوه‌ی انجام آن آشنا شدیم. کافیست در فایل styles.xml استایل مربوط به Theme متریال پروژه را به Theme.AppCompat.Light.NoActionBar تغییر دهم:

:styles.xml

```
<style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
</style>
```

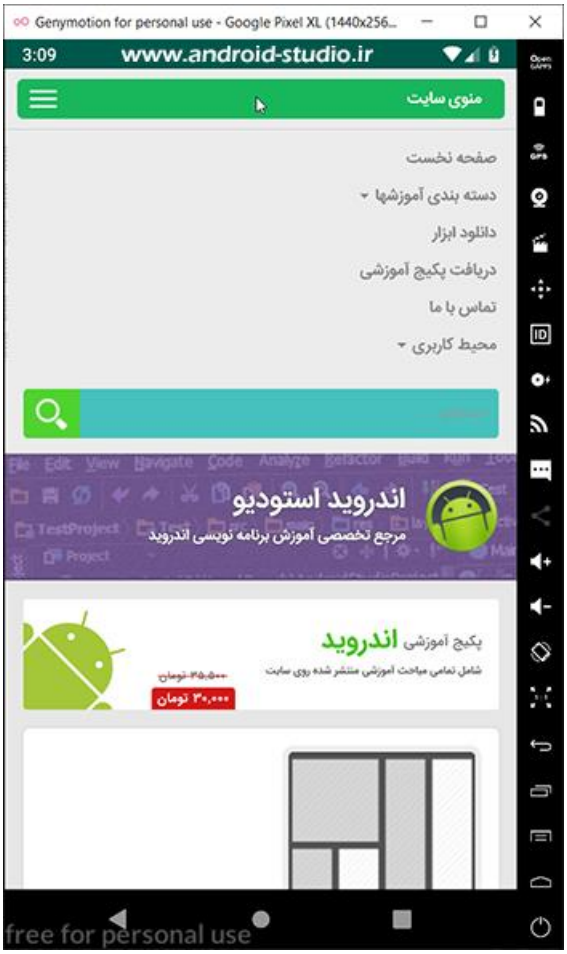


بجز loadUrl متدهای دیگری مانند ()canGoBack، ()canGoForward، ()clearHistory، ()getUrl، ()getTitle و... قابل استفاده است که در صورت نیاز می‌توانید در مورد هرکدام جستجو کنید. در ابتدای مبحث گفتیم در WebView امکان کنترل و اعمال تنظیمات بر روی اجزای دریافتی از صفحه‌ی وب را داریم.

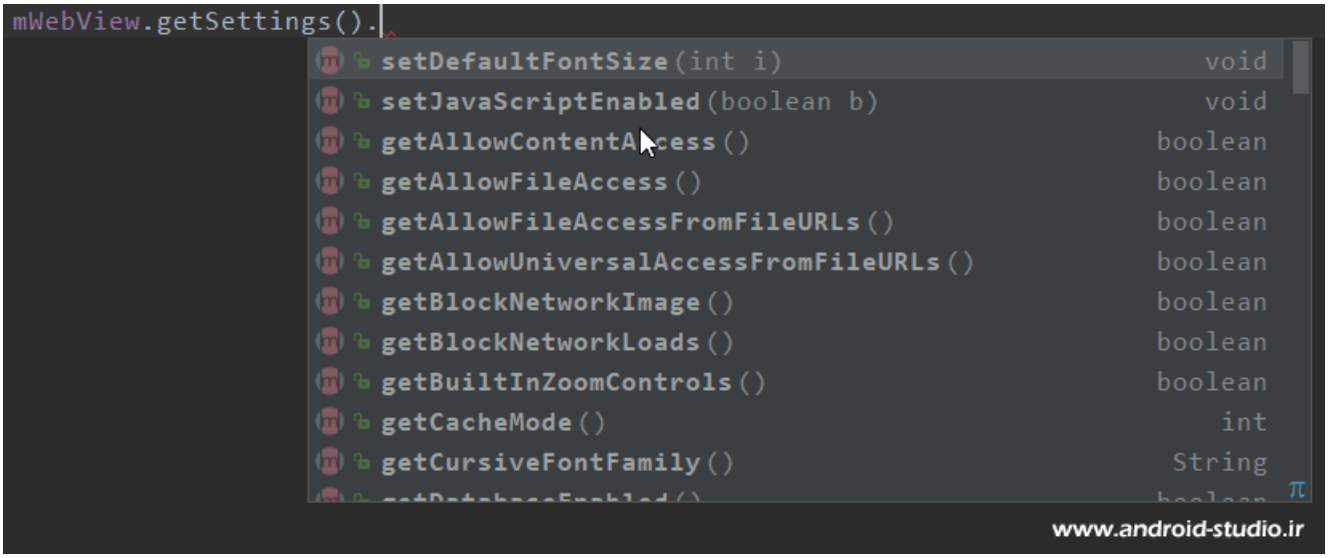
به عنوان مثال اجرای کدهای جاوا اسکریپت (جاوا اسکریپت را با جاوا اشتباه نگیرید) به صورت پیش فرض غیر فعال است که این امر باعث می‌شود منوی سایت من باز نشود. زیرا باز و بسته شدن زیرمنوها توسط کتابخانه‌ی جی‌کوئری (jQuery) که با جاوا اسکریپت نوشته شده مدیریت و اجرا می‌شود.

