

به نام خداوند جان و خرد

# برنامه نویسی موبایل با C#

با استفاده از

## Xamarin.Android

مهندس سید منصور عمرانی

انتشارات پندار یاری

سرشناسه : عمرانی، سیدمنصور، ۱۳۵۶ -  
 عنوان و نام پدیدآور : برنامه نویسی موبایل با #C با استفاده از Xamarin.Android  
 مشخصات نشر : تهران : پندار پارس، ۱۳۹۴.  
 مشخصات ظاهری : ۱۵۴ ص.: مصور، جدول .  
 شابک : 978-600-6529-77-6 : ۱۴۰۰۰۰ ریال  
 وضعیت فهرست نویسی : فیپاک مختصر  
 یادداشت : فهرستنویسی کامل این اثر در نشانی: <http://opac.nlai.ir> قابل دسترسی است  
 شماره کتابشناسی ملی : ۳۷۹۶۳۵۱

### انتشارات پندار پارس



دفتر فروش: انقلاب، ابتدای کارگر جنوبی، کوی رشتچی، شماره ۱۴، واحد ۱۶ [www.pendarepars.com](http://www.pendarepars.com)  
 تلفن: ۶۶۵۷۲۳۳۵ - تلفکس: ۶۶۹۲۶۵۷۸ همراه: ۰۹۲۱۴۳۷۱۹۶۴ [info@pendarepars.com](mailto:info@pendarepars.com)



نام کتاب : برنامه نویسی موبایل با #C با استفاده از Xamarin.Android  
 ناشر : انتشارات پندار پارس  
 تألیف : Mark Reynolds  
 ترجمه : سید منصور عمرانی  
 چاپ نخست : اردیبهشت ۹۴  
 شمارگان : ۵۰۰ نسخه  
 چاپ، صحافی : روز  
 قیمت : ۱۴۰۰۰ تومان  
 شابک : ۹۷۸-۶۰۰-۶۵۲۹-۷۷-۶

•••••  
 \*هرگونه کپی برداری، تکثیر و چاپ کاغذی یا الکترونیکی از این کتاب بدون اجازه ناشر تخلف بوده و پیگرد قانونی دارد \*

## فهرست

فصل ۱. آناتومی یک برنامه‌ی Android	۱
سکوی Android	۱
Linux	۱
کتابخانه‌های بومی	۲
ماشین اجرای Android	۲
Android Application Framework	۲
برنامه‌ها	۲
بسته‌ی اندروید (.apk)	۳
مانیفست برنامه	۳
نسخه‌های اندروید	۴
اجزای یک برنامه‌ی اندروید	۵
Activity	۵
چرخه‌ی حیات یک activity	۵
وضعیت‌های یک activity	۵
رویدادهای یک activity	۶
سرویس	۷
فراهم کننده‌ی محتوا (Content Provider)	۷
دریافت کنندگان پیام‌های Broadcast	۷
View و ViewGroup	۸
روش‌های مختلف ایجاد View	۸
المان‌های واسط کاربر (UI widget) ها	۸
طرح‌های رایج	۸
طرح‌های Adapter	۹
فایل XML طرح واسط کاربر	۱۰
تگ‌ها و ویژگی‌ها	۱۱
شناسه‌ها (ID)	۱۱
استفاده از طرح‌های XML توسط activity ها	۱۱
Intent	۱۲
منابع	۱۲
فایل R.java	۱۲
خلاصه	۱۳
فصل ۲. معماری Xamarin.Android	۱۵
چرا Xamarin.Android؟	۱۵
نقاط قوت	۱۶
نقاط ضعف	۱۶
Mono چیست؟	۱۷
Mono و Dalvik در کنار هم	۱۷
واسط بومی جاوا یا JNI	۱۸
اشیا نظیر (peer objects)	۱۸
بسته‌بندی برنامه‌های Xamarin.Android	۱۹

۱۹	سیستم مقیدسازی اندروید
۲۰	اصول رعایت شده در طراحی Xamarin.Android
۲۰	خصوصیت‌های C#
۲۱	نماینده‌ها
۲۱	ثوابت اندروید و نوع‌های شمارش‌پذیر C#
۲۲	محیط‌های توسعه
۲۲	Xamarin Studio
۲۳	Xamarin برای Visual Studio
۲۳	مقایسه‌ی محیط‌های توسعه
۲۴	سازگاری
۲۴	خلاصه
۲۵	فصل ۳. ایجاد یک برنامه‌ی اندروید: « مکان‌های مورد علاقه »
۲۵	برنامه‌ی نمونه
۲۶	نصب Xamarin.Android
۲۸	ایجاد برنامه
۲۸	محیط توسعه‌ی Xamarin Studio
۲۹	پنجره‌ی Project Options
۳۰	مشخص کردن فریم‌ورک مقصد
۳۱	مشخص کردن نام بسته و آیکنی برای برنامه
۳۱	Activity اولیه
۳۲	اجرا و اشکال‌زدایی برنامه
۳۵	ایجاد و سفارشی کردن شبیه‌سازها
۳۷	استفاده از شبیه‌ساز x86
۳۷	اجرا و اشکال‌زدایی برنامه توسط یک رسانه‌ی اندرویدی واقعی
۳۸	فعال کردن USB debugging
۳۸	نصب یک درایور USB
۳۸	اجرای برنامه‌های اندروید روی یک دستگاه واقعی
۳۸	پشت صحنه
۳۸	اشیا نظیر
۳۹	فایل AndroidManifest.xml
۴۰	خلاصه
۴۱	فصل ۴. ایجاد سرویس ذخیره‌سازی اطلاعات
۴۲	ایجاد کلاس موجودیت POI
۴۳	ایجاد واسط انتزاعی سرویس ذخیره‌سازی مکان‌ها
۴۴	پیاده‌سازی سرویس ذخیره‌سازی مکان‌ها
۴۵	استفاده از Xamarin.Android NUnitLite
۴۶	آماده‌سازی تست‌ها
۴۷	ایجاد متدهای تست
۴۷	تست ایجاد یک مکان جدید
۴۸	تست به روزرسانی یک مکان از پیش موجود