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

    mWebView = findViewById(R.id.web_view);

    String siteUrl = "https://android-studio.ir";
    mWebView.loadUrl(siteUrl);

    mWebView.getSettings().setJavaScriptEnabled(true);

}
```

با استفاده از متد getSettings() امکان اجرای تنظیمات وسیعی وجود دارد که برای هرکدام یک متد مشخص تهیه شده. همانطور که در کد بالا ملاحظه می‌کنید با استفاده از متد setJavaScriptEnabled و مقدار true برای آن، کدهای جاوا اسکریپت فعال (Enable) می‌شوند. مجدد پروژه را Run یا Apply Changes می‌کنم تا تغییرات روی شبیه ساز بروزرسانی شود:



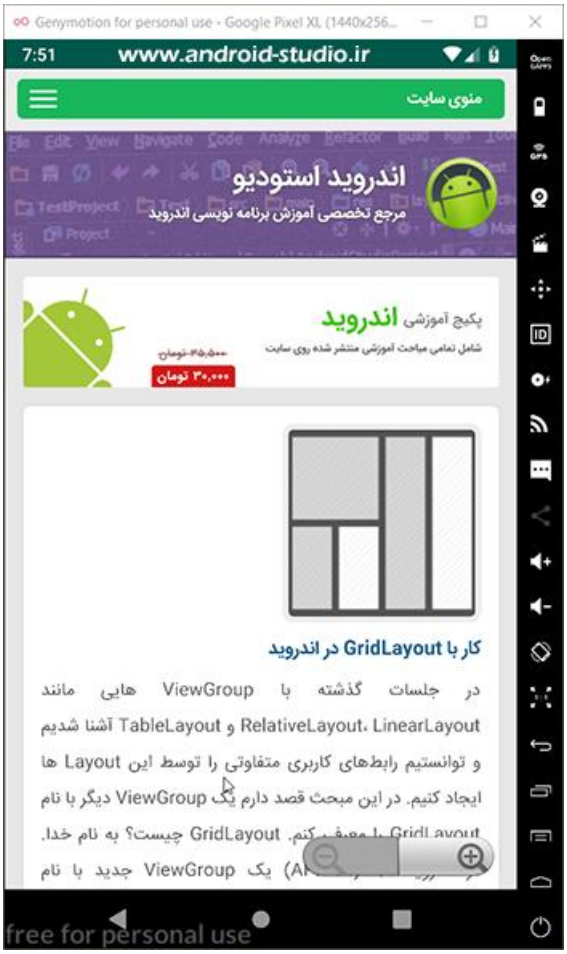
متدهای دیگری نیز در اختیار توسعه دهنده قرار گرفته تا بتواند بر حسب نیاز خود، سایر قسمت‌ها را شخصی سازی کند:



برای مثال اولین مورد موجود در لیست (setDefaultFontSize) برای تعیین اندازه فونت بکار می‌رود.

یا با استفاده از متدهای زیر، دکمه‌های زوم به صفحه اضافه می‌شوند:

```
mWebView.getSettings().setSupportZoom(true);
mWebView.getSettings().setBuiltInZoomControls(true);
mWebView.getSettings().setDisplayZoomControls(true);
```

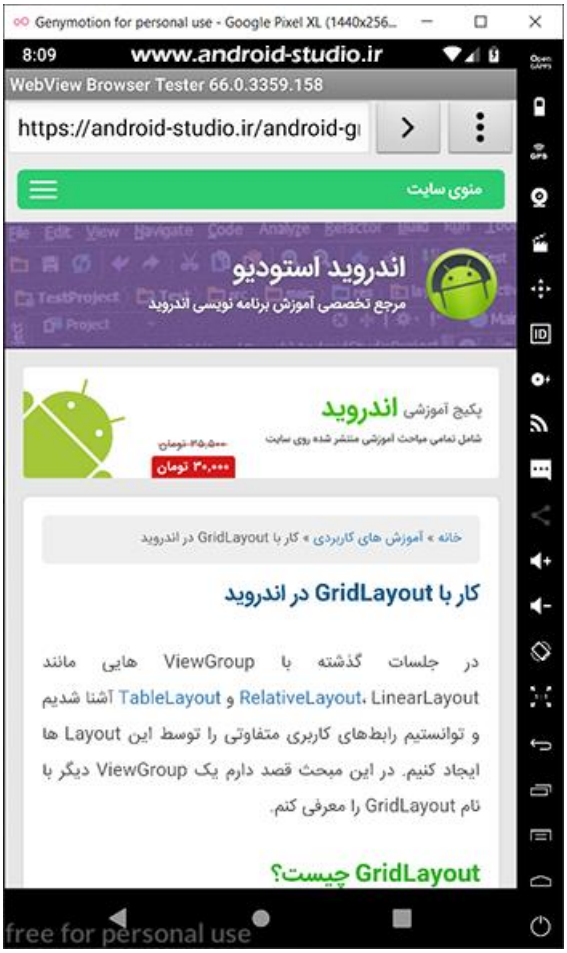


(دکمه‌های Zoom هنگام اسکرول صفحه ظاهر می‌شوند)

تعدادی دیگر از قابلیت‌ها را توسط متد setWebViewClient می‌توان مدیریت کرد. جهت مدیریت بهتر کدها ابتدا یک کلاس داخلی با نام دلخواه mWebViewClient درون اکتیویتی و بعد از بلاک مربوط به متد onCreate می‌سازم که از کلاس WebViewClient ارث بری می‌کند:

```
public class MainActivity extends AppCompatActivity {

    private WebView mWebView;

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

        mWebView = findViewById(R.id.web_view);

        String siteUrl = "https://android-studio.ir";
        mWebView.loadUrl(siteUrl);

        mWebView.getSettings().setJavaScriptEnabled(true);

        mWebView.getSettings().setSupportZoom(true);
        mWebView.getSettings().setBuiltInZoomControls(true);
        mWebView.getSettings().setDisplayZoomControls(true);

    }

    private class mWebViewClient extends WebViewClient {


    }

}
```

اولین متدی که به این کلاس اضافه می‌کنم مربوط به مدیریت کلیک روی لینک‌های موجود در صفحه‌ی وب است. روی لینک یکی از مطالب وب‌سایت کلیک می‌کنم:



مشاهده می‌کنید صفحه‌ی جدید در قالب یک مرورگر با نام WebView Browser Tester باز شده که چندان مطلوب نیست. متد shouldOverrideUrlLoading لینک‌ها را مستقیما درون خود وب‌ویو لود کرده و به مرورگر دیگری منتقل نمی‌شود.

```
private class mWebViewClient extends WebViewClient {

    shou
public boolean shouldOverrideUrlLoading(W… WebViewClient
public boolean shouldOverrideUrlLoading(WebView view, String url) {...}
public boolean shouldOverrideKeyEvent(Web… WebViewClient
public WebResourceResponse shouldInterceptRequest WebVi…
public WebResourceResponse shouldInterceptRequest WebVi…
Fetching Documer
www.android-studio.ir
```

از لیست متدی را انتخاب می‌کنم که پارامترهای ورودی آن از نوع WebView با نام view و String با نام url هستند. این متد با این دو پارامتر مدتیست از سوی اندروید منقضی یا Deprecated تلقی شده با اینحال همچنان استفاده می‌شود.

```
private class mWebViewClient extends WebViewClient {

    @Override
    public boolean shouldOverrideUrlLoading(WebView view, String url) {

        view.loadUrl(url);
        return true;

    }
}
```

متد را به اینصورت اصلاح و تکمیل کردم. حالا با لمس یا کلیک روی هر لینک، آدرس موردنظر در قالب url توسط loadUrl لود و به view منتقل شده که باعث می‌شود صفحه‌ی جدید مستقیما درون WebView نمایش داده شود.

قبل از اجرا و تست این متد، دو متد پرکاربرد دیگر را معرفی می‌کنم. متدهای onPageStarted و onPageFinished به ترتیب، زمان بارگزاری صفحه و پایان این پروسه را برمی‌گردانند.