۵۰	تست حذف یک مکان از پیش موجود
۵۱	اجرای آزمون‌ها
۵۲	Json.NET
۵۳	دانلود Json.NET
۵۳	پیاده‌سازی و آزمایش متدهای سرویس POIJsonService
۵۵	پیاده‌سازی قابلیت کش کردن اشیا POI
۵۶	پیاده‌سازی متد SavePOI()
۵۸	پیاده‌سازی متد GetPOI()
۵۸	پیاده‌سازی متد DeletePOI()
۵۹	خلاصه
۶۱	<b>فصل ۵. افزودن یک نمای لیستی</b>
۶۱	ایجاد نمای لیستی برنامه‌ی POI
۶۶	افزودن RelativeLayout
۶۷	افزودن کنترل ImageView
۶۷	افزودن کنترل LinearLayout
۶۸	افزودن دو کنترل TextView برای نام و آدرس مکان مورد علاقه
۶۹	افزودن کنترل TextView برای فاصله‌ی مکان POI
۶۹	<b>پر کردن کنترل ListView</b>
۷۰	استفاده‌ی اشتراکی از یک منبع داده‌ی IPOIDataService
۷۱	دسترسی‌ها
۷۲	<b>ایجاد آداپتور POIListViewAdapter</b>
۷۲	نوشتن یک سازنده
۷۳	پیاده‌سازی خصوصیت Count
۷۳	پیاده‌سازی متد getItem()
۷۳	پیاده‌سازی ایندکسر
۷۳	پیاده‌سازی متد getView()
۷۴	استفاده‌ی مجدد از سطرها
۷۴	پر کردن View با مشخصات رکورد
۷۵	<b>مقیدسازی و استفاده از POIListViewAdapter</b>
۷۵	<b>تعریف اکشن‌هایی برای برنامه در ActionBar دستگاه اندروید</b>
۷۶	تعریف فایل menu.xml
۷۷	تنظیم منوها در onCreateOptionsMenu()
۷۷	اداره کردن رویداد انتخاب اکشن در متد onOptionsItemSelected()
۷۸	<b>تعریف کارت SD برای شبیه‌ساز</b>
۷۸	<b>اجرای برنامه‌ی POIApp</b>
۷۸	Android Device Monitor
۸۰	اداره کردن کلیک شدن رکوردها
۸۱	خلاصه
۸۳	<b>فصل ۶. افزودن Detail View</b>
۸۳	ایجاد نمایی به نام POIDetail
۸۶	کار با کنترل‌های ورود اطلاعات

۸۶	.....POIDetailActivity ایجاد
۸۷	.....detail view مقید کردن تعدادی متغیر به کنترل‌های ورود اطلاعات
۸۸	.....سوئیچ کردن بین activity های برنامه.
۸۸	.....حالت ۱: کلیک شدن دکمه‌ی New برای ایجاد یک مکان جدید
۸۹	.....حالت ۲: کلیک شدن یکی از اشیا POI در لیست مکان‌های نمایش داده شده
۸۹	.....OnCreate() دریافت اطلاعات پاس داده شده به POIDetailActivity در متد
۹۰	.....پر کردن باکس‌های متنی
۹۱	.....افزودن قابلیت ذخیره‌سازی و حذف مکان‌ها
۹۲	.....Delete غیر فعال کردن دکمه‌ی
۹۳	.....نوشتن متد SavePOI()
۹۳	.....نوشتن متد DeletePOI()
۹۴	.....اعتبارسنجی
۹۴	.....استفاده از خصوصیت EditText.Error
۹۶	.....افزودن نمایش پیغامی برای تایید عمل حذف
۹۸	.....نمایش پیام‌هایی برای نتیجه‌ی عملیات
۹۸	.....POIListActivity بازتازه کردن
۹۹	.....جمع‌بندی
۹۹	.....خلاصه
۱۰۱	.....فصل ۷. فراهم کردن قابلیت مکان‌یابی خودکار
۱۰۱	.....Location سرویس
۱۰۲	.....تعریف کردن دسترسی‌های مورد نیاز برنامه
۱۰۳	.....تنظیم شبیه‌ساز
۱۰۴	.....به دست آوردن شی LocationManager
۱۰۵	.....درخواست آگاه‌سازی از تغییر موقعیت دستگاه
۱۰۵	.....پیاده‌سازی واسط ILocationListener
۱۰۶	.....افزودن قابلیت استفاده از سرویس Location به برنامه‌ی POIApp
۱۰۶	.....افزودن قابلیت استفاده از سرویس Location به POIListActivity
۱۰۸	.....افزودن سرویس Location به POIDetailActivity
۱۰۸	.....به روز کردن فاصله‌ی یک شی POI با موقعیت جاری
۱۰۹	.....نوشتن کُد دکمه‌ها
۱۱۲	.....به دست آوردن آدرس یک موقعیت مکانی
۱۱۴	.....آگاه کردن کاربر از مشغول بودن برنامه
۱۱۴	.....اداره کردن تغییر پیکربندی
۱۱۵	.....ذخیره و بازیابی وضعیت activity
۱۱۷	.....جلوگیری از منهدم شدن activity
۱۱۸	.....افزودن قابلیت استفاده از نقشه
۱۱۹	.....هدایت کاربر به برنامه‌ی نقشه
۱۲۰	.....بررسی وجود برنامه‌ی نقشه
۱۲۱	.....خلاصه

۱۲۳	فصل ۸. افزودن قابلیت استفاده از دوربین
۱۲۳	انتخاب مکانیزمی برای استفاده از دوربین موبایل
۱۲۴	دسترسی ها و قابلیت ها
۱۲۵	تنظیم شبیه ساز
۱۲۵	توسعه دادن سرویس داده های برنامه
۱۲۵	تعریف متدی به نام getImageFilename() در واسط IPOIDataService
۱۲۶	پایاده سازی متد getImageFilename() در کلاس POIJsonService
۱۲۶	به روز کردن متد DeletePOI()
۱۲۶	عکس گرفتن توسط POIDetailActivity
۱۲۷	افزودن دکمه ای برای عکس گرفتن
۱۲۸	ایجاد Intent
۱۲۸	بررسی نصب بودن برنامه ای برای استفاده از دوربین
۱۲۸	فراهم کردن اطلاعات اضافی برای شی Intent
۱۲۹	مهیا کردن نام و مسیر فایل
۱۲۹	مشخص کردن حداکثر اندازه ی تصویر
۱۲۹	اجرا کردن Intent
۱۳۰	تکمیل متد NewPhotoClicked()
۱۳۱	پردازش نتیجه ی اجرای Intent
۱۳۲	نمایش عکس های از پیش موجود در POIDetailActivity
۱۳۳	نمایش تصویر اشیا POI در POIListActivity
۱۳۳	خلاصه
۱۳۵	فصل ۹. انتشار برنامه
۱۳۵	گزینه های مختلف توزیع برنامه
۱۳۶	آماده شدن برای انتشار APK
۱۳۷	غیر فعال کردن دیباگ
۱۳۷	فایل AndroidManifest.xml
۱۳۷	فایل AssemblyInfo.cs
۱۳۷	لینک کردن
۱۳۷	گزینه های لینک کردن
۱۳۸	اثرات جانبی linking
۱۳۹	انتخاب ABI های مورد پشتیبانی
۱۴۰	انتشار یک apk امضا شده
۱۴۰	keystroke
۱۴۱	منتشر کردن برنامه توسط Xamarin.Android
۱۴۳	انتشار مجدد
۱۴۴	خلاصه





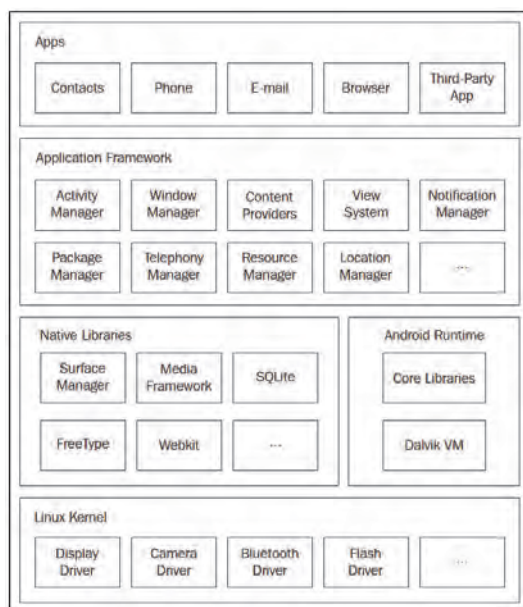
## فصل ۱. آناتومی یک برنامه‌ی Android

تمرکز ما در این کتاب نوشتن برنامه‌های اندروید با C# و Xamarin.Android است. اما نخست به بررسی سکوی Android می‌پردازیم. می‌خواهیم ببینیم Android چیست و چرا با آن می‌توان چنین برنامه‌های موبایلی زیبا و مختلفی نوشت؟ در این فصل به موضوع‌های زیر می‌پردازیم:

- سکوی Android
- برنامه‌های Android (خشت‌های اولیه‌ی ساختمان اندروید)

### سکوی Android

سکوی Android یکی از موفق‌ترین سکوهایی بوده که در سال‌های اخیر برای تلفن‌های همراه ساخته شده است. این سکوی قابلیت‌های بیشماری فراهم می‌کند که برای ساخت برنامه‌های زیبای موبایلی به آنها نیاز است. دیاگرام زیر نمای سطح بالایی از ساختار سیستم عامل Android نشان می‌دهد. در ادامه هر یک از اجزای این ساختار را به اختصار توضیح می‌دهیم.



شکل ۱-۱. ساختار سیستم عامل Android

### Linux

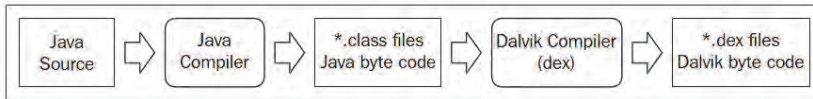
Android سیستم عاملی است که بر پایه‌ی لینوکس نوشته شده و به طور ویژه برای رسانه‌های موبایلی مانند تلفن‌های همراه و تبلت طراحی شده است. نسخه‌های جدید اندروید بر پایه‌ی Linux Kernel Version 3.x و نسخه‌های پیش از Android 4.0 بر پایه‌ی Linux Kernel Version 2.6 کار می‌کنند.

## کتابخانه‌های بومی

Android به همراه خود مجموعه‌ای از کتابخانه‌های بومی دارد که با C/C++ نوشته شده است. این کتابخانه‌ها سرویس‌های گوناگونی فراهم کرده و به طور گسترده‌ای از جوامع اپن سورس گرفته شده‌اند.

## ماشین اجرای Android

در سیستم عامل Android برنامه‌ها با استفاده از یک ماشین مجازی به نام Dalvik VM اجرا می‌شوند که شبیه ماشین مجازی جاوا (JVM) است، اما این ماشین مجازی برای دستگاه‌هایی که حافظه و قدرت پردازشی پایینی دارند بهینه شده است. زبان برنامه‌نویسی برنامه‌های اندروید جاوا است. برای ایجاد یک برنامه‌ی اندروید نخست سورس آن با استفاده از کامپایلر جاوا به بایت کد کامپایل می‌شود. سپس یک مرحله کامپایل دیگر روی آنها اجرا شده و بایت کد جاوا به بایت کد دالویک تبدیل می‌شود تا ماشین مجازی دالویک (Dalvik VM) بتواند آنها را اجرا کند (شکل ۱-۲).



شکل ۱-۲. مراحل ایجاد یک برنامه‌ی اندروید

کامپایلر دالویک همراه کتابخانه‌های اصلی اندروید (core libraries) وجود دارد. این کتابخانه‌ها برای هماهنگی با سکوی جاوای به خصوصی (مانند JEE، JSE یا JME) طراحی نشده‌اند، بلکه به عنوان یک سکوی چند رگه یا hybrid عمل می‌کنند که بیشتر با سکوی JSE منهای کمپوننت‌های AWT و Swing آن که به واسطه کاربر تمرکز دارد هماهنگی دارند. فریم‌ورک برنامه‌های اندروید یا AAF<sup>۱</sup> روش دیگری برای ساخت واسطه کاربر فراهم می‌کند.