```
private class mWebViewClient extends WebViewClient {

    @Override
    public boolean shouldOverrideUrlLoading(WebView view, String url) {

        view.loadUrl(url);
        return true;

    }

    @Override
    public void onPageStarted(WebView view, String url, Bitmap favicon) {
        super.onPageStarted(view, url, favicon);
    }

    @Override
    public void onPageFinished(WebView view, String url) {
        super.onPageFinished(view, url);
    }

}
```

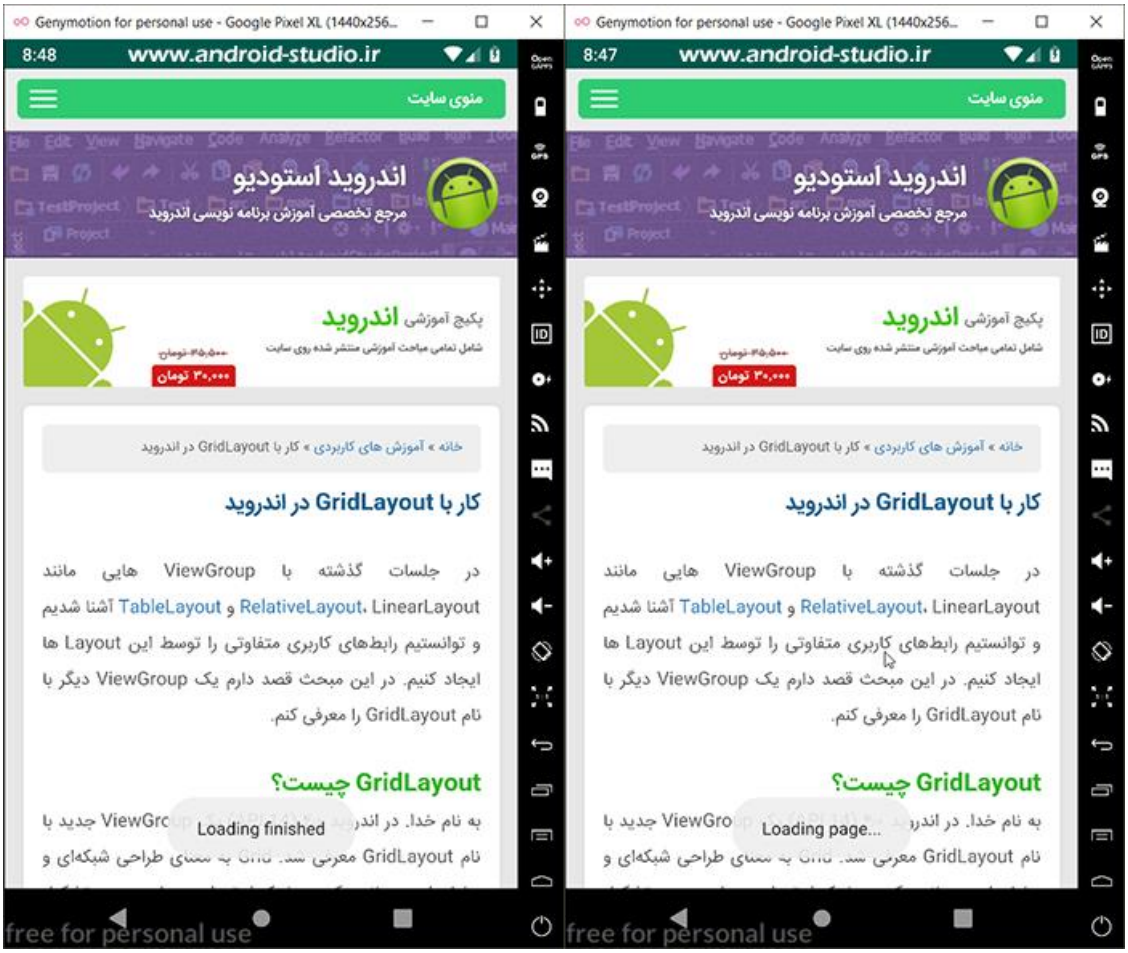
برای مثال می‌توان در متد onPageStarted یک ProgressBar را اجرا کرده و در onPageFinished آنرا متوقف نمود. با اینحال برای درک بهتر و همچنین به حاشیه نرفتن مبحث آموزشی، از Toast استفاده می‌کنم:

```
private class mWebViewClient extends WebViewClient {

    @Override
    public boolean shouldOverrideUrlLoading(WebView view, String url) {

        view.loadUrl(url);
        return true;

    }

    @Override
    public void onPageStarted(WebView view, String url, Bitmap favicon) {

        Toast.makeText(MainActivity.this, "Loading page...", Toast.LENGTH_SHORT).show();

    }

    @Override
    public void onPageFinished(WebView view, String url) {

        Toast.makeText(MainActivity.this, "Loading finished", Toast.LENGTH_SHORT).show();

    }

}
```

در نهایت متد setWebViewClient را درون onCreate تعریف کرده و ورودی آنرا کلاس mWebViewClient قرار می‌دهم:

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

    mWebView = findViewById(R.id.web_view);

    String siteUrl = "https://android-studio.ir";
    mWebView.loadUrl(siteUrl);

    mWebView.getSettings().setJavaScriptEnabled(true);

    mWebView.getSettings().setSupportZoom(true);
    mWebView.getSettings().setBuiltInZoomControls(true);
    mWebView.getSettings().setDisplayZoomControls(true);

    mWebView.setWebViewClient(new mWebViewClient());

}
```

پروژه را اجرا کرده و روی یک لینک کلیک می‌کنم:



مشاهده می‌کنید صفحه‌ی جدید درون خود WebView بارگزاری شد نه یک مرورگر. همچنین پیغام‌های بارگزاری (Loading page...) و اتمام بارگزاری (Loading finished) را می‌بینید که در قالب Toast چاپ شده‌اند.

سورس کامل MainActivity.java:

```
package ir.android_studio.webview;

import androidx.appcompat.app.AppCompatActivity;

import android.graphics.Bitmap;
import android.os.Bundle;
import android.webkit.WebView;
import android.webkit.WebViewClient;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    private WebView mWebView;

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

        mWebView = findViewById(R.id.web_view);

        String siteUrl = "https://android-studio.ir";
        mWebView.loadUrl(siteUrl);

        mWebView.getSettings().setJavaScriptEnabled(true);

        mWebView.getSettings().setSupportZoom(true);
        mWebView.getSettings().setBuiltInZoomControls(true);
        mWebView.getSettings().setDisplayZoomControls(true);

        mWebView.setWebViewClient(new mWebViewClient());

    }

    private class mWebViewClient extends WebViewClient {

        @Override
        public boolean shouldOverrideUrlLoading(WebView view, String url) {

            view.loadUrl(url);
            return true;

        }

        @Override
        public void onPageStarted(WebView view, String url, Bitmap favicon) {

            Toast.makeText(MainActivity.this, "Loading page...", Toast.LENGTH_SHORT).show();

        }

        @Override
        public void onPageFinished(WebView view, String url) {

            Toast.makeText(MainActivity.this, "Loading finished", Toast.LENGTH_SHORT).show();

        }

    }

}
```

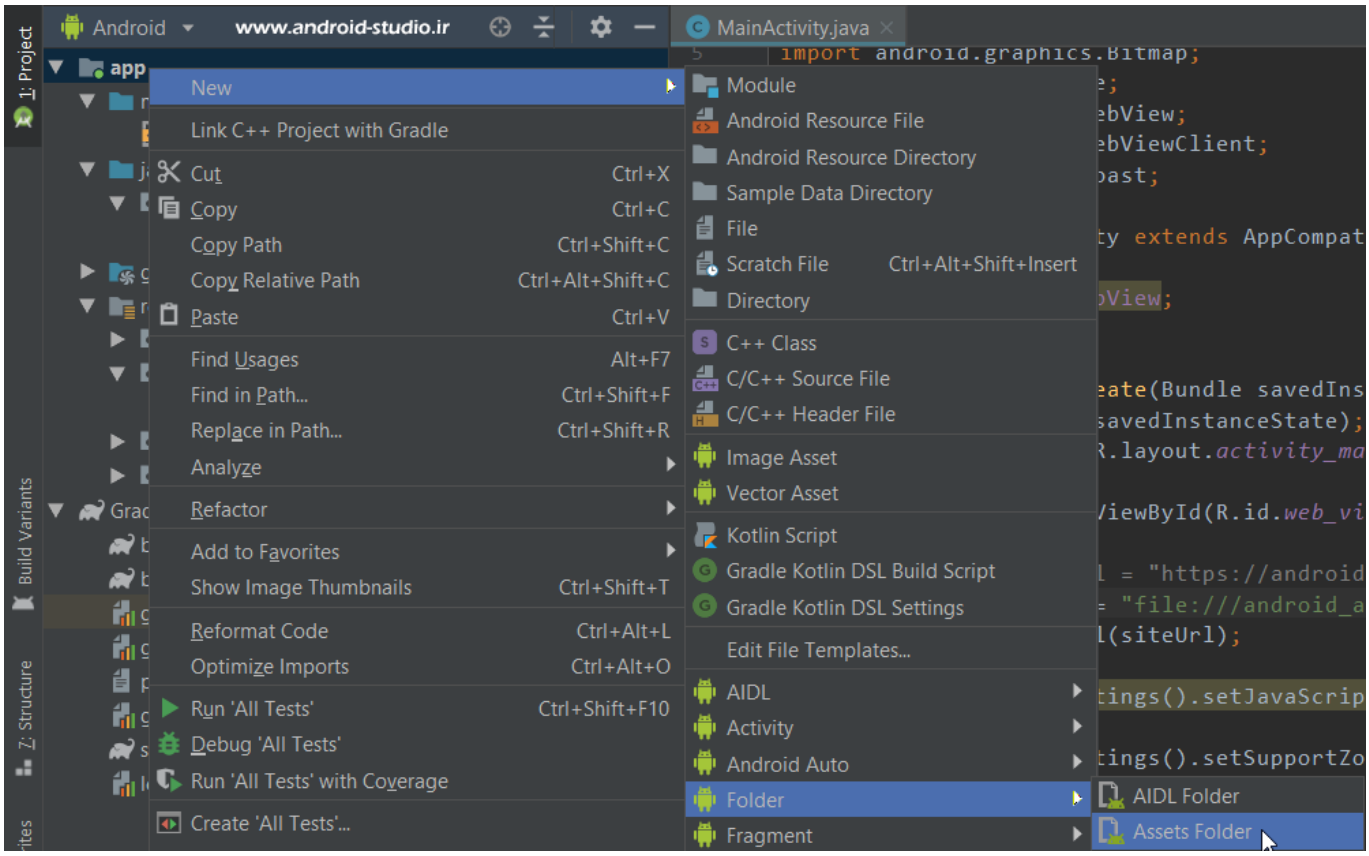
نمایش صفحات HTML محلی و آفلاین در WebView

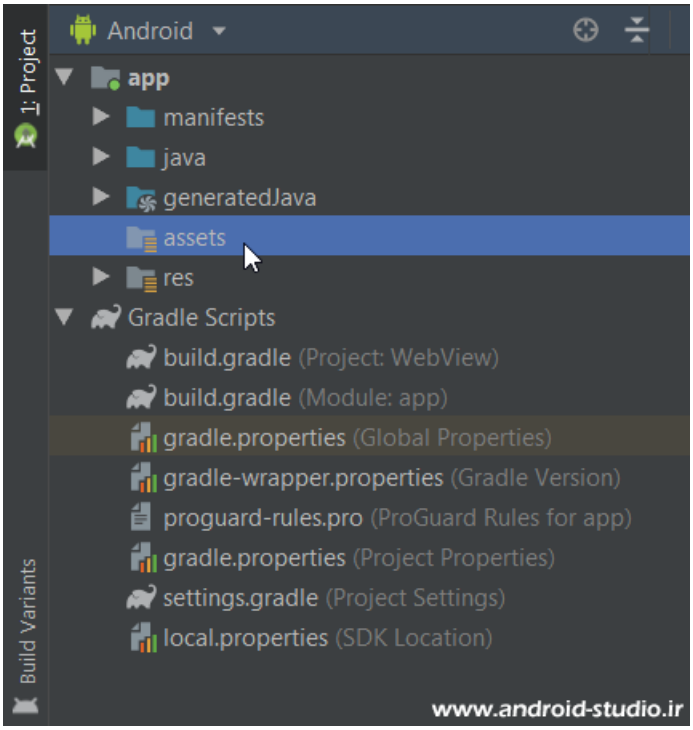
در قسمت قبل محتوای مدنظر ما از یک url و بصورت آنلاین دریافت می‌شد که مستلزم برقراری ارتباط اینترنتی بود. اما ممکن است در مواردی نیاز به نمایش محتوایی با فرمت وب و HTML بصورت آفلاین داشته باشیم. به عنوان مثال طراحی یک یا چند صفحه‌ی خاص از اپلیکیشن برای توسعه دهنده‌ای که با اصول طراحی صفحات وب نیز آشناست، ممکن است پیاده سازی آن در فرمت HTML و CSS ساده‌تر از کار با Layout های اندروید باشد. یا ممکن است شخص بخواهد یک اپلیکیشن کتابچه را به‌طور کامل توسط صفحات وب پیاده‌سازی کند. محتوای این صفحات، ایستا (استاتیک) است و نیازی به اتصال به سرور نیست.