## Android Application Framework

Android Application Framework یا فریم‌ورک ساخت برنامه‌های اندرویدی یکی از اجزای اصلی سکوی است که شامل مجموعه‌ای از کتابخانه‌های جاوا است و با آنها می‌توانید واسطه کاربر ایجاد کنید، با اجزای سخت‌افزاری دستگاه مانند دوربین و سرویس مکان‌یابی آن تعامل داشته باشید، فایل‌های مختلف را از روی حافظه‌ی دستگاه بارگذاری کرده و با آنها کار کنید و بسیاری کارهای مفید دیگر.

## برنامه‌ها

بالای پشته‌ی اجزای اندروید برنامه‌های اندروید قرار می‌گیرد، همان اجزایی که چیز ارزشمندی برای کاربر فراهم می‌کنند. اندروید همراه خودش تعدادی برنامه دارد که کارکردهایی پایه‌ای مختلفی فراهم می‌کنند مانند برنامه‌ی مدیریت تماس، چک کردن ایمیل، مرورگر وب و غیره.

<sup>۱</sup> Android Application Framework

اما اندروید موفقیت خود را مرهون برنامه‌هایی است که دیگران برایش نوشته‌اند و افرادی که موبایل یا تبلت اندرویدی دارند آن برنامه‌ها را در دستگاه‌های خود نصب می‌کنند، مانند برنامه‌ی تماشای زنده‌ی مسابقات ورزشی، ویرایش کلیپ ویدئویی که توسط دوربین دستگاه ضبط شده، برنامه‌های مشارکت در شبکه‌های اجتماعی و بسیاری کارهای دیگر.

### بسته‌ی اندروید (apk)

برنامه‌های اندروید در قالب چیزی به نام بسته‌ی اندروید یا Android Package عرضه می‌شود. بسته‌ی اندروید چیزی است که از کامپایل سورس برنامه‌ی اندروید به دست می‌آید و در حقیقت یک فایل فشرده با پسوند apk است که همه‌ی کدهای کامپایل شده و فایل‌های برنامه داخل آن قرار گرفته است. یک بسته‌ی اندروید شامل چنین چیزهایی است:

- اعلامیه یا مانیفست برنامه
- فایل‌های اجرایی دالویک (فایل‌های \*.dex)
- فایل‌های منبع (عکس، متن، xml ...)
- کتابخانه‌های بومی

بسته‌های اندروید مستقیماً از طریق ایمیل، آدرس URL یا از روی کارت حافظه‌ی گوشی قابل نصب هستند. همچنین می‌توان آنها را به طور غیر مستقیم از طریق فروشگاه‌های اینترنتی مانند Google Play دانلود و نصب کرد.

### مانیفست برنامه

همه‌ی برنامه‌های اندروید فایلی به نام AndroidManifest.xml دارند که به آن مانیفست برنامه گفته شده و همه‌ی آنچه را که سیستم عامل اندروید برای اجرای موفقیت‌آمیز برنامه نیاز دارد تعریف می‌کند. از جمله چیزهایی که در مانیفست برنامه تعریف می‌شود می‌توان به موارد زیر اشاره کرد:

- سطح حداقلی API مورد نیاز برنامه<sup>۲</sup>
- قابلیت‌های نرم‌افزاری/سخت‌افزاری مورد نیاز برنامه<sup>۳</sup>
- دسترسی‌های مورد نیاز برنامه<sup>۴</sup>
- صفحه‌ی اولیه‌ی (Android activity) که هنگام اجرای برنامه نمایش داده می‌شود<sup>۵</sup>

<sup>۲</sup> یعنی برنامه در پایین‌ترین حد خود با چه سطحی از API اندروید می‌تواند کار کند و به چه API ای از Android نیاز دارد. مترجم.

<sup>۳</sup> یعنی برنامه به چه قابلیت‌هایی از سیستم عامل اندروید نیاز دارد و می‌خواهد از کدام اجزای سخت‌افزاری دستگاه استفاده کند. مترجم.

<sup>۴</sup> یعنی برنامه به چه دسترسی‌هایی نیاز دارد و از چه دسترسی‌هایی استفاده می‌کند. در حالت عادی برنامه‌ها اجازه ندارند هر کاری انجام بدهند. برای این که یک برنامه بتواند کاری را انجام بدهد که به مجوز دسترسی نیاز دارد باید آن مجوز دسترسی را در مانیفست خود مشخص کند. سپس موقع نصب برنامه، سیستم عامل اندروید این مطلب را به صاحب دستگاه اطلاع داده و از او می‌پرسد آیا مایل است این دسترسی‌ها را به برنامه بدهد یا خیر. مترجم.

- کتابخانه‌های دیگری که افزون بر AAF برنامه به آنها نیاز دارد<sup>۶</sup>
- و مانند اینها

### نسخه‌های اندروید

نسخه‌های اندروید تا حدی عجیب و غریب و گیج کننده است. نسخه‌ی اندروید از یک شماره نسخه، سطح API و اسم مستعار تشکیل می‌شود. مساله این است که هنگام ارجاع به نسخه‌ای از سیستم عامل اندروید، گاهی این اجزا به جای هم استفاده می‌شود.

هر شماره‌ی نسخه به معنی منتشر شدن نسخه‌ی جدیدی از سیستم عامل اندروید است. انتشار نسخه‌ی جدید اندروید می‌تواند برای ارائه‌ی قابلیت‌های جدید یا رفع باگ‌های نسخه‌ی پیشین انجام شود. سطح API نیز نمایانگر قابلیت‌های کتابخانه‌ی اندروید است. هنگام افزایش سطح API، قابلیت‌های جدیدی برای استفاده فراهم می‌شود که برنامه‌نویسان اندروید می‌توانند در نوشتن برنامه‌هایشان از آنها استفاده کنند.

جدول زیر فهرست نسخه‌های مختلف سیستم عامل اندروید را به ترتیب نزولی بر حسب زمان انتشار آنها نشان می‌دهد.

تاریخ انتشار	نام مستعار	سطح API	نسخه‌ی سیستم عامل
2013/10/31	KitKat	19	4.4
2013/07/24	Jelly Bean	18	4.3
2012/11/13		17	4.2 و 4.22
2012/07/09		16	4.1 و 4.11
2011/12/16	Ice Cream Sandwich	15	4.0.3 و 4.0.4
2011/10/19		14	4.0 و 4.01 و 4.02
2011/07/15	Honeycomb	13	3.2
2011/05/10		12	3.1.x
2011/02/22		11	3.0.x
2011/02/02	Gingerbread	10	2.3.3 و 2.3.4
2010/12/06		9	2.3 و 2.3.1 و 2.3.2
2010/05/20	Froyo	8	2.2.x
2010/01/12	Éclair	7	2.1.x
2009/12/03		6	2.0.1
2009/10/26		5	2.0
2009/09/15	Donut	4	1.6

<sup>۵</sup> هر برنامه‌ی اندروید از تعدادی صفحه یا فرم تشکیل می‌شود. یکی از چیزهایی که در مانیفست برنامه مشخص می‌شود این است که کدام یک از این فرم‌ها فرم اولیه‌ی برنامه است که سیستم عامل اندروید هنگام اجرای برنامه باید آن را بارگذاری کند. مترجم.

<sup>۶</sup> یعنی کتابخانه‌های دیگری که این برنامه از آنها استفاده کرده و به آنها وابستگی دارد. مترجم.

## اجزای یک برنامه‌ی اندروید

هر برنامه‌ی اندروید از تعدادی کلاس و مجموعه‌ای از فایل‌های مختلف تشکیل می‌شود. در این قسمت به بررسی اجزای مختلف یک برنامه‌ی اندروید می‌پردازیم که خشت‌های اولیه‌ی ساختمان برنامه را تشکیل داده و برنامه‌ی اندروید بر پایه‌ی آنها ساخته می‌شود.

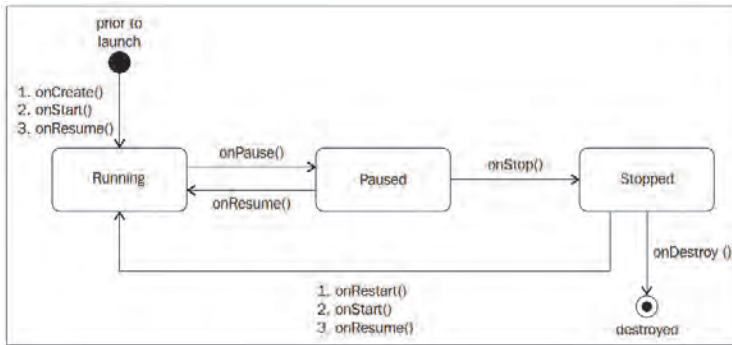
### Activity

یکی از زیربنایی‌ترین بخش‌های یک برنامه‌ی اندروید چیزی به نام activity است. activity به طور ساده به معنی فعالیت یا کاری است که کاربر می‌تواند با برنامه انجام بدهد، مانند نمایش فهرست تماس‌ها، ثبت مشخصات فرد جدیدی در دفترچه‌ی تماس، نمایش مکانی روی نقشه و غیره.

هر برنامه می‌تواند از تعدادی activity تشکیل شود. برای بخش بصری برنامه نیز از View استفاده می‌شود و کاربر از طریق View می‌تواند با activity های برنامه کار می‌کند. اگر با الگوی Model-View-Controller یا MVC آشنا باشید، activity ها در واقع نقش کنترلر را در ساختمان برنامه‌ی اندروید بازی می‌کنند.

### چرخه‌ی حیات یک activity

activity ها چرخه‌ی حیات مشخصی دارند که می‌توان آنها را به شکل وضعیت، گذار یا رویداد توضیح داد. نمودار زیر نمای گرافیکی چرخه‌ی حیات activity را نشان می‌دهد.



شکل ۳-۱. چرخه‌ی حیات یک activity

### وضعیت‌های یک activity

دقت کنید وضعیت‌های activity نشان داده در نمودار بالا تنها یک مفهوم هستند. یعنی activity متغیری یا خصوصیتی به نام state ندارد که وضعیت آن را مشخص کند. وضعیت activity تنها یک مفهوم ضمنی است و برای توضیح بهتر چرخه‌ی حیات activity از آن استفاده کرده‌ایم.

در جدول زیر رفتار یک activity را بر حسب وضعیت آن توضیح داده‌ایم.

وضعیت	توضیح
Running	activity ایجاد و راه‌اندازی شده و برای کاربر قابل مشاهده بوده و کاربر می‌تواند با آن کار کند
Paused	نمای activity توسط یک activity دیگر بلاک شده است (یک activity دیگر روی آن قرار گرفته است)
Stopped	activity برای کاربر قابل مشاهده نیست، اما منهدم نشده است. بلکه وضعیتش به طور موقت حفظ شده و در پس زمینه قرار گرفته است. در چنین وضعیتی activity اجازه‌ی انجام هیچ پردازشی را ندارد.