در این روش فایل‌های مرتبط با صفحات وب را در فولدر assets پروژه قرار می‌دهیم. این فولدر به صورت پیش‌فرض وجود ندارد. برای ساخت آن در قسمت نمایش ساختار پروژه روی app راست کلیک کرده و مسیر

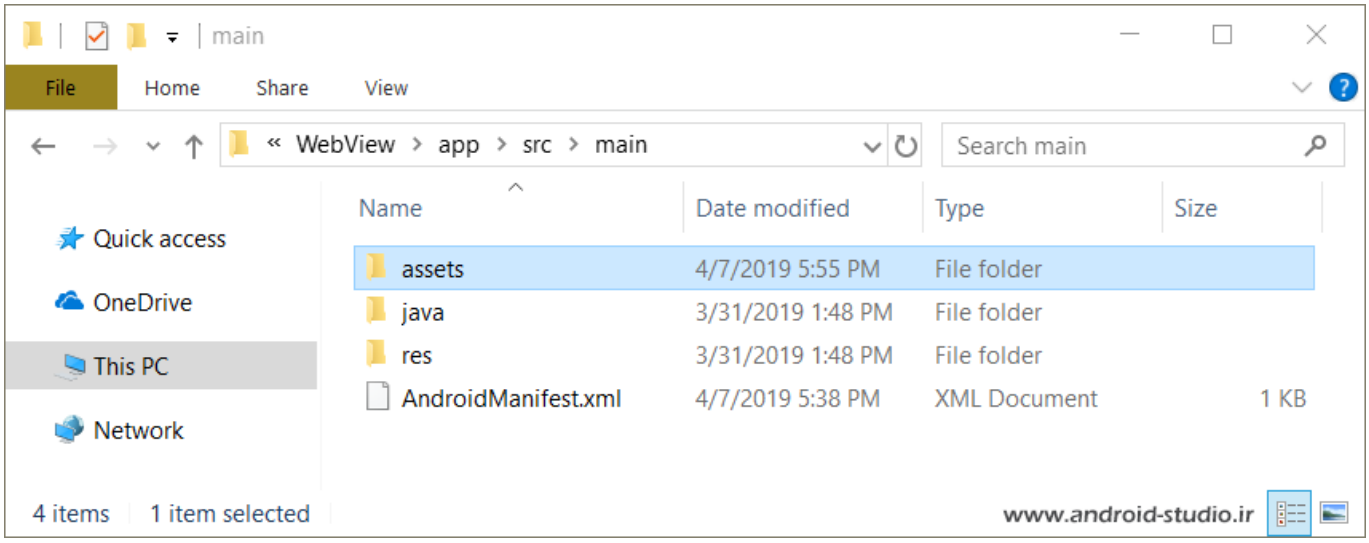
New > Folder > Assets Folder

را دنبال کرده و در پنجره‌ی باز شده Finish بزنید تا فولدر ساخته شود:

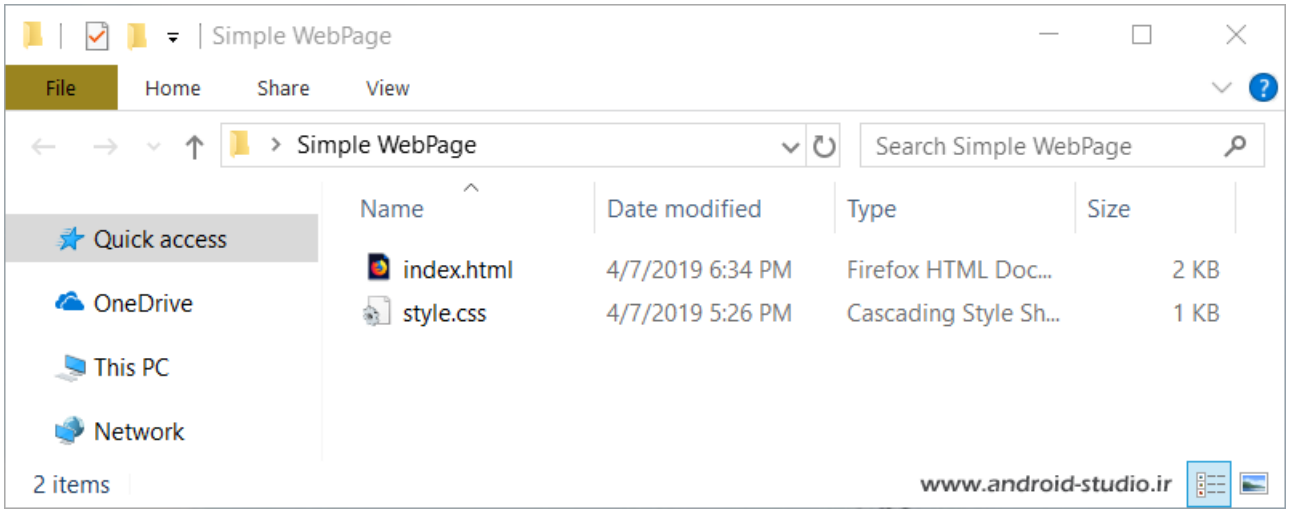




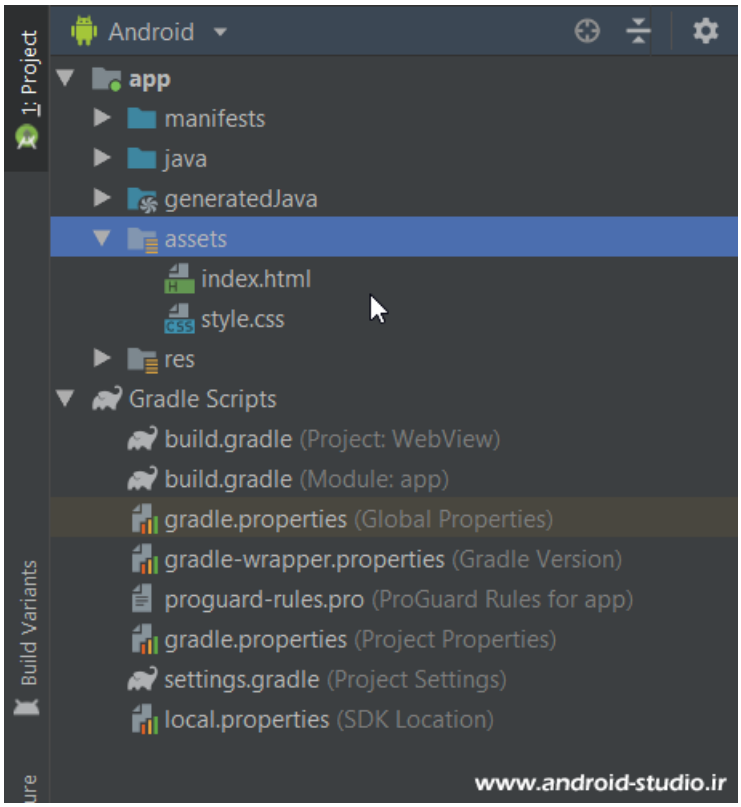
همچنین می‌توان این کار را در خارج از محیط اندروید استودیو و در محل قرارگیری پروژه، این فولدر را اضافه کرد:



من یک صفحه‌ی ساده‌ی وب ساختم که شامل یک فایل html و یک css است:



این دو فایل را کپی کرده و در محیط اندروید استودیو یا File explorer سیستم عامل درون فولدر paste می‌کنم:



در مرحله آخر برای اجرای این فایلِ محلی بجای url وب‌سایت، آدرس زیر را در siteUrl جایگزین آدرس وب‌سایت می‌کنم:

```
String siteUrl = "file:///android_asset/index.html";
```

به اینصورت فایل index.html موجود در فولدر assets پروژه اندرویدی در WebView اجرا می‌شود:

در این پروژه صرفا جهت آشنایی با نحوه‌ی اضافه کردن صفحات وب به اپلیکیشن یک صفحه‌ی HTML بسیار ساده را استفاده کردم اما پیچیده‌ترین صفحات وب (شامل متن، تصویر، کدهای جاوا اسکریپت و..) را به تعداد زیاد و نامحدود می‌توان در اپلیکیشن‌های اندرویدی بکار برد.

مطالعه‌ی بیشتر:

https://developer.android.com/reference/android/webkit/WebView

https://developer.android.com/reference/android/webkit/WebSettings

https://developer.android.com/guide/webapps/webview

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