## رویدادهای یک activity

هنگام تغییر وضعیت activity رویدادهایی برای آن رخ می‌دهد که با آنها کارهای مختلفی می‌توان انجام داد. این رویدادها چنین هستند:

رویداد	اداره‌گر رویداد	زمان فراخوانی	کارهایی که هنگام وقوع این رویداد می‌توان انجام داد
Create	onCreate	هنگام ایجاد شدن activity (مانند اجرای برنامه توسط کاربر)	<ul style="list-style-type: none"> <li>• ایجاد View ها</li> <li>• مقداردهی اولیه‌ی متغیرها</li> <li>• تخصیص منابع بلند مدت</li> </ul>
Start	onStart	پس از onCreate یا onRestart و درست پیش از آن که activity برای کاربر قابل مشاهده گردد	<ul style="list-style-type: none"> <li>• تخصیص دادن منابع</li> </ul>
Resume	onResume	پیش از آن که activity آماده‌ی تعامل با کاربر گردد	<ul style="list-style-type: none"> <li>• آماده کردن اجزای بصری واسط کاربر یا UI widget ها و نمایش دادن آنها</li> <li>• شروع پخش انیمیشن‌ها یا کلیپ‌های ویدئویی</li> <li>• شروع گوش دادن به تغییرات GPS</li> </ul>
Pause	onPause	زمانی که نمای activity (مثلا به دلیل قرار گرفتن view یک activity دیگر روی آن) بلاک شده و قابل تعامل و دریافت ورودی کاربر نیست	<ul style="list-style-type: none"> <li>• commit نمودن تغییرات ذخیره نشده</li> <li>• متوقف کردن موقت انیمیشن‌ها یا کلیپ‌های ویدئویی</li> <li>• متوقف کردن گوش دادن به تغییرات GPS</li> </ul>
Stop	onStop	هنگامی که نمای activity دیگر برای کاربر قابل مشاهده نیست	<ul style="list-style-type: none"> <li>• آزاد کردن منابع</li> </ul>
Restart	onRestart	هنگامی که activity دوباره به پیش‌زمینه برگردانده می‌شود (مثلا به این دلیل که کاربر دکمه‌ی Back را فشار داده است)	<ul style="list-style-type: none"> <li>• تخصیص دادن منابع</li> </ul>
Destroy	onDestroy	پیش از خاتمه و منهدم شدن activity	<ul style="list-style-type: none"> <li>• پاک‌سازی منابع تخصیص داده شده در onCreate</li> </ul>

یکی از چیزهایی که برنامه‌نویسان اندروید تازه‌کار را گیج می‌کند نحوه‌ی تشخیص دادن جهت دستگاه و تغییر جهت آن از حالت portrait (ایستاده) به landscape (خوابیده) یا بالعکس است. در حالت عادی هنگام تغییر جهت دستگاه، اندروید activity های برنامه را منهدم کرده و دوباره ایجاد می‌کند تا مطمئن شود View ها از مناسب‌ترین چیدمان استفاده خواهند کرد.

اگر در برنامه‌ی خود چنین چیزی را در نظر نگیرید، تغییر جهت دستگاه باعث قطع اجرای برنامه شده و به نوبه‌ی خود می‌تواند به از دست رفتن اطلاعات activity ها منجر شود. در صورتی که برنامه‌ی شما به چنین رفتاری نیاز دارد، می‌توانید هنگام وقوع چنین رویدادی اطلاعات activity را جایی حفظ کنید تا هنگام ایجاد دوباره‌ی activity بتوانید اطلاعات آن را احیا کنید. در فصل ۷ «ساخت برنامه‌های POIApp آگاه از موقعیت» ملاحظات ویژه‌ی اداره کردن وضعیت و دغدغه‌های دیگری که در این رابطه وجود دارد را توضیح می‌دهیم.

## سرویس

سرویس‌ها اجزایی هستند که نیازی به واسط کاربر و در نتیجه تعامل با کاربر ندارند و در پس‌زمینه برای اجرای پردازش‌های مختلف به کار می‌روند. برای نمونه در حین این که کاربر با activity های برنامه مشغول کار است سرویس‌ها می‌توانند داده‌ها را کش کنند، آهنگ و موسیقی پخش کنند یا بدون آن که کار کاربر قطع شده یا با اختلال مواجه شود پردازشی را در پس‌زمینه انجام بدهند.

## فراهم‌کننده‌ی محتوا (Content Provider)

فراهم‌کنندگان محتوا چیزهایی هستند که دسترسی به مخزن مرکزی داده‌ها مانند لیست تماس‌ها را مدیریت می‌کنند. یک فراهم‌کننده‌ی محتوا بخشی از برنامه است که معمولاً برای مدیریت داده‌هایش واسط کاربر هم دارد. همچنین واسط برنامه‌نویسی استاندارد دارد تا برنامه‌های دیگر بتوانند از مخزن او استفاده کنند.

## دریافت‌کنندگان پیام‌های Broadcast

دریافت‌کنندگان پیام‌های Broadcast اجزایی هستند که پیام‌های عمومی اندروید را دریافت کرده و به دنبال آن کاری انجام می‌دهند. معمولاً پیام‌های Broadcast توسط سیستم عامل اندروید هنگام وقوع رویدادهایی مانند کم شدن باتری دستگاه، روشن شدن دوربین، عکس گرفتن، روشن شدن Bluetooth و مانند آن منتشر می‌شود. برنامه‌ها نیز خودشان می‌توانند پیام Broadcast بفرستند. برای نمونه یک Content Provider می‌تواند هنگام به روز شدن اطلاعات (مانند به روز شدن یک Contact)، پیام Broadcast منتشر کند. با وجودی که دریافت‌کننده‌ی پیام Broadcast واسط کاربر ندارد می‌تواند به طور غیر مستقیم وضعیت برنامه را تغییر بدهد (مثلاً نمای دیگری از برنامه را نشان بدهد).

## View و ViewGroup

در برنامه‌های اندروید هر چیزی که دیده می‌شود یک View است. دکمه‌ها، برچسب‌ها، تکست‌باکس‌ها، دکمه‌های رادیویی، لیست‌ها، همه و همه View هستند. برای سازمان‌دهی View ها از ViewGroup استفاده می‌شود و با استفاده از آن می‌توان View ها را به شکل سلسله‌مراتبی دسته‌بندی کرد. در واقع یک ViewGroup یک View ویژه است که برای چیدن View های دیگر کنار هم و ایجاد layout واسط کاربر به کار می‌رود.

### روش‌های مختلف ایجاد View

View ها و ViewGroup ها را به دو شکل می‌توان ایجاد کرد: از طریق کُد یا از طریق فایل XML. در حالت اول برای ایجاد View از توابع API اندروید استفاده شده و برنامه‌نویس خودش View ها را ایجاد کرده و موقعیت و چیدمان آنها را مشخص می‌کند. اما در روش دوم که به آن روش اظهاری یا declarative گفته می‌شود، View ها و چیدمان آنها داخل یک فایل XML تعریف می‌شود. مزایای روش declarative نسبت به روش برنامه‌نویسی چنین است:

- واسط کاربر برنامه را از کدهای برنامه جدا می‌کند
- می‌توانید کُد برنامه را یک بار بنویسید، اما چندین فایل View (مثلا برای دستگاه‌های مختلف) ایجاد کنید.
- می‌توانید با استفاده از ابزارهایی مانند Android Studio و پلاگین Android برای Eclipse حين ساخت واسط کاربر، نمای واقعی آن را هم مشاهده کنید، بدون آن که مجبور باشید پس از هر تغییری هر بار برنامه را کامپایل و اجرا کنید.

با وجودی که من خودم بیشتر مواقع روش declarative را ترجیح می‌دهم، اما به این نتیجه رسیده‌ام در عمل معمولا به ترکیبی از روش اظهاری و برنامه‌نویسی نیاز است.

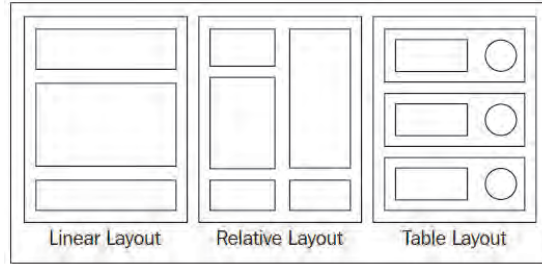
### المان‌های واسط کاربر (UI widget ها)

Android مجموعه‌ی وسیعی از ویجت یا المان واسط کاربر دارد که توسط آنها می‌توانید واسط‌های کاربری زیبایی برای برنامه‌ها ایجاد کنید. همه‌ی این ویجت‌ها View محسوب می‌شوند و با استفاده از ViewGroup های مختلف می‌توان آنها را روی صفحه چید و طرح‌های پیچیده‌ای خلق کرد. در فریم‌ورک اندروید المان‌های واسط کاربر در بسته‌ی android.widget قرار دارند.

### طرح‌های رایج

در فریم‌ورک اندروید تعدادی زیرکلاس برای ViewGroup تعریف شده که هر کدام آنها روش یکتا و مفیدی برای چیدمان و سازمان‌دهی واسط کاربر فراهم می‌کند. در شکل ۴-۱ چند طرح نمونه نشان داده شده که هر کدام برای پاسخ به نیاز متفاوتی استفاده می‌شود.





شکل ۴-۱. انواع layout برای چیدن view ها

طرح	توضیح	سناریو
Linear	اجزا را پشت سر هم در یک سطر افقی یا عمودی نمایش داده و در صورت لزوم میله‌ی اسکرول فراهم می‌کند	زمانی به کار می‌رود که المان بتواند به شکل افقی یا عمودی کش پیدا کند
Relative	موقعیت اجزا را نسبت به هم یا نسبت به پدرشان تنظیم می‌کند	زمانی به کار می‌رود که موقعیت اجزا نسبت به هم (مانند «سمت چپ...» یا «مرزهای المان پدر (مانند «چسبیده به کنار راست...» یا «در مرکز...» تعریف می‌شود
Table	اجزا را داخل خانه‌های یک جدول قرار می‌دهد	زمانی به کار می‌رود که موقعیت اجزا را بتوان به شکل سطر و ستون ترتیب داد. این طرح برای ساخت فرم‌های ورود اطلاعات بسیار مفید است.

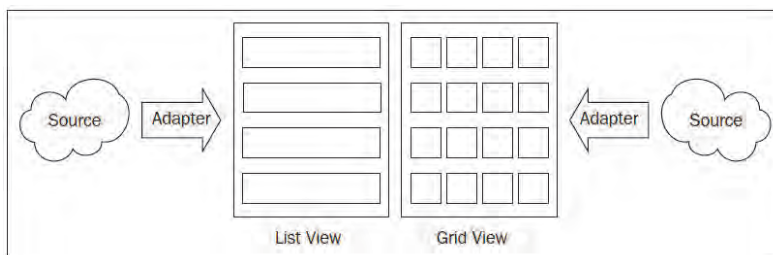
اگر به طرح‌های پیچیده‌تری نیاز دارید، می‌توانید این layout ها را به صورت تو در تو هم استفاده کنید. البته طرح‌هایی که بسیار تو در تو هستند می‌توانند کارایی را کاهش بدهند. از این رو سعی کنید از اتخاذ چنین طرح‌هایی خودداری کنید.

### طرح‌های Adapter

در فریم‌ورک اندروید برای طرح‌هایی که محتوای آنها پویا است (مثلا قرار است لیستی از رکوردهای یک منبع داده نمایش داده شود)، کلاس‌هایی به نام آداپتور (Adapter) فراهم شده که از AdapterView مشتق می‌شوند. دو نمونه از رایج‌ترین آداپتورها عبارتند از:

- List View: این آداپتور آیتم‌ها را به صورت تک ستونی با قابلیت اسکرول نمایش می‌دهد
- Grid View: این آداپتور آیتم‌ها را به صورت شبکه‌ای یا گرید نمایش می‌دهد.

در شکل ۵-۱ نمای صوری آداپتور List View و Grid View نشان داده شده است.



شکل ۵-۱. رایج‌ترین طرح‌های آداپتور: نمای لیستی (List View) و نمای شبکه‌ای (Grid View)

### فایل XML طرح واسط کاربر

در روش تعریف واسط کاربر با استفاده از XML، فریم‌ورک اندروید تگ‌هایی دارد که با آنها می‌توانید انواع و اقسام اجزای واسط کاربر را ایجاد کنید. از نظر مفهومی فایل‌های XML تعریف واسط کاربر شبیه فایل‌های HTML یا فایل‌های XAML فناوری WPF میکروسافت هستند و همان گونه که در فایل‌های html یا xaml با نوشتن تگ‌های مختلف می‌توانید واسط کاربر صفحه‌ی وب یا برنامه‌ی WPF را ایجاد کنید، در اندروید هم می‌توانید با گنجاندن تگ‌های اندروید در یک فایل xml، واسط کاربر برنامه را ایجاد کنید. در کُد زیر یک View ساده با طرح linear حاوی یک تکست‌باکس برای وارد کردن متن و یک دکمه‌ی جستجو نشان داده شده است.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:text="Enter Search Criteria"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/searchCriteriaTextView"/>
    <Button
        android:text="Search"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/searchButton"/>
</LinearLayout>
```

همان گونه که می‌بینید تکست‌باکس با استفاده از تگ <TextView> و دکمه با استفاده از تگ <Button> تعریف شده است. این دو تگ نیز داخل تگ <LinearLayout> قرار داده شده‌اند که از طرح خطی در جهت عمودی برای چیدمان اجزایش استفاده می‌کند.

## تگ‌ها و ویژگی‌ها

سازندگان سیستم عامل اندروید تلاش کرده‌اند نام تگ‌ها و ویژگی‌های آنها در فایل XML با نام کلاس‌ها و خصوصیت‌های متناظر آنها در فریم‌ورک اندروید (برای ایجاد واسط کاربر از طریق کُد) هماهنگی داشته باشد. برای نمونه در مثال پیشین از تگ `<LinearLayout>`، `<TextView>` و `<Button>` استفاده کردیم. به طور مشابه در Android Application Framework کلاس‌هایی به نام `LinearLayout`، `TextView` و `Button` وجود دارد. همچنین مشابه ویژگی `android:text` در تگ `<Button>` متدی به نام `setText()` در کلاس `Button` وجود دارد که با آن می‌توان محتوای دکمه را مشخص کرد.

## شناسه‌ها (ID)

هر `View` (هر المان بصری) می‌تواند شناسه یا ID یکتایی داشته باشد. توسط این شناسه می‌توان از طریق کُد به `View` دسترسی پیدا کرد. برای تعریف شناسه در فایل XML از ویژگی `android:id` استفاده می‌شود. سعی کنید برای شناسه‌ی `View` ها نام کاربرپسندی انتخاب کنید. هنگام انتخاب شناسه‌ی `View` می‌توانید از کاراکترهای ویژه‌ای هم استفاده کنید که مفهوم خاصی برای اندروید دارد. برای نمونه شناسه‌ی زیر را در نظر بگیرید:

```
android:id="@+id/searchButton"
```

در اینجا معنی کاراکتر `@` این است که پارسر اندروید باید با مابقی رشته `(+id/searchButton)` به صورت شناسه‌ی یک منبع برخورد کند (همان گونه که کمی جلوتر خواهیم دید در اندروید شناسه‌ی منابع از نوع `int` است). کاراکتر `+` هم به این معنی است که نام مشخص شده نام یک منبع جدید است و باید به فایل منبع برنامه (فایلی به نام `R.java`) افزوده شود.

چیزی که در اینجا رخ می‌دهد این است که پارسر اندروید `View` تعریف شده با ویژگی `android:id` را به عنوان یک منبع در نظر گرفته و خودش شناسه‌ی آن را تولید می‌کند. برای این کار در فایل `R.java` کلاس استاتیکی به نام `id` ایجاد کرده و برای آن خصوصیت استاتیکی از نوع `int` به نام `searchButton` تعریف می‌کند. شناسه‌ی `View` در حقیقت مقدار `id.searchButton` است که با استفاده از کاراکتر `+` از پارسر اندروید خواسته‌ایم آن را به طور خودکار تولید کند.

```
public static final class id {
    public static final int searchButton=0x7f050002;
}
```

## استفاده از طرح‌های XML توسط activity ها

می‌توان فایل‌های XML مربوط به تعریف واسط کاربر را به سادگی در حین اجرای برنامه بارگذاری کرد. معمولاً این کار داخل اداره‌گر رویداد `onCreate()` در کلاس `activity` با استفاده از متد `setContentView()` انجام می‌شود. به کُد زیر توجه کنید:

```
setContentView(R.layout.main);
```

## Intent

Intent ها پیام‌هایی هستند که در یک برنامه‌ی اندروید می‌توانند برای کمپوننت‌های مختلف فرستاده شوند تا آن کمپوننت‌ها کاری انجام بدهند. مانند موارد زیر:

- اجرای یک activity با مقدار برگشتی
- شروع یا خاتمه دادن اجرای یک سرویس
- اطلاع دادن شرایطی مانند پایین آمدن شارژ باتری دستگاه یا تغییر time zone
- درخواست انجام عملی از یک برنامه‌ی دیگر مانند درخواست نمایش محلی روی نقشه از برنامه‌ی Map یا درخواست عکس گرفتن و ذخیره نمودن عکس از برنامه‌ی Camera

## منابع

نوشتن برنامه‌های اندروید تنها از کدنویسی تشکیل نمی‌شود. یک برنامه‌ی پیشرفته‌ی موبایلی به چیزهای مختلفی نیاز دارد که از میان آنها می‌توان به تصاویر، فایل‌های صوتی، انیمیشن، منو، شیوه و غیره اشاره کرد. فریم‌ورک اندروید توابعی دارد که با آنها می‌توانید انواع و اقسام منابع مختلف را بارگذاری کرده و از آنها در برنامه‌ی اندروید خود استفاده کنید.

## فایل R.java

در یک برنامه‌ی اندروید برای ارجاع به منابع از اعداد صحیح استفاده می‌شود. این اعداد به طور خودکار هنگام افزودن منابع به پروژه‌ی برنامه‌ی اندروید تولید شده و داخل یک فایل جاوا به نام R.java قرار داده می‌شود. در گد زیر فایل R.java یک برنامه‌ی ساده نشان داده شده است:

```
public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon = 0x7f020000;
    }
    public static final class id {
        public static final int myButton = 0x7f050000;
        public static final int searchButton = 0x7f050002;
        public static final int searchCriteriaTextView = 0x7f050001;
    }

    public static final class layout {
        public static final int main = 0x7f030000;
        public static final int search = 0x7f030001;
    }
    public static final class string {
        public static final int app_name = 0x7f040001;
        public static final int hello = 0x7f040000;
    }
}
